Design and Reference Guide

9.0.1 Release

Copyright © 2025 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

Introduction 1

About the Financial Model 2

Extensible Dimensionality 2

Cube Design Choices within an Application 6

Application Model Design Considerations 7

Standard Cube Design 8

Standard Large Sparse Cube Design 9

Hybrid Cube Design 10

Dynamic Cube Services 12

Dynamic Cubes Overview 13

Dynamic Dimensions Overview 14

Configure a Dynamic Dimension 14

Workspace Assembly Dynamic Dimensions 17

Configure a Dynamic Cube 27

Dynamic Cube Data Binding Types 29
Dynamic Cube Query Performance 35
Dynamic Data Management 37
Extracting Dynamic Dimensions 47
Dynamic Dimensions and Smart Integration Connector 48
Master Data Considerations 51
Dynamic Cube Services API Reference 55
BI Blend 93
Key Elements of BI Blend 94
Relational Blending 94
Relational Blending Types 94
Model Blending 95
Relational Blending API 98
Cache Level Types 99
Relational Model Blending API Functions 100
Formulas 113

Formula Structure 114

Applying Formulas 116

Formula Types 118

Formula Execution 120

Writing Formulas 124

Writing Stored Calculations 133

Avoiding Data Explosion 149

Key Functions 153

Account 153

Consolidation 155

Data 158

Entity 167

Flow 168

FX 169

Journals 172

Member 173

Scenario 176

Status 177

Time 178

Functions 178

Examples of Key Functions in Use 180

Finance Function Types 193

Create Two Accounts 199

Consolidation 200

Example of a Consolidation 202

Launching a Consolidation 203

Calculation Status 206

Currency Translation 208

How the Origin Dimension Works with the Consolidation Dimension 209

Intercompany Elimination 212

Direct and Indirect Eliminations 215

Equity Pickup 216
Entity Aggregation 218
How Entity Aggregation Works 219
Example of an Entity Aggregation 220
Launching an Aggregation 223
Entity Aggregation Input 224
Working With Hybrid Scenarios 225
Data Bindings 225
Data Filters 227
Share Data From the Source Scenario 227
Copy Data From the Source Scenario 228
Navigation 229
About the OneStream Windows Application 229
Launching the OneStream Windows App 230
Defining Server Connections 231
Logging In 232

OnePlace Layout 234
Point of View 241
Docking 247
Benefits 247
Docking Areas 248
Layouts 249
Tabs 251
Pane 253
Floating Window 254
Refreshing Content 255
Undock Content 255
Docking Known Issues 256
Logon Page 256
POV Pane 256
Floating Window 256
Navigate Recent Pages 257

List Box 257

Docking Use Cases 257

USE CASE 258

Filter Runtime Data and Reduce Maintenance with Parameters 266

About Parameters 267

Parameter Types 268

Literal Value Parameters 269

Input Value Parameters 269

Delimited List Parameters 270

Bound List Parameters 271

Member List Parameters 272

Member Dialog Parameters 273

Benefits 273

Ways to Use Parameters 274

Cube Views 275

Data Adapters and Dashboard Components 275

Extensible Documents 276
Books 276
Requirements 277
Definition and Assignment 277
Syntax 277
Best Practices 278
Design 278
Naming 279
About Parameter Definitions 280
Property Types 280
Create a Parameter 281
Specify Literal Value Settings 283
Specify Input Value Parameter Settings 283
Specify Delimited List Settings 284
Specify Bound List Settings 285
Sample Method-Based Parameter 289

Sample SQL-Based Parameter 289
Specify Member List Settings 290
Specify Member Dialog Settings 291
Sample Delimited List Parameter in a Cube View 294
Manage Parameters 296
Assign a Parameter 296
Assign to a Report Subheader 297
Assign to a Cube View Column Header 298
Modify a Parameter 300
Copy a Parameter 300
Delete or Remove a Parameter 301
Rename a Parameter 302
Identify Parameter Assignments 303
Nest Parameters 304
Presenting Data With Extensible Documents 307
Extensible Document Framework 307

About Creating Extensible Documents 308
Document Variable Setup 314
Parameters in Extensible Documents 321
Create an Extensible Document in Microsoft Word 323
Creating an Extensible Document in Text Editor 354
Creating a Document in Microsoft PowerPoint 381
Creating a Document in Microsoft Excel 383
Using Extensible Documents in OneStream 387
Implementing Security 398
Application Security 398
Security Best Practices 399
Import 404
Forms 404
Adjustments 405
Entity Security 405
Relationship Security 405

Change the Relationship Security 407

Security Configurations 412

System Security 416

Managing a OneStream Environment 416

Disable Inactive Users 418

Set the Inactivity Threshold 419

Review Settings 422

Business Rules 424

System Table 424

Business Rules 425

Platform Engines 426

Data Management Automation With PowerShell 429

Instantiation of Standard .NET Classes 429

Using PowerShell Script Editor 429

Configuring PowerShell for the Client API 430

Running Unsigned Scripts 430

Configuration for .Net Framework v.4.0 431

Learning PowerShell 431

Using the Client API in a PowerShell Script 432

Exposing Data Management Automation through OneStream Web API 435

OneStream Web API endpoints: 435

Extracting and Loading Dimensions 437

Extracting Dimension Changes 438

Loading the XML File 442

Common Load Errors and Resolutions 443

Project Extract and Load 445

Project File 445

File Extract 447

Zip Extract 449

File Load .xfProj 450

Zip Load 452

Cubes 453

Dimensions Overview 453

Configurable Dimensions 455

System-defined Dimensions 539

Time Profiles 543

Anatomy of a Time Profile 544

Time Profile Toolbar 544

Profile 546

Time Periods Tab 550

Cubes 554

Structure of a Cube 555

Cube Properties 555

Cube Dimensions 570

Cube Reference 574

Data Management Access Security 582

Integration 584

Foreign Exchange Rates 600
Grid Settings Intersection 600
Lock FX Rates 602
System Changes: Administrator Access to Lock FX Rates 605
Lock and Unlock Rates 606
BRAPI Rule Functions 610
Spreadsheet or Excel Functions 610
Member Filters 612
Member Filter Builder 613
Member Script Abbreviations and Examples 615
Member Script Constants 617
Member Expansion Functions 618
Reverse Order Member Expansions 633
Workflow Member Expansions 638
Member Expansion Where Clause 643
Time Functions 648

Substitution Variables 665
Workflow Substitution Variables 668
Cube View Substitution Variables 672
General Substitution Variables 674
Member Filter Substitution Variables 676
Custom Substitution Variables 678
Samples of Member Filter Functions 678
Workflow 693
Workflow 694
Workflow 694
Workflow Profiles 698
Data Units 733
Workflow Stage Import Methods 748
Workflow Channels 763
Confirmation Rules 770
Certification Questions 775

Data Collection 779
Data Sources 779
Data Source Properties 779
Fixed Files 781
Delimited Files 782
Connector Data Source 782
Data Management Export Sequences 809
Source Dimensions 809
Source Dimension Properties 809
Matrix Settings 813
Numeric Settings 814
Bypass Settings 816
Stored Text Settings 817
Using Excel as a Data Source 817
Forms 818
Forms Channel Workflow 818

Form Allocations 818
Form Templates 824
Transformation Rules 836
Transformation Rules Toolbar 836
General Settings 837
Mapping Types 838
Derivative 843
Logical Operators 850
Derivative Rule Expressions 856
Derivative Types 859
Order 860
Lookup 860
Transformation Rule Profiles 861
Loading Data With Excel Templates or CSV Files 861
Loading Stage Data 862
Loading Form Data 865

Loading Journal Data 872
Loading Cell Detail 881
Cell Detail CSV Template 884
Exporting Data Attachments 886
Loading Excel Templates to Custom Tables 887
Importing Text-based Data 891
Long Running Requests 892
Journals Overview 893
Journals Toolbar 894
Input Journals 894
Process 899
Confirm 900
Certify 900
Multi-Period Processing 902
Analysis Pane 902
Cube Views and Dashboards for Analysis 902

Intercompany Matching 903
Create a New Journal 904
Create Journal 904
Create Journal Using Selected Template 906
Create Journal Using File 907
Open a Journal 908
Loading Journal Data 910
Journal Excel Template 910
Header Abbreviations 913
Journal CSV Template 914
Loading Journal Details via Workflow 916
Extracting/Loading Journal Details via BRApi 917
Delete a Journal 919
Process a Journal 921
Validate a Journal 921
Reject a Journal 922

Post a Journal 922
Unpost a Journal 923
Journal Templates 923
Journal Template Properties 924
Journal Template Group Properties 927
Security 927
Presenting Data With Books, Cube Views and Dashboards 929
Presenting Data Using Report Books 929
Book Designer Toolbar 930
Book Properties 932
File Type 933
Excel Export Item 934
Report 934
Loop 935
Loop Variables 937
Change Parameters 938

If Statement 942
Else/Else If Statement 943
Create a Report Book 944
Book Preview Toolbar 946
Build Reports Through Cube Views 949
Build a Simple Cube View 950
Cube View Shortcuts 960
Member Filter Builder 960
Cube View Performance 968
Advanced Cube Views 976
Format a Cube View 979
Calculations 1042
Dashboards 1077
Dashboards 1078
Dashboard Design Mode 1085
Preview Dashboards 1089

Dashboard Maintenance Unit Properties 1091
Dashboard Group Properties 1091
Dashboard Profile Properties 1092
Dashboard Properties 1093
Literal Parameter Values 1096
Action (Primary Dashboard Only) 1096
Grid Layout Type 1097
Dashboard Components 1103
Components 1103
Position on Dashboard 1104
Data Adapters 1287
Cube View MD 1290
Dimension Leveling 1296
Method Query 1302
SQL 1309
BI Blend Adapter 1310

Parameters 1312
Files 1313
Strings 1313
Linked Dashboards 1314
User Culture 1322
Workspaces 1338
Benefits 1338
The Workspaces Page 1339
Workspaces Filter 1344
Workspaces Properties 1347
Workspaces Setup 1351
Workspace Assemblies 1352
Finance Business Rules 1369
XF Project 1376
Build Cube Views in Workspaces 1381
Configure the Service Factory 1384

Runtime Logic 1385

Reference the Service Factory in Properties 1391

Examples 1394

Dynamic Dashboards 1434

Embedded Dynamic Repeater Dashboard 1441

Dynamic Cube Views 1455

Build Data Management in Workspaces 1461

Load and Extract for Data Management Objects 1476

Substitution Variable Items 1483

Substitution Variable Syntax 1484

Add a Workspace Substitution Variable 1484

Find and Assign Values Using Object Lookup 1485

Cube Views Example 1489

Extensible Document Settings 1490

Options 1495

XFCell 1496

Use Business Rules in Extensible Documents 1497

Application Tools 1498

Application Security Roles 1498

Application Properties 1504

General Properties 1504

Dimension Properties 1508

Standard Reports 1509

Header Labels 1510

Header Bar 1511

Footer 1511

Business Rules 1512

Business Rule Search 1518

Business Rule Properties 1521

Business Rule Types 1524

Parser Rule 1527

Connector Rule 1529

Conditional Rule 1533
Derivative Rule 1535
Cube View Extender 1537
Dashboard DataSet 1538
Dashboard Extender 1539
Dashboard XFBRString 1540
Extensibility Rule 1540
Extender 1544
Journals Event Handler 1546
Save Data Event Handler 1547
Forms Event Handler 1547
Data Quality Event Handler 1548
Data Management Event Handler 1549
Workflow Event Handler 1549
Client Updater 1550
Data Management 1552

Search 1552

Data Management 1552

Sequences 1553

Steps 1557

Execute Scenario Hybrid Source Data Copy 1559

Clear Data 1559

Copy Data 1561

Custom Calculate 1563

Execute Business Rule 1565

Export Data 1566

Export File 1569

File Source Type 1570

Export Report 1571

Reset Scenario 1572

Run a Sequence or Step 1573

Spreadsheet 1574

Microsoft Excel 2016 Chart Types 1577
Retrieve Functions 1583
File, Open and File, Save As Functionality 1583
Reduce Cell Text Size 1585
OneStream Ribbon 1585
Task Scheduler 1587
Grid View 1587
Task Scheduler Details 1588
Load 1602
Extract 1604
Create a New Task 1606
User Authentication and Single Sign-On 1648
How Users are Configured for Authentication 1649
About Managing Users and Groups 1650
Ways to add Users and Groups to an Application 1650
Creating and Managing Users 1651

Requirements 1651

Tips and Best Practices 1652

Creating Users 1652

Managing Users 1655

Loading and Extracting Users 1659

Creating and Managing Groups 1660

Requirements 1661

About Groups and Inherited Security 1662

About Exclusion Groups 1663

Tips and Best Practices 1663

Creating Groups 1664

Creating Exclusion Groups 1664

Managing Groups 1666

Managing Group Membership 1667

Loading and Extracting Groups 1667

System Security Roles 1670
Application Server Configurations 1671
System Roles 1671
System Configurations 1672
General System Configurations 1672
Environment System Configurations 1676
Recycling System Configurations 1680
Database System Configurations 1681
Audits 1684
Manage System Security 1686
Access System Security Roles 1687
Manage System Security Users 1688
Manage System Security Groups 1689
Manage System Security Roles 1689
Load and Extract 1689
BRApi 1690

Combined Roles 1691
File Share Security Roles 1691
System User Interface Roles 1692
System Security Users and Groups 1694
Managing Users and Groups Using BRApi Functions 1695
Examples 1696
Create a User 1697
Create a Group 1697
International Settings 1697
Specifying Culture Settings on the Client Machine Operating System 1698
Specifying Server Side User Culture Settings 1699
Working With System Applications 1699
System Dashboards 1700
Logging 1702
Logon Activity 1702

Task Activity 1704
Enhancing Task Activity 1706
Task Activity in Excel 1706
Enhancing Task Activity: Cancel Cube Views 1710
Cube View Paging 1718
Quick View Paging 1722
Error Logs 1725
System Tools 1727
System Business Rules 1727
Sample System Business Rule 1728
Database 1729
Environment 1731
Monitoring Environments 1732
Specifying General Settings 1733
Specifying Monitoring Settings 1733
Specifying Metrics Settings 1734

Web Servers 1735
Configuration 1735
Audit 1737
Mobile Web Servers 1739
WebTo App Server Connections 1740
Working With Application Server Sets 1742
Accessing Server Sets Behaviors 1743
Server Sets Configuration 1745
Using Server Sets Audit 1746
Monitoring Server Sets Performance 1747
Application Server Behavior 1747
Application Server Configurations 1749
Application Server Hardware 1749
Application Server Audit 1750
Application Server Performance 1751
Database Servers (Connection Items) 1752

Behaviors 1752

Configuration 1753

Hardware 1754

Audit 1755

Performance 1756

Diagnostics 1757

OneStream Database Servers (Schema Items) 1759

Configuration 1759

Audit 1760

Diagnostic 1761

Azure Configurations 1761

Azure Subscription Settings 1762

Environment Monitoring 1762

Task Load Balancing 1767

Database Server Connection 1767

Azure Server Sets Settings 1769

File Explorer 1770
File Share 1771
Folders 1772
Uploads 1777
Supported File Sizes 1777
Whitelist File Extensions 1777
Overview 1778
Whitelist Files Extensions Prerequisites 1781
Define the List of File Extensions 1781
Add Documents using File Explorer 1784
Troubleshooting 1785
Conclusion 1786
Application Database/System Database 1786
Documents 1786
Load/Extract System Artifacts 1787
Time Dimensions 1789

Using OnePlace Workflow 1791
Workflow Tasks 1794
Import 1795
Right-Click Options 1797
Validate 1798
Load 1799
Pre-Process 1800
Input Forms 1800
Journals Toolbar 1804
Input Journals 1804
Process 1809
Confirm 1810
Certify 1810
Multi-Period Processing 1812
Analysis Pane 1812
Cube Views and Dashboards for Analysis 1812

Intercompany Matching 1813

Using OnePlace Cube Views 1815

Toolbar 1816

Shortcuts, Scaling, and HotKeys 1819

Shortcuts 1819

Scaling 1820

HotKeys 1821

Right-Click Options 1821

Spreading Type 1822

Spreading Properties 1825

Allocation 1827

Data Attachments for Selected Cell/Data Unit 1837

Spell Checking in Data Attachments 1839

Cell Detail 1842

Cell POV Information 1845

Cell Status 1848

Data Unit Statistics 1849
Drill Down 1850
Show Cube View as a Report 1854
Using OnePlace Dashboards 1856
Dashboard Toolbar 1857
Printing Options 1857
Storing Documents 1859
Excel Add-In 1861
Connect the Excel Add-In 1862
Add the OneStream Ribbon 1862
Log Into the Excel Add-In 1864
Update the Excel Add-In 1866
Compatibility 1866
Add-In Components 1866
Task Pane 1866
OneStream Ribbon 1868

Error Logs 1884

Work with Data 1885

Cube Views 1885

Quick Views 1905

Table Views 1921

Parameters 1925

In-Sheet Actions 1928

Right-Click Options 1931

Retrieve Functions 1936

XF Retrieves 1944

Visual Basic for Applications (VBA) Procedures 1946

Formatting in Excel 1950

Preserve Formatting in Excel 1951

Styles 1953

Conditional Formatting 1960

Selection Styles for Cube Views 1965

Introduction

The purpose of this guide is to provide best practices, tips, in-depth explanations of key capabilities, and configurations for implementers, administrators, and end users. Use this guide to enhance your understanding of applications, from specific functions to comprehensive reporting.

About the Financial Model

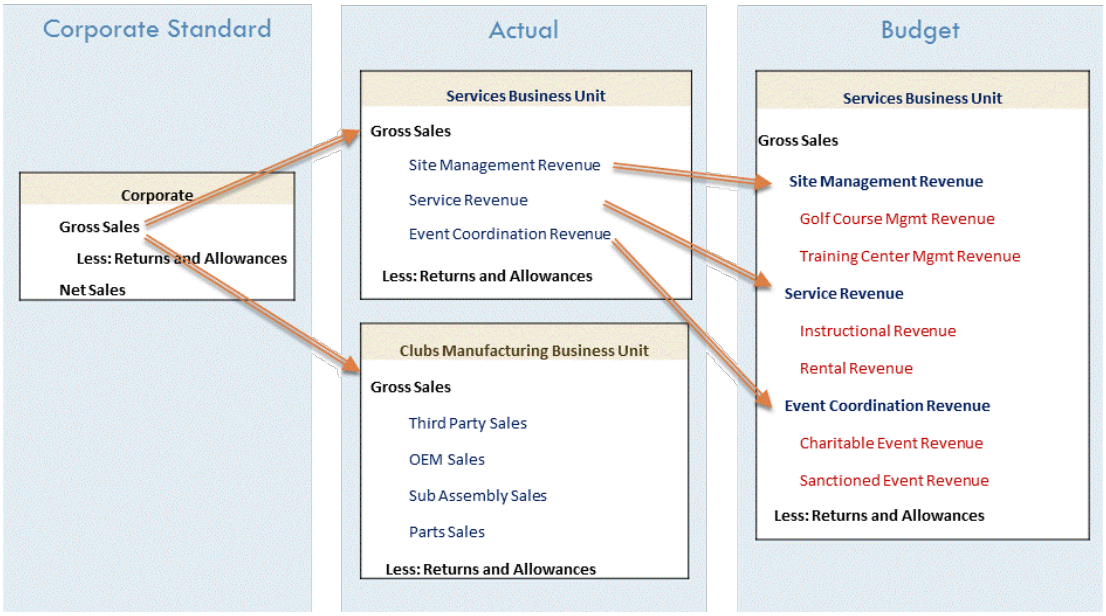
The financial model includes Extensible Dimensionality, which allows you to extend the Dimensions to suit your own financial purposes. Cubes and design model considerations, along with the use of an aggregate storage model called BI Blend, which supports large-volume reporting for data that is not appropriate 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.

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 vs. a manufacturing business unit as well as across the Actual and Budget scenarios. Notice how each business unit can look at Gross Sales differently in the Actual scenario. Also, the Services business unit can 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. In the example above, there are four different Account dimensions 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 rollup 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 above example, there are three cubes: Corporate, Clubs, and Services. When looking at the Corporate cube, data from the three cubes is all 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.

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

Extensible Cube

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.

Extensible Workflow

There can be different Workflow Profile hierarchies per Scenario cype which is defined at the cube level. For 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 signoff levels.

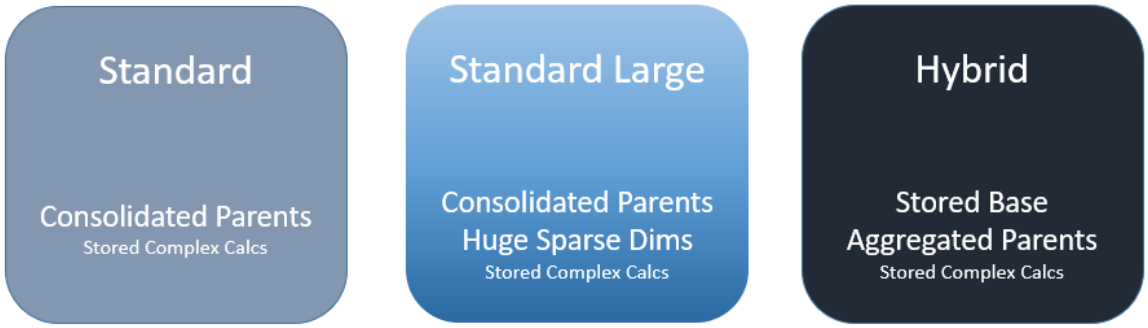
Cube Design Choices within an Application

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 (“sparsely populated”)?

- 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 ~2 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” 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 of the house” queries.
- Quick Views may be impacted as they do not include Sparse Row Suppression settings.

Hybrid Cube Design

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 is 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 of the house” 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.

Dynamic Cube Services

Dynamic Cube Services can assist with managing and analyzing complex volumes of financial data, leading to faster and more accurate analysis, reporting, and decision-making.

Dynamic Cube Services provides these features for designers with advanced knowledge of the OneStream architecture:

- Consolidate financial data from various sources
- Create and compare different budgeting and forecasting scenarios
- Track key performance indicators in real-time, enabling better management and decision-making
- Analyze risk data across multiple dimensions, helping mitigate financial risks
- Help ensure that financial data is accurate and up-to-date for internal compliance and auditing

Dynamic Cubes Overview

Dynamic cubes structure data into a multidimensional cube, ready for simplified reporting and analysis using tools such as cube views, quick views and dashboards. Dynamic cubes offer a solution for application designers to surface non-traditional data, such as transactional data, without extra loads ; or to re-aggregate existing Cube or Solution data in different ways.

Dynamic cubes enable real-time connectivity to a source, such as an external database or source cube, without needing to store the data in the target cube. They are based on Workspace Assembly files that can define the dimension metadata and hierarchies, and instruct the cube on how to retrieve and/or commit data records.

Data is integrated using data bindings configured on the cube, rather than Workflow-based imports. Dynamic cubes can be supplied content from other cubes, tables, and external databases. Data can be copied from a source and stored in data record tables, or it can be shared. A shared configuration can function as a reference to the source system, which means the data might not be physically stored in OneStream.

Dynamic cubes allow data collection through the Forms Origin member. The data submission flow can be customized using Workspace Assembly files to target tables, or to direct submitted data to an external system.

The elimination of stored metadata can be accomplished using Dynamic Dimensions. Such Dimensions are created and referenced by the dynamic cube in the same manner as a standard cube. However, members, properties, and relationships, are defined using Workspace Assembly rules, which manage the retrieval of data records as well as the cadence and method to refresh data and dimension metadata.

> **NOTE:** Dynamic Cubes do not always require Dynamic Dimensions: data can be provided dynamically for cubes built with traditional stored dimensions too.

Dynamic Dimensions Overview

Dynamic dimensions are dimensions in which dimension members are not stored in an internal member table. Instead, the dynamic dimension is typically used to surface members from external sources for use in cubes. Dynamic dimensions are a key design tool that enable external database integration with dynamic cubes.

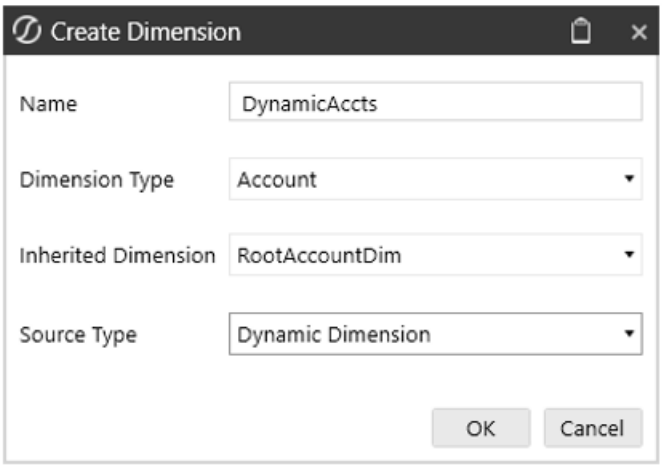
Dynamic dimension members cannot be created, updated, or deleted through the standard Dimension Library. Workspace Assembly rules define and retrieve members from external sources. They are used to apply properties, and to define relationships used to create a dimensions. Metadata management of the dynamic dimension is performed by the definition within workspace assembly rules.

Dynamic dimension members are read in real-time and cached each time the workspace assembly executes, according to the caching parameters defined in their rules. Dynamic dimension members only exist in code and are not saved to tables. Once referenced by stored data, they are partially kept in an internal table, called DynamicMember, to facilitate audit and application management.

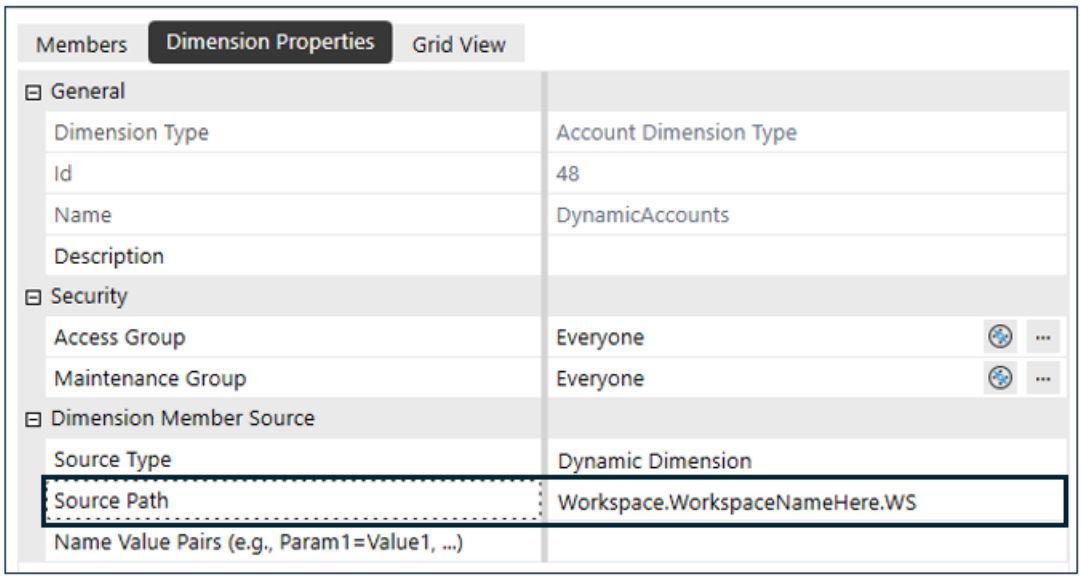
Configure a Dynamic Dimension

Dynamic Dimensions are created for “right-hand” Account-type dimensions only: Account, Flow and User-Defined 1 through 8.

The dimension is created and assigned a **Source Type** of **Dynamic Dimension** in the **Create Dimension dialog box** for the **Source Type** property.



You can create and define members using a Workspace Service Factory, which is referenced through a workspace in the **Source Path** property. See Workspaces > Configure the Service Factory in the Design and Reference Guide for more information.



Capabilities

Dynamic Dimensions support the following functions:

- Enrichment of metadata using all OneStream properties.
- Extensibility is fully supported and integrated.
- Members are supported in Sparse Row Suppression.
- Can be extracted for use as a standard dimension.

Limitations

Consider these limitations when using Dynamic Dimensions:

- Dynamic dimensions are not available on data unit dimensions..
- When writing to standard cubes or consolidating dynamic cubes, dynamic dimensions store records in data record tables. Redefining these dimension members will not clear stored data records; however, if a member is recreated with the same member ID, the records will continue to work properly. Specify external member IDs to help maintain references.
- Dimension members created by mirroring an existing dimension are created with new member IDs.
- Dynamic members that are removed or no longer exist, do not populate as orphan members. Dynamic members that are defined but not linked to any parent, will appear as orphans.
- DynamicCalc formulas on members are not supported when defined within a dynamic dimension.

- Standard dimensions should never be extended from dynamic dimensions. Extending a dynamic dimension with another dynamic dimension is possible, but not recommended.

Recommendations

Consider these recommendations when using Dynamic Dimensions:

- Extensibility designs are only supported as dynamic dimensions extend from standard stored dimensions.
- Use naming conventions to clearly identify that a dimension is dynamic.

Workspace Assembly Dynamic Dimensions

Dynamic Dimension Services execute assembly files for dynamic dimensions. The assembly file uses public classes to construct the DynamicDimensionInfo object, which creates the Dynamic Dimension. The DynamicDimensionInfo object uses the following syntax:

Syntax: DynamicDimensionInfo(ContentTimeStamp, NumSecondsBeforeReadContentTimeStamp, DynamicMemberCollection, DynamicRelationshipCollection)

For additional details on required objects and syntax, refer to Dynamic Cube Services API Reference.

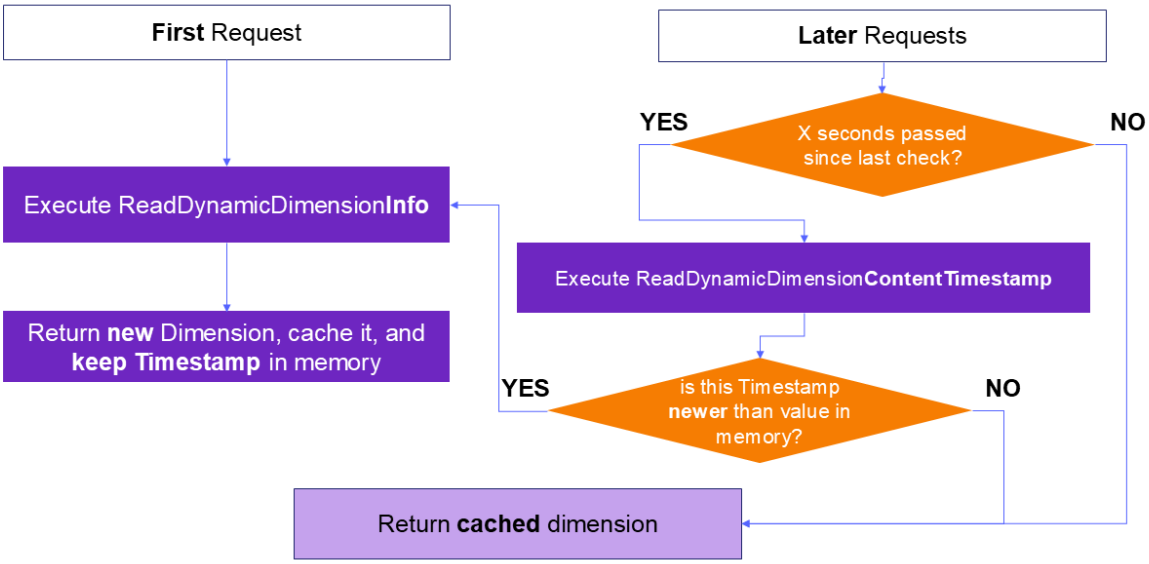
When designing a dynamic dimension, consider the following:

- **Frequency and trigger** for the dimension to update during workspace or assembly file changes, or time-based updates (daily, hourly, or by the minute). For details, see Frequency and Trigger.

- **Member definitions**, including properties, relationships, and aggregation factors. For details, see Member Definitions.
- **Metadata Enrichment** of dynamic members for properties, constraints, and member formulas. For details, see Metadata Enrichment.

Frequency and Trigger

Each application server processes and caches metadata using a simple algorithm. When a dimension object is created, it gets a timestamp and a wait period before checking for updates. During this period, the cached object is returned from memory for faster access. After the timeout, the system calls ReadDynamicDimensionContentTimestamp to decide whether to recreate the object or continue using the cached version.



> **Example:** The timeout ("X seconds" in the diagram above) is defined with the NumSecondsBeforeReadContentTimestamp argument of the dimension constructor:

Syntax: DynamicDimensionInfo(__, NumSecondsBeforeReadContentTimeStamp,__ ,__)

Selection of such value should be based on the expected frequency of changes in the dimension source. If the cadence is unknown, or the members are manually created in the assembly file, the update can be triggered through changes to the assembly file by using the time variable **SharedConstants.DateTimeMinValue** in ReadDynamicDimensionContentTimestamp.

The logic in ReadDynamicDimensionContentTimestamp will naturally determine how often the dimension is recreated. If you use a higher refresh frequency, such as by minute, application performance may be impacted, especially with high volumes.

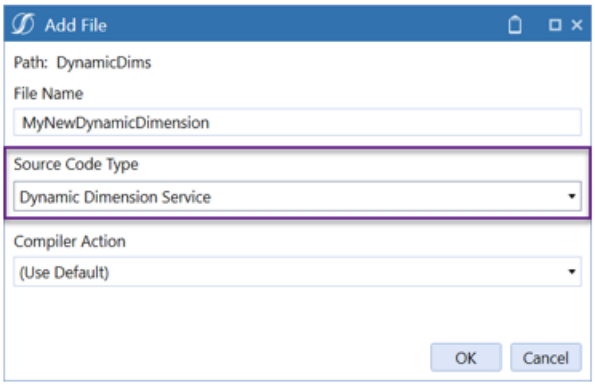
Refer to the following table for time-based refresh trigger rates:

Refresh Type	Example ReadDynamicDimensionContentTimestamp implementation
Daily Refresh	Dim now = Datetime.now Return new DateTime(now.Year, now.Month, now.Day, 0, 0, 0, 0, DateTimeKind.utc)
Hourly Refresh	Dim now = Datetime.now Return new DateTime(now.Year, now.Month, now.Day, now.Hour, 0, 0, 0, DateTimeKind.utc)

Refresh Type	Example ReadDynamicDimensionContentTimestamp implementation
Minute Refresh	Dim now = Datetime.now Return new DateTime(now.Year, now.Month, now.Day, now.Hour, now.Minute, 0, 0, DateTimeKind.utc)

Member Definitions

After a dimension is created using the **Source Code Type** of Dynamic Dimension Service, all metadata management is performed through workspace assembly files. Members can be created and defined in the assembly file. The assembly file can also source members and define properties from external systems, such as a database table. The members may be defined as either base or parent.



A new dynamic dimension service assembly file is created and populated with sample code that exposes the public classes and functions.

The assembly file creates the member and its relationship to parents in the dimension hierarchy. These objects are then passed to the DynamicDimensionInfo constructor. This object includes a data-time property that supports controlling and triggering the dimension refresh.

Syntax: DynamicDimensionInfo(ContentTimestamp, numSecodsBeforeCheckingTimestamp , DynamicMemberCollection, DynamicRelationshipCollection)

The following objects will have to be created to support the constructor:

- **DynamicMemberCollection**: The list of members in the dimension. Note: when specifying member names, if they contain unsupported special characters, they will be automatically stripped.
- **DynamicRelationshipCollection**: The list of parent-child relationship between members, defining the hierarchy structure. Note that it should also contain the relationship between any top member and Root, or between top members and members of an inherited (extended) dimension.

For additional details, refer to Dynamic Cube Services API Reference.

Note that the system provides a shortcut method to build the entire dimension (and attach it to the Root member) with a single SQL query. See the API section for further details.

If the resulting dimension must be nested under an existing member instead of Root, the relationship can be modified like this:

```
Dim dynamicDim = api.ReadDynamicDimensionInfo(...)
dynamicDim.Relationships.GetAllRelationships().Find(Function(r) r.ParentId.Equals
(DimConstants.Root)).ParentId = newParentId
```

```
{
    var dynamicDim = api.ReadDynamicDimensionInfo(...);
    dynamicDim.Relationships.GetAllRelationships().Find(r => r.ParentId.Equals
(DimConstants.Root)).ParentId = newParentId;
}
```

Enabling the Service

The Dynamic Dimension service must be returned by a Service Factory.

Example:

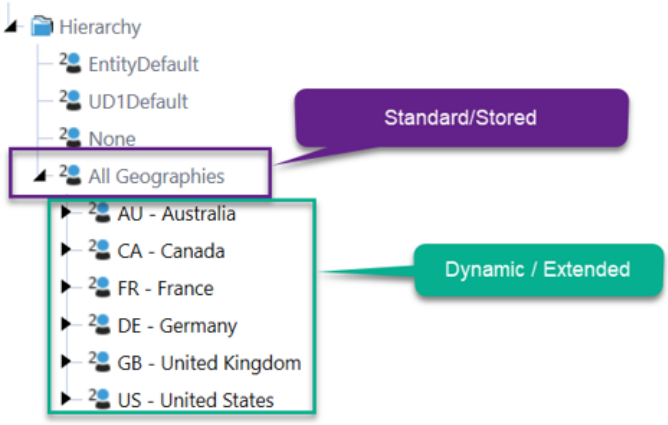
```
Select Case wsasType
Case Is = WsAssemblyServiceType.DynamicDimension
     ' we can switch on the dimension name
     If itemName.XFEqualsIgnoreCase("MyUD1") Then
          Return New MyNewDynamicDimension()
```

```
{
    switch (wsasType)
    {
        case object _ when wsasType == WsAssemblyServiceType.DynamicDimension:
            {
                // we can switch on the dimension name
                if (itemName.XFEqualsIgnoreCase("MyUD1"))
                    return new MyNewDynamicDimension();
                break;
            }
    }
}
```

See Workspaces > Configure the Service Factory in the Design and Reference Guide for more information.

Extensibility

Dynamic dimension assembly files can reference a stored dimension member to support extensibility. Standard dimensions should never be extended from dynamic dimensions. Extending a dynamic dimension with another dynamic dimension is possible, but not recommended.



Metadata Enrichment

When objects are initiated as dynamic members or a relationship collection, the objects can be enriched to configure all stored and vary-by properties. Members created within the assembly file, or originating in source systems, can be configured with all required and optional properties, including vary-by-ScenarioType and Time properties. Example:

```
Dim dynmem As DynamicMember = myDynamicDimensionInfo.Members.GetMember("My Member Name")
dynmem.VaryingMemberProperties.GetUDProperties().Text1.SetStoredValue
(ScenarioType.Unknown.id, DimConstants.Unknown, "Something")
```

```
{
    DynamicMember dynmem = myDynamicDimensionInfo.Members.GetMember("My Member Name");
```

```
    dynmem.VaryingMemberProperties.GetUDProperties().Text1.SetStoredValue
(ScenarioType.Unknown.id, DimConstants.Unknown, "Something");
}
```

Example

The following example implements a simple dynamic dimension from an SQL table.

```
Namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
    Public Class MyDynamicDimensionService
        Implements IWsasDynamicDimensionV800
        Function ReadDynamicDimensionContentTimestamp(ByVal si As SessionInfo, ByVal api As
IWsasDynamicDimensionApiV800, ByVal args As DynamicDimensionArgs) As DateTime  Implements
IWsasDynamicDimensionV800.ReadDynamicDimensionContentTimestamp
            Try
                ' Return SharedConstants.DateTimeMinValue if you are unable to determine
                ' when the underlying dimension members or relationships have been changed.
                ' In that case, the cache for this dimension will only be refreshed when
                ' a user edits this dimension's properties, this workspace's properties,
                ' or this workspace assembly's code.
                Return SharedConstants.DateTimeMinValue

            Catch ex As Exception
                Throw New XFException(si, ex)
            End Try
        End Function
        Public Function ReadDynamicDimensionInfo(ByVal si As SessionInfo, ByVal api As
IWsasDynamicDimensionApiV800, ByVal args As DynamicDimensionArgs) As DynamicDimensionInfo
Implements IWsasDynamicDimensionV800.ReadDynamicDimensionInfo
            Try
                ' retrieve parameters from Dimension configuration
                Dim nameValuePairs = New NameValueFormatBuilder
(args.Dim.DimMemberSourceNVPairs)
                Dim secsToWait As Integer = nameValuePairs.NameValuePairs.XFGetValue
("SecsToWait", 5)

                ' Define timestamp
                Dim contentTimestamp As DateTime = DateTime.Now

                ' build the dimension from a query on the Application database
                Dim dynamicDim = api.ReadDynamicDimensionInfo(si, args, contentTimestamp, _
                        SharedConstants.Unknown, api.DbConnApp, _
                        "SELECT Child, Description, Parent FROM MyCustomDimensionTable", _
                        Nothing, Nothing, "Child", "Description", _
                        String.Empty, String.Empty, "Parent", String.Empty, String.Empty,
String.Empty, False)

                ' Tweak the aggregation weight for one member
```

```
                Dim someMemberID As Integer = dynamicDim.Members.GetMember
("MyMember").Member.MemberId
                dynamicDim.Relationships.GetAllRelationships().Find(Function(r)
r.ChildId.Equals(someMemberID)).UDAggWeight = 0.0

                ' Set a text property on a member
                Dim dynmem As Integer = dynamicDim.Members.GetMember("AnotherMember")
                dynmem.VaryingMemberProperties.GetUDProperties().Text1.SetStoredValue( _
                    ScenarioType.Unknown.id, DimConstants.Unknown, "Something")

                ' return the hierarchy
                Return dynamicDim
            Catch ex As Exception
                Throw New XFException(si, ex)
            End Try
        End Function
    End Class
End Namespace
```

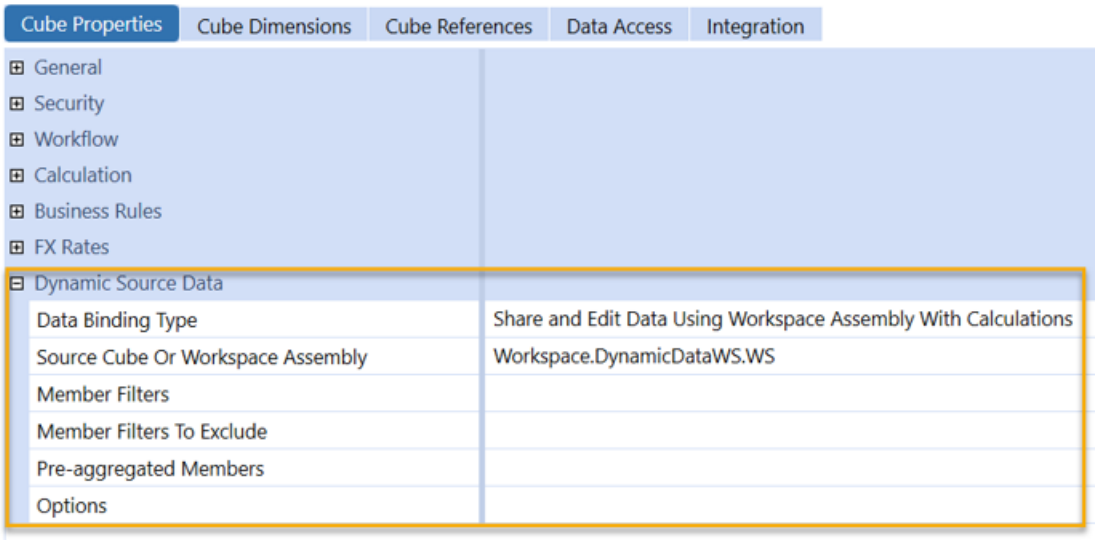
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Security;
using System.Text;
using System.Threading.Tasks;
using Microsoft.VisualBasic;

namespace TestProject.Workspace.__WsNamespacePrefix.__WsAssemblyName
{
    public class MyDynamicDimensionService : IWsasDynamicDimensionV800
    {
        public DateTime ReadDynamicDimensionContentTimestamp(SessionInfo si,
IWsasDynamicDimensionApiV800 api, DynamicDimensionArgs args)
        {
            try
            {
                // Return SharedConstants.DateTimeMinValue if you are unable to determine
                // when the underlying dimension members or relationships have been changed.
                // In that case, the cache for this dimension will only be refreshed when
                // a user edits this dimension's properties, this workspace's properties,
                // or this workspace assembly's code.
                return SharedConstants.DateTimeMinValue;
            }
            catch (Exception ex)
```

```
            {
                throw new XFException(si, ex);
            }
        }
        public DynamicDimensionInfo ReadDynamicDimensionInfo(SessionInfo si,
IWsasDynamicDimensionApiV800 api, DynamicDimensionArgs args)
        {
            try
            {
                // retrieve parameters from Dimension configuration
                var nameValuePairs = new NameValueFormatBuilder
(args.Dim.DimMemberSourceNVPairs);
                int secsToWait = nameValuePairs.NameValuePairs.XFGetValue("SecsToWait", 5);

                // Define timestamp
                DateTime contentTimestamp = DateTime.Now;

                // build the dimension from a query on the Application database
                var dynamicDim = api.ReadDynamicDimensionInfo(si, args, contentTimestamp,
SharedConstants.Unknown, api.DbConnApp, "SELECT Child, Description, Parent FROM
MyCustomDimensionTable", null/* TODO Change to default(_) if this is not a reference type
*/, null/* TODO Change to default(_) if this is not a reference type */, "Child",
"Description", string.Empty, string.Empty, "Parent", string.Empty, string.Empty,
string.Empty, false);

                // Tweak the aggregation weight for one member
                int someMemberID = dynamicDim.Members.GetMember("MyMember").Member.MemberId;
                dynamicDim.Relationships.GetAllRelationships().Find(r => r.ChildId.Equals
(someMemberID)).UDAggWeight = 0.0;

                // Set a text property on a member
                int dynmem = dynamicDim.Members.GetMember("AnotherMember");
                dynmem.VaryingMemberProperties.GetUDProperties().Text1.SetStoredValue
(ScenarioType.Unknown.id, DimConstants.Unknown, "Something");

                // return the hierarchy
                return dynamicDim;
            }
            catch (Exception ex)
            {
                throw new XFException(si, ex);
            }
        }
    }
}
```

Configure a Dynamic Cube

You can turn a cube into a dynamic cube by setting a **Data Binding Type** property for the cube. In these configurations, data integration is based on the workspace assembly file definition and is limited to the Forms origin to meet any manual data entry requirements.

Dynamic cubes can be configured with out-of-the-box properties that support the design of specialty cubes in a similar manner as hybrid scenarios. Workspace assembly configurations are used for more advanced solutions, such as surfacing external sources in a cube.



Data storage varies by use and design of the dynamic cube, as well as by the Data Binding Type property. Consider the following:

- Copy input data options store data records in data record tables at base level intersections.
- Share options for dynamic cubes manage the data as a pointer to the source system.
- Dynamic cube data integration is restricted to the Forms origin member for data entry.

- Dynamic cube data entry requires base-level intersections. O#BeforeAdj is not supported as a data entry point.
- Dynamic cubes support write-back data to external sources and can write back text-based data through V#Annotations to external sources.

Shared data acts as a pointer to parent and base level data unit members. However, account, flow, and user-defined parent members are calculated on the fly. Base-level data unit members are required when data is integrated with a copy operation.

Data bindings have the following variations for edit and calculations:

- **Edit:** Lets the dynamic cube use a workspace assembly file to write-back to the defined source, such as an external table. Note that writeback to stored cubes is not supported.
- **With Calculations:** Lets dynamic cubes use workspace assembly files to execute member formulas that are stored in tables. Data is retrieved from the original source for the data unit and combined with any manual entry. Member formula calculations are stored and retrieved from regular tables, which are then merged into the results viewed in the dynamic cube.
- **CalcStatus:** Is supported and defined using a workspace assembly file.

Capabilities

Dynamic Cubes support the following functions:

- Can be used in consolidation, financial intelligence, and data unit calculation sequence adherence.
- Data unit dimensions can be targeted for shared data.
- Extensibility is supported.

Limitations

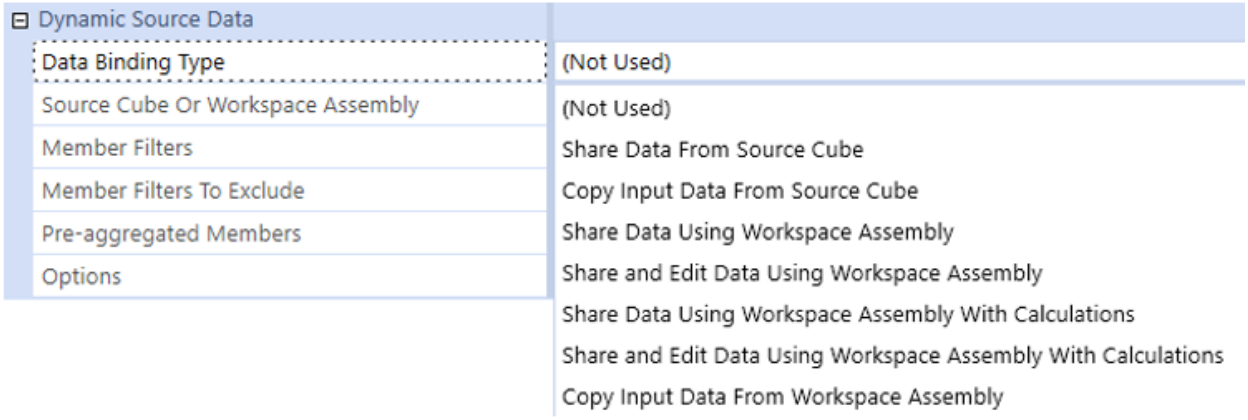
Consider these limitations when using Dynamic Cubes:

- Data unit size and capacity recommendations are the same as a standard cube. Follow the best practices for Cube design.
- Text-based data integration is restricted to V#Annotations.

Dynamic Cube Data Binding Types

In **Cube Properties** > **Dynamic Source Data**, there are seven Data Binding Type options for Dynamic Cube Services.

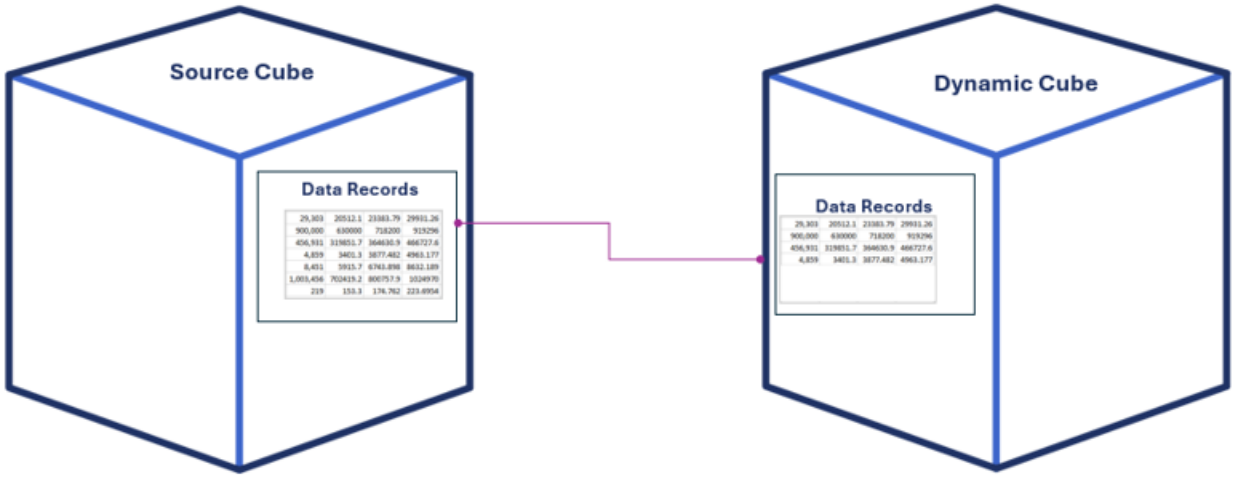
- Share Data From Source Cube
- Copy Input Data From Source Cube
- Share Data Using Workspace Assembly
- Share and Edit Data Using Workspace Assembly
- Share Data Using Workspace Assembly With Calculations
- Share and Edit Data Using Workspace Assembly With Calculations
- Copy Input Data From Workspace Assembly



> **NOTE:** In this topic, the source data can refer to information that comes from OneStream (stored cubes or Stage) or from external systems.

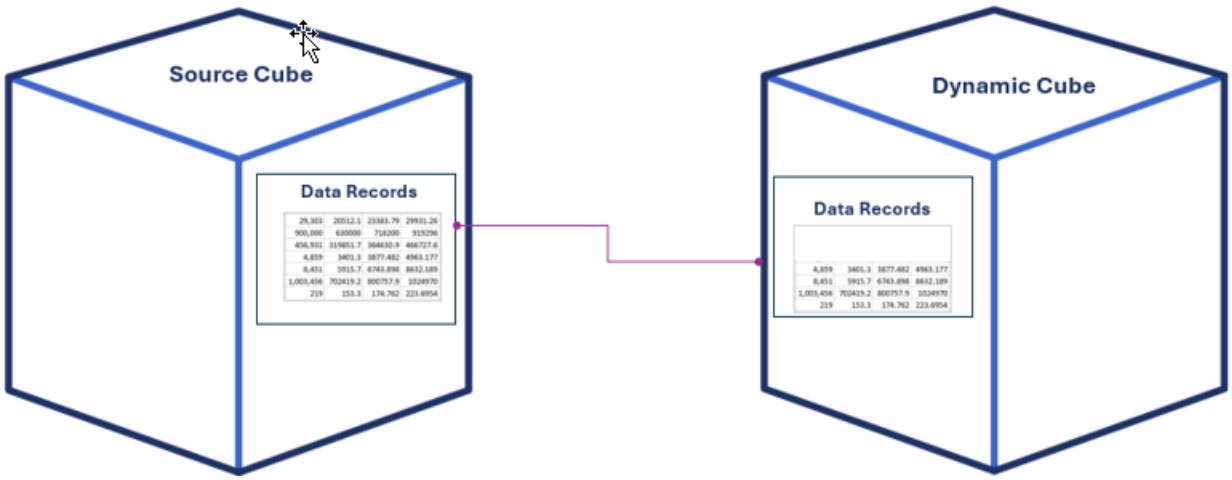
Share Data From Source Cube

Data is read-only and referenced from the source cube. This setting lets you refer to data from other cubes without duplicating the data, and also lets you filter which members to share. Benefits include a smaller data unit for filtering which members to share, as well as the ability to filter a specific member for data analysis when discontinuing an operation.



Copy Input Data From Source Cube

Base-level data is copied from the source cube as data records in dynamic cubes. Required for alternate rate-type and rule-type translations. You can copy base-level data from another cube and filter on members you want to copy. For example, you might only want to copy data from a P&L account and then retranslate in the source cube.

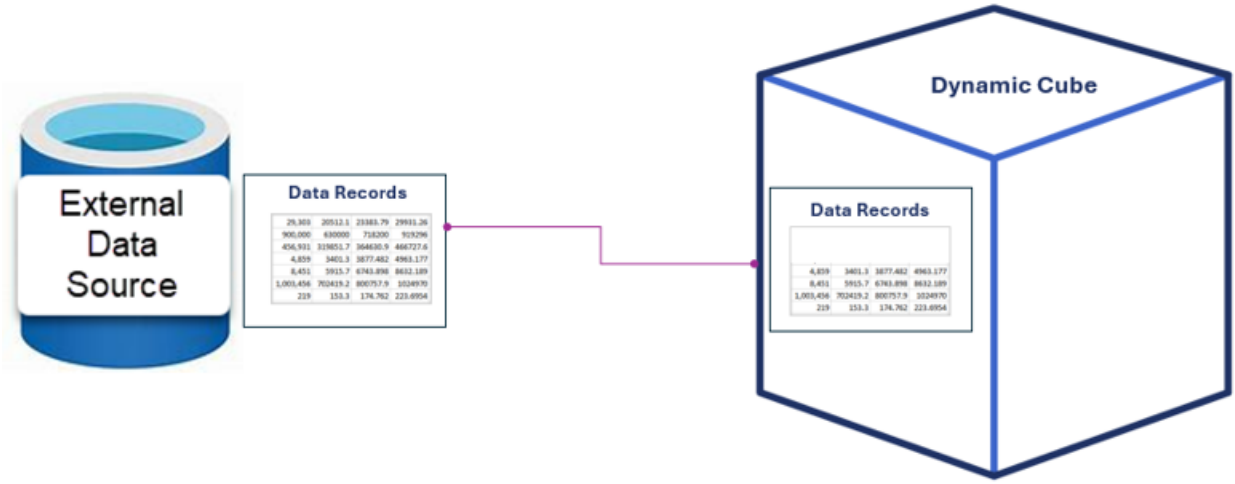


Share Data Using Workspace Assembly

This data binding lets you share data from any source through workspace assemblies. You can view data in cube views and filter which members you want to share.

For example, you can do the following:

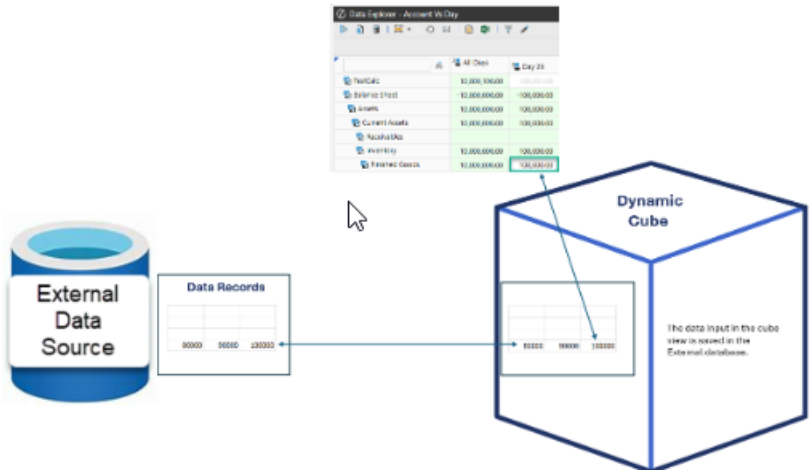
- Only share data for specific members such as income statement accounts.
- Rearrange or integrate data already collected in another cube using volatile dynamic dimensions and custom tables



Share and Edit Data Using Workspace Assembly

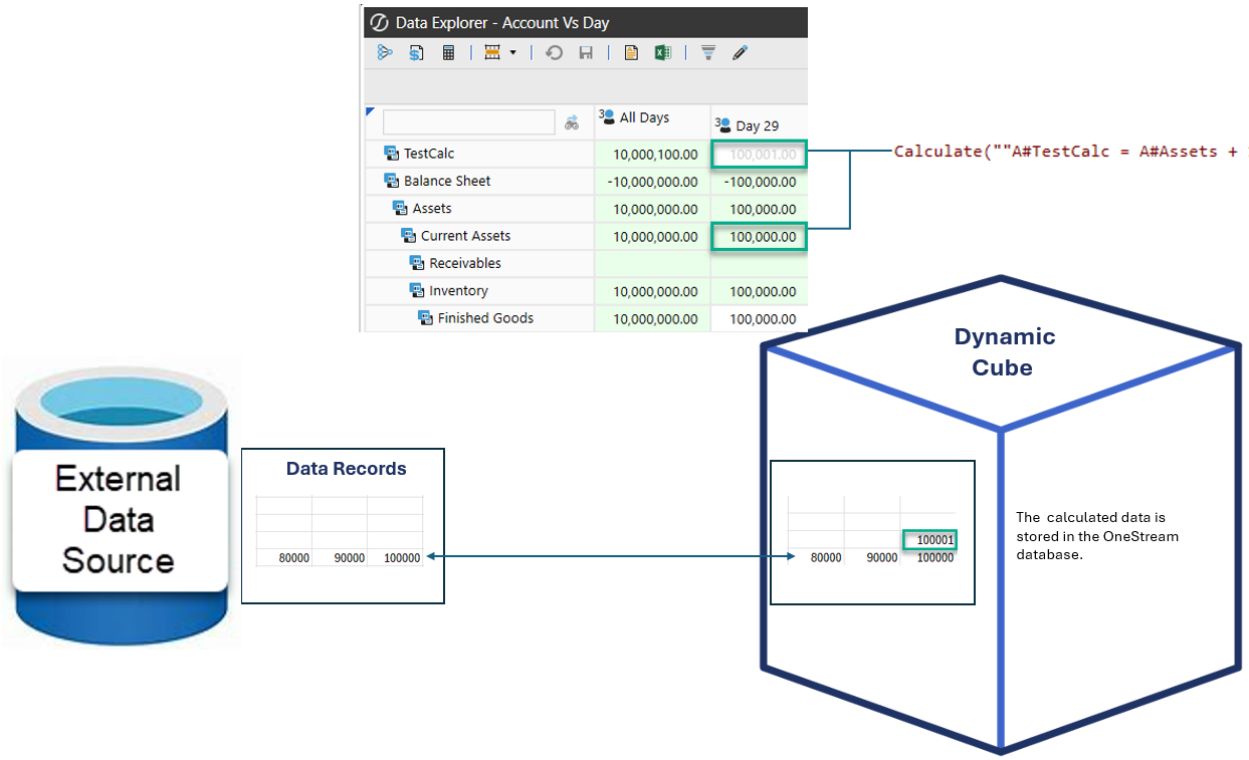
Share data from a non-cube source through workspace assemblies. For example, you can do the following:

- Use forms, such as cube views, to edit the data.
- Input data into a form to write back to the source, as well as reference data from the original data source.
- Filter members for sharing. For example, share data for specific members from an external data source, or integrate data already collected in another cube using data kept in custom tables.



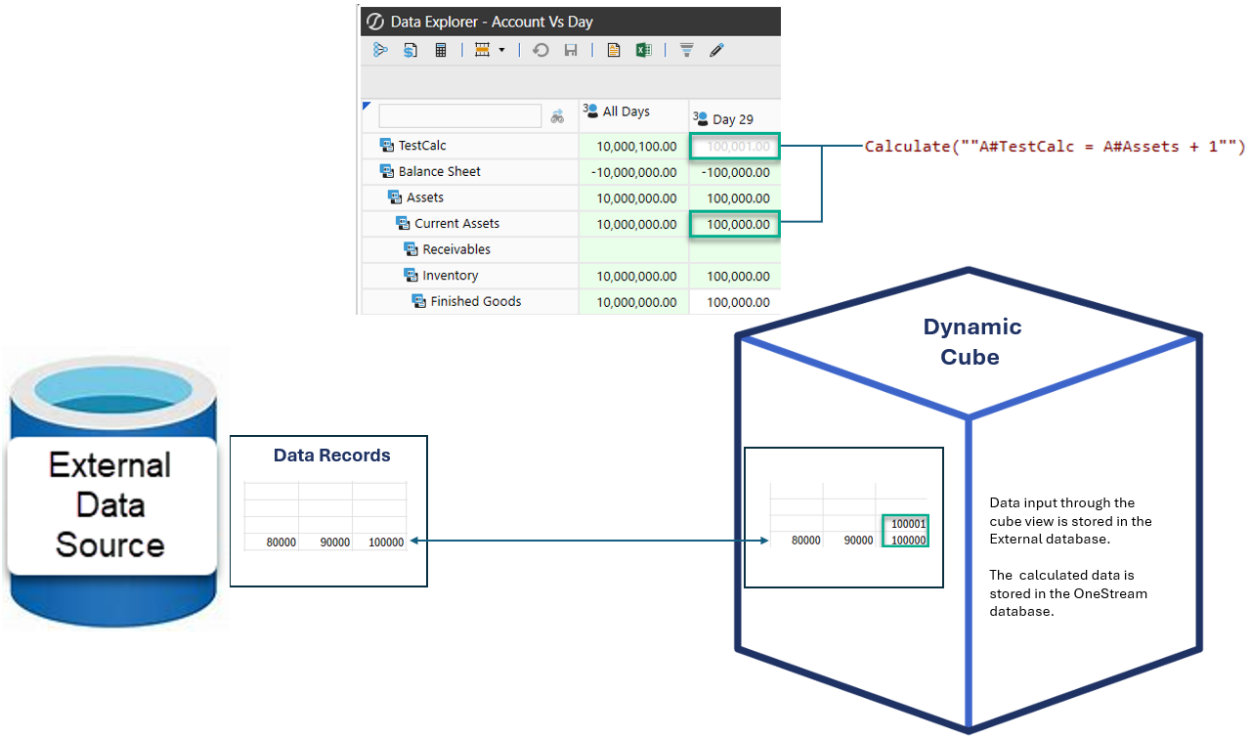
Share Data Using Workspace Assembly with Calculations

This model lets you share reference data from any data source and calculate data. You can filter members to share. For example, you might want to share data for specific members. Calculations defined in dynamic dimensions can be executed through workspace assemblies.



Share and Edit Data Using Workspace Assembly with Calculations

This model shares reference data from any non-cube source. You can use forms like cube views to edit the source data. You can filter members to share. For example, you might want to share data only for specific members. You can also perform calculations through workspace assemblies.

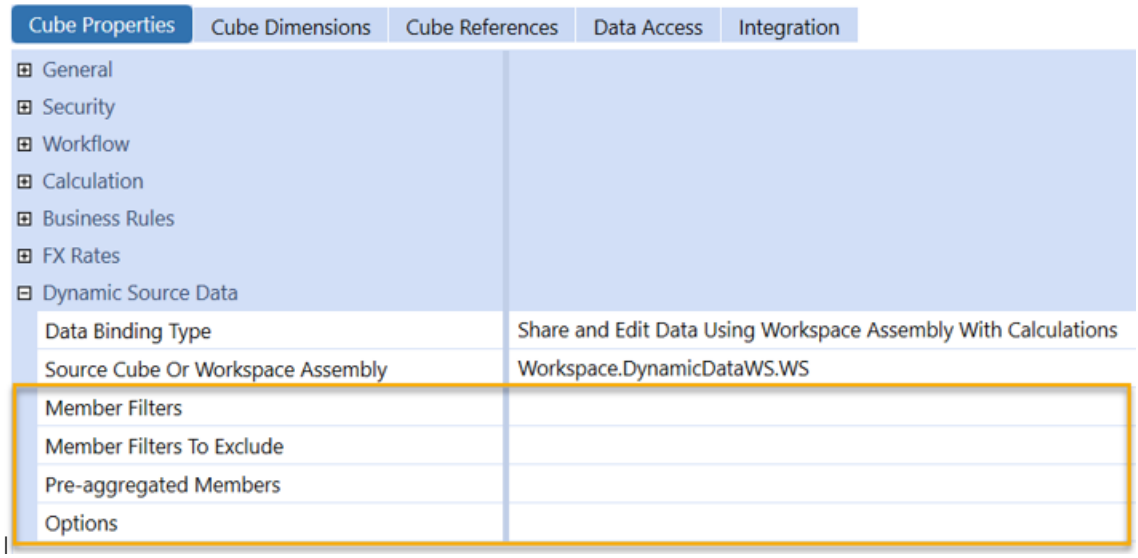


Copy Input Data From Workspace Assembly

With this model you can copy data from any source to data record tables.

Dynamic Cube Query Performance

A dynamic cube improves reporting query performance by filtering the results from source cubes into smaller data units.



In Cube Properties > Dynamic Source Data section, set these properties:

- **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. To include all source data in query results, leave the field blank.
- **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. To exclude no data, leave the field blanka.

> **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. 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 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. 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. To include multiple options, enter the name-value pairs in a comma separated list.

Dynamic Data Management

Workspace assembly files manage the retrieval and optional write-back of data between the external source and the data unit defined by the dynamic cube dimensions.

Workspace Assembly Dynamic Data

The Service Factory file references the DynamicData component. It initializes a class for a dynamic cube retrieval of data. Refer to the following example.

VB Code Example

```
case WsAssemblyServiceType.DynamicData:
    if (itemName.XFEqualIsIgnoreCase( "Dynamic Cube's Name"))
```

```
    {
    return new WsasDynamicData();
    }
```

C# Code Example

```
switch (WsAssemblyServiceType.DynamicData)
{
    case WsAssemblyServiceType.DynamicData:
        if (itemName.XFEqualIsIgnoreCase("Dynamic Cube's Name"))
        {
            return new WsasDynamicData();
        }
        break;
}
```

When designing a DynamicData workspace assembly class file, consider the following three methods:

- Triggering and refresh time-stamp controls. For details, see Workspace Assembly Dynamic Dimensions > Frequency and Trigger.
- **ReadDynamicDataUnitData**: Returns a data unit object. For details, see Read Dynamic Data Unit.
- **SaveDynamicDataCells**: (Optional) Manages the write-back of numeric and text data to a source. For details, see Save Dynamic Data Cells.

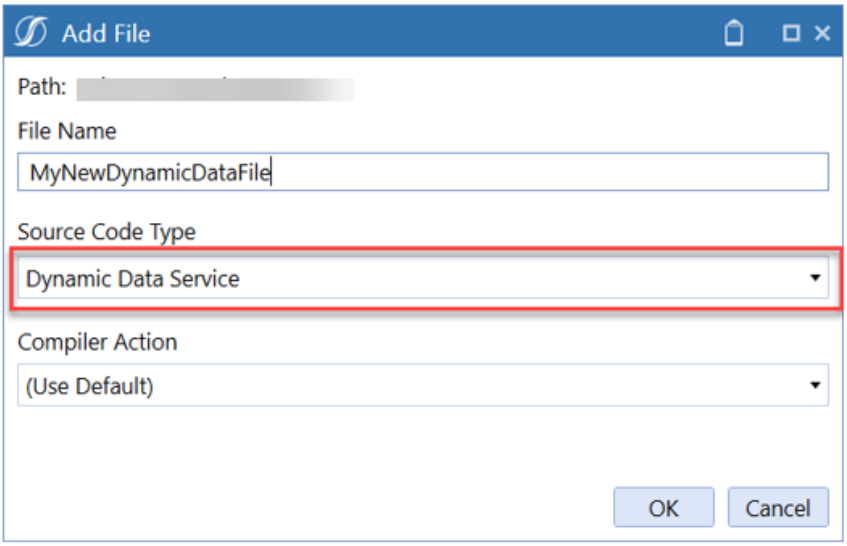
Read Dynamic Data Unit

Workspace Assembly files are designed to emulate the content of data record tables. The core definition must construct a list of data records that contain the required content of a data unit. This allows the Finance engine to properly interpret the data. The resulting data, which can be numeric or text-based, is in memory cached dynamic data.

A data unit consists of these objects:

- Entity
- Parent
- Scenario
- Consolidation
- Time (Year)

A dynamic data service enables a dynamic cube to read and write data from an internal or external source.



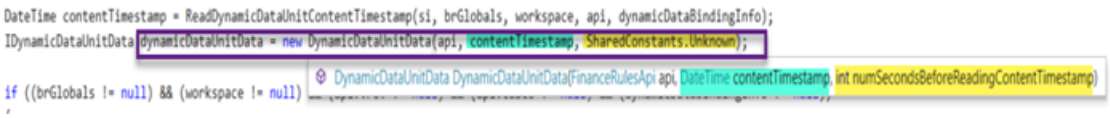
The file contains three functions to be implemented:

- **ReadDynamicDataUnitContentTimeStamp**: Return a timestamp representing when the data might have changed. This will be used to determine whether cached data should be refreshed. Refer to Workspace Assembly Dynamic Dimensions.htm.

- **ReadDynamicDataUnitData**: Return a DynamicDataUnitData object representing the requested DataUnit. The object can be populated with data coming from anywhere, including external databases and existing cubes. .
- **SaveDynamicDataCells**: This function will receive cells modified via forms, giving developers the ability to send data to external systems or custom tables. .

When creating a **DynamicDataUnitData** object to implement ReadDynamicDataUnitData, specify a timestamp indicating when the returned data was last changed ([contentTimestamp]) and how long to wait before the timestamp should be checked again (**numSecondsBeforeReadingContentTimeStamp**).

Refresh triggering options occur upon a change to the workspace assembly file. This occurs daily, hourly, and by the minute. Refer to the following example.



Data can be provided to the DynamicDataUnitData object as a DataRecord object or DataBufferCellPk:

- **DataRecord:** One or more objects, representing all values in the year for a specific intersection. For example, having 12 values in a monthly application. See DataRecord.
- **DataBufferCellPk:** Representing a specific intersection for a specific period only. For example, a single value in a single period, be it YTD or Periodic. See DataBufferCellPk.

DataRecord

DataRecord objects are typically created by specifying a DataRecordPk object (which contains the record coordinates, i.e. the members that define the intersection), and the number of cells it should have (one for each period in the year, e.g. 12 in a monthly application).

```
' define the Pk
Dim drPk As New DataRecordPk( api.Pov.Cube.CubeId, _
    api.Pov.Entity.MemberId, _
    DimConstants.Unknown, _      ' this is parentId, optional at base level
    api.Pov.Cons.MemberId, api.Pov.Scenario.MemberId, _
    2025, _          ' use straight years here, not "timeId" ones
    accountId, flowId, originId, icId, _
    u1id, u2id, u3id, u4id, u5id, u6id, u7id, u8id)
' create the record
Dim newRecord as new DataRecord(drPk, 12)
```

```
// define the Pk
var drPk = new DataRecordPk(api.Pov.Cube.CubeId,
    api.Pov.Entity.MemberId,
    DimConstants.Unknown,  // this is parentId, optional at base level
    api.Pov.Cons.MemberId, api.Pov.Scenario.MemberId,
    2025,  // use straight years here, not "timeId" ones
    accountId, flowId, originId, icId,
    u1id, u2id, u3id, u4id, u5id, u6id, u7id, u8id);
// create the record
var newRecord = new DataRecord(drPk, 12);
```

> **NOTE:** When using dynamic dimensions, IDs are automatically generated, and member names could overlap with members or regular stored dimensions. The recommended best practice is to retrieve the dynamic member ID using the external ID or name.

```
Dim accId = api.Members.GetMemberUsingExternalMemberId( _
                api.Pov.AccountDim.DimPk, myExternalId).MemberId
```

```
{
    var accId = api.Members.GetMemberUsingExternalMemberId(api.Pov.AccountDim.DimPk,
myExternalId).MemberId;
}
```

Values can then be set in the record by manipulating the DataCells list of DataCacheCell objects. Note that YTD values are recommended for best performance.

```
' index determines the period - 0 is January, 1 is February, etc
newRecord.DataCells(0).SetData(myDecimalAmount, DataCellExistenceType.IsRealData,
DataCellStorageType.NotStored)
newRecord.DataCells(1) = new DataCacheCell(myDecimalAmount, new DataCellStatus
(DataCellExistenceType.IsRealData, DataCellStorageType.NotStored)
```

```
{
    // index determines the period - 0 is January, 1 is February, etc
    newRecord.DataCells(0).SetData(myDecimalAmount, DataCellExistenceType.IsRealData,
DataCellStorageType.NotStored);
    newRecord.DataCells(1) = new DataCacheCell(myDecimalAmount, new DataCellStatus
(DataCellExistenceType.IsRealData, DataCellStorageType.NotStored));
}
```

DataRecord objects can then be added to the DynamicDataUnitData instance using the following methods:

- **setDataRecord**: To assign an individual instance
- **setDataRecords**: To assign a List or other collection of instances in bulk.

DataBufferCellPk

DataBufferCellPk objects represent individual values, and can be added directly to the DynamicDataUnitData object with the following methods:

- **SetDataCell**: Sets numeric values to a defined time at the V#YTD or V#Periodic members.
- **SetDataCellAnnotation**: Sets text values to a defined time at the V#Annotation member.
- **SetDataCellUsingCellIndex**: Sets numeric values to a zero-based time index supporting full data unit emulation at the V#YTD or V#Periodic members.
- **SetDataCellAnnotationUsingCellIndex**: Sets text values to a zero-based time index supporting full data unit emulation at the V#Annotation member.

```
dynamicDataUnitData.SetDataCellUsingCellIndex(si, dataCellMonthIndex, dataBufferCellPk,
amount, DataCellExistenceType.IsRealData, DataCellStorageType.Input);
Return dynamicDataUnitData
```

```
{
    dynamicDataUnitData.SetDataCellUsingCellIndex(si, dataCellMonthIndex, dataBufferCellPk,
amount, DataCellExistenceType.IsRealData, DataCellStorageType.Input);/* TODO ERROR: Skipped
SkippedTokensTrivia */
    return dynamicDataUnitData;
}
```

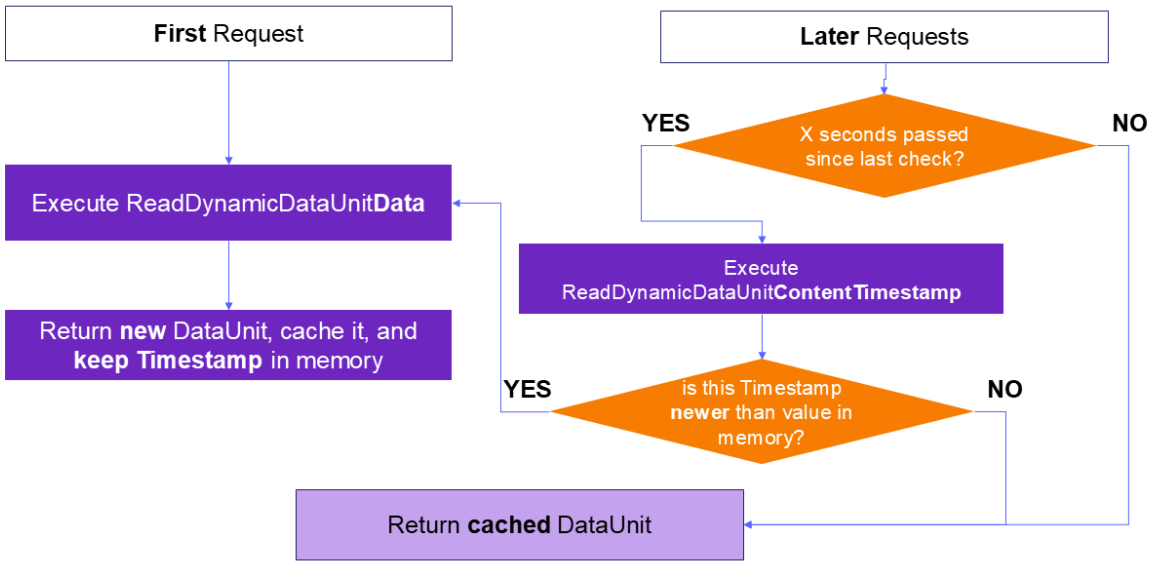
The workspace assembly file should define all the base-level dimension intersections that are valid in the dynamic cube. Data will populate all the undefined members available to the data buffer. Data buffer members can be fixed to a base level member if they are not definable in the source system, such as the origin dimension targeting Import, AdjInput or Forms. Without reference, the consolidation dimension will be populated at local and translated. Local and specified currencies can be targeted.

> **NOTE:** The generated dataunit will not be further filtered by the system on the main dimensions (Year, Scenario, Entity, Parent, Consolidation, and Cube). The system assumes that all records and cells appearing in the DynamicDataUnitData object belong to the dataunit that was requested, regardless of any DataRecordPK or DataBufferCellPK attached to the object. For example, if the requested dataunit was for entity *Chicago* and we added records pointing to entity *New York*, those records will still appear as pointing to *Chicago* !

Caching

DataUnits are cached in memory. For best performance, developers should ensure that data is served from such cache as much as possible; every execution of ReadDynamicDataUnitData will be significantly slower than just reusing cached data.

The algorithm used by OneStream is similar to the one seen for Dynamic Dimensions:



> **Example:** In the diagram above, "X seconds" is the value passed to the DynamicDataUnitData constructor as argument NumSecondsBeforeCheckingTimeStamp; likewise, the Timestamp is passed as the other argument to the same constructor.

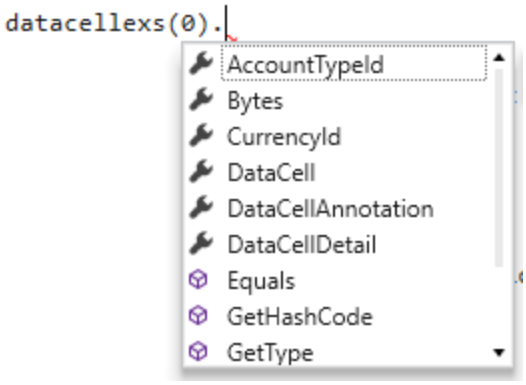
It is not possible to externally invalidate the cache. Because this logic is executed on each application server individually, if source data changes and caching timeouts are very long, a situation might arise where different users connected to different servers might observe different data until the timeout expires. If this is not acceptable, developers can set the timeout to a short value and add logic in ReadDynamicDataUnitContentTimestamp to look up a timestamp from a shared source (for example, a custom table); for best performance, this logic should be kept to a minimum, since this method could be called very often and by multiple processes.

Save Dynamic Data Cells

A dynamic cube designed for a workspace assembly using any data-binding type containing "Edit" must providemethods to manage the write-back of records. Write back is supported only on base level intersections of the O#Forms member.

The SaveDynamicDataCells method will receive a list of records edited. The records are defined by OneStream-specific member IDs. For this reason, when using external sources such as an external database, the code logic will likely need to perform mapping to align with the target record keys. The developer is completely free to define such mapping or transformation logic in code, and to use whichever process is necessary to send data back (database writes, webservices, etc).

A new dynamic data service assembly file automatically creates the **SaveDynamicDataCells** function. The function receives a list of the records that changed as **dataCellEx** objects, which have properties to retrieve all the details related to the DataCell.



Both numeric and text values can be sent back to the source system. Changed cell records are attached to the Origin and View members in the following ways:

- O#Forms:V#YTD or O#Forms:V#Periodic for numeric data
- O#Forms:V#Annotations for text data

When evaluating records, the view dimension should be used to target the results to the appropriate column or column type. Note how the example below performs checks on the View IDs to decide how data should be saved.

```
private void SaveDynamicDataCell(SessionInfo si, DbConnInfo dbConnExt, Cube cube, Member scenario, DynamicDataBindingInfo dynamicDataBindingInfo, DataCellEx dataCellEx)
{
    try
    {
        if ((dbConnExt != null) && (dataCellEx?.DataCell?.DataCellPk != null))
        {
            DataCellPk dataCellPk = dataCellEx.DataCell.DataCellPk;
            ViewMember viewMember = ViewMember.GetItem(dataCellPk.ViewId) ?? ViewMember.YTD;

            string scenarioName = BRApi.Finance.Members.GetMemberName(si, DimType.Scenario.Id, dataCellPk.ScenarioId);

            if ((viewMember.Id == ViewMember.YTD.Id) || (viewMember.Id == ViewMember.Periodic.Id))
            {
                double newAmount = (double)dataCellEx.DataCell.CellAmount;
                DataCellStatus newDataCellStatus = dataCellEx.DataCell.CellStatus;

                if (!newDataCellStatus.IsNoData)
                {
                    var modifiedValuesByColName = new Dictionary<string, object>(StringComparer.InvariantCultureIgnoreCase);
                    modifiedValuesByColName["Amount"] = newAmount;

                    BRApi.Database.InsertOrUpdateRow(dbConnExt, "FactFinance", dbWheres, defaultColNamesAndValuesForInsert, modifiedValuesByColName);
                }
                else
                {
                    BRApi.Database.DeleteRows(dbConnExt, "FactFinance", dbWheres);
                }
            }
            else if (viewMember.Id == ViewMember.Annotation.Id)
            {
```

Extracting Dynamic Dimensions

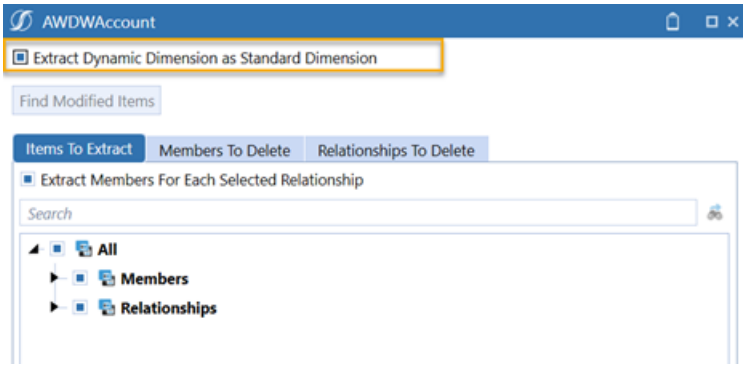
Dynamic Dimensions are defined, maintained, and initialized through workspace assembly files. For dynamic dimensions and dynamic cubes to functions in other applications and environments, workspaces must be present, as well as connectivity to required external sources. Where underlying requirements are not supported, or the requirement for a dynamic dimension does not exist, dynamic dimensions can be extracted as a standard dimensions.

Note the following:

- They are only supported by dimensions using metadata extract.
- Dimensions extracted as dynamic dimensions only present the dimension properties in the XML output.

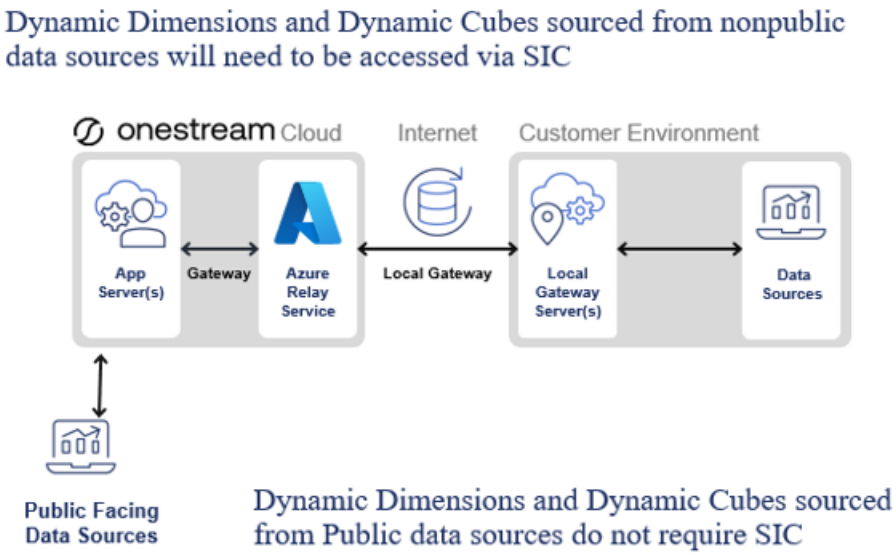
To convert a dynamic dimension to a standard dimension:

1. Go to **Application** > **Load Extract**.
2. Select **Metadata** as the extract choice.
3. Select the dynamic dimension you want to extract.
4. Right-click and select **Select Member to Extract**.
5. In the dialog box, check the box labeled **Extract Dynamic Dimension as Standard Dimension**.



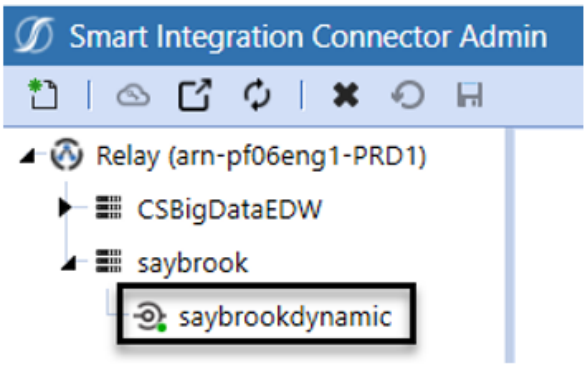
Dynamic Dimensions and Smart Integration Connector

In some instances, dynamic dimensions and dynamic cubes access data sources that are public facing. In other instances, data sources are in a virtual private network (VPN) and require the use of Smart Integration Connector (SIC) to access. Refer to the following image.



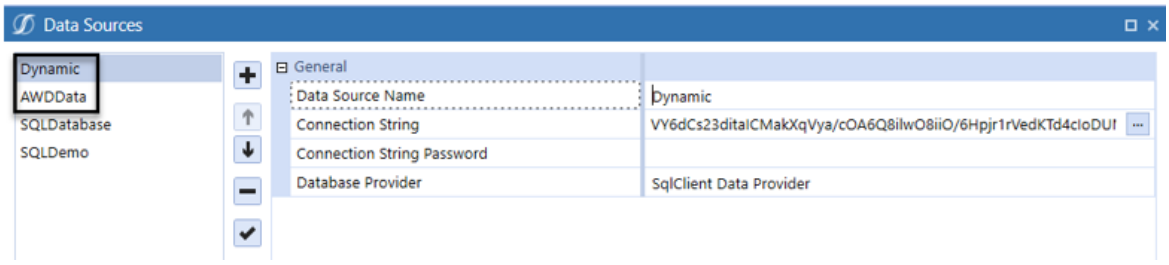
Smart Integration Connector setup is required to access non-public data sources.

1. A local gateway server should be installed and configured. There can be multiple servers including, development, production, disaster recovery, high availability, and multiple geographic locations depending on requirements.
2. The local gateway server should be added to OneStream.

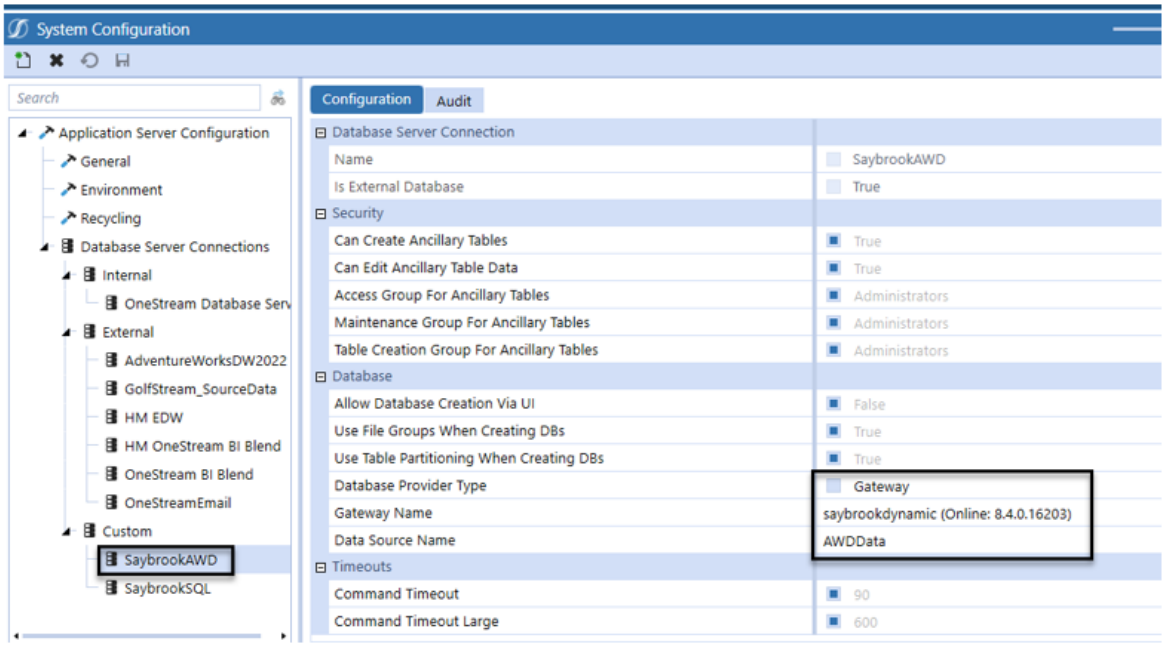


Local data sources must be green, indicating it is accessible.

3. Local data sources must be configured on the local gateway server.



4. Local data sources must be added in the system configuration as Gateway Provider Types.

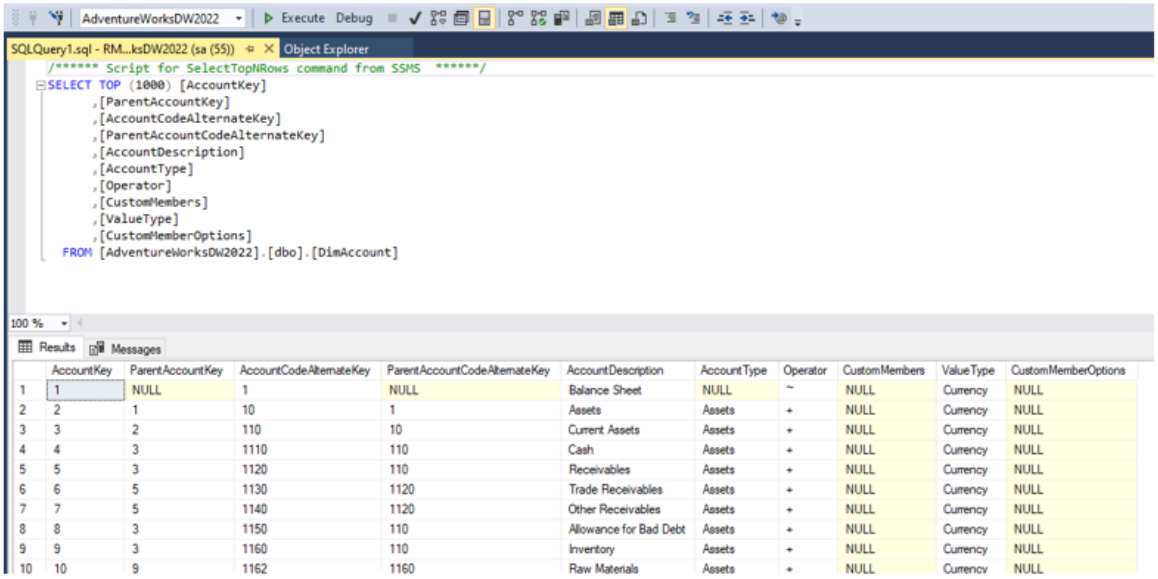


5. Data sources can be referenced in the workspace assembly.

```
using (DbConnInfo dbConnExt = BRApi.Database.CreateExternalDbConnInfo(si, "SaybrookAWD"))
```

The code example above allows the OneStream cloud to communicate with the non-public data sources.

The example in the image below is a dynamic hierarchy assembly that accesses a data source with a Smart Integration Connector.



	AccountKey	ParentAccountKey	AccountCodeAlternateKey	ParentAccountCodeAlternateKey	AccountDescription	AccountType	Operator	CustomMembers	ValueType	CustomMemberOptions
1	1	NULL	1	NULL	Balance Sheet	NULL	~	NULL	Currency	NULL
2	2	1	10	1	Assets	Assets	+	NULL	Currency	NULL
3	3	2	110	10	Current Assets	Assets	+	NULL	Currency	NULL
4	4	3	1110	110	Cash	Assets	+	NULL	Currency	NULL
5	5	3	1120	110	Receivables	Assets	+	NULL	Currency	NULL
6	6	5	1130	1120	Trade Receivables	Assets	+	NULL	Currency	NULL
7	7	5	1140	1120	Other Receivables	Assets	+	NULL	Currency	NULL
8	8	3	1150	110	Allowance for Bad Debt	Assets	+	NULL	Currency	NULL
9	9	3	1160	110	Inventory	Assets	+	NULL	Currency	NULL
10	10	9	1162	1160	Raw Materials	Assets	+	NULL	Currency	NULL

In this example, the name of the Smart Integration Connector gateway connection is SaybrookAWD.

Master Data Considerations

Dynamic cubes and dynamic dimensions let you dynamically add members from external systems. Master data can then be managed by a central master data tool.

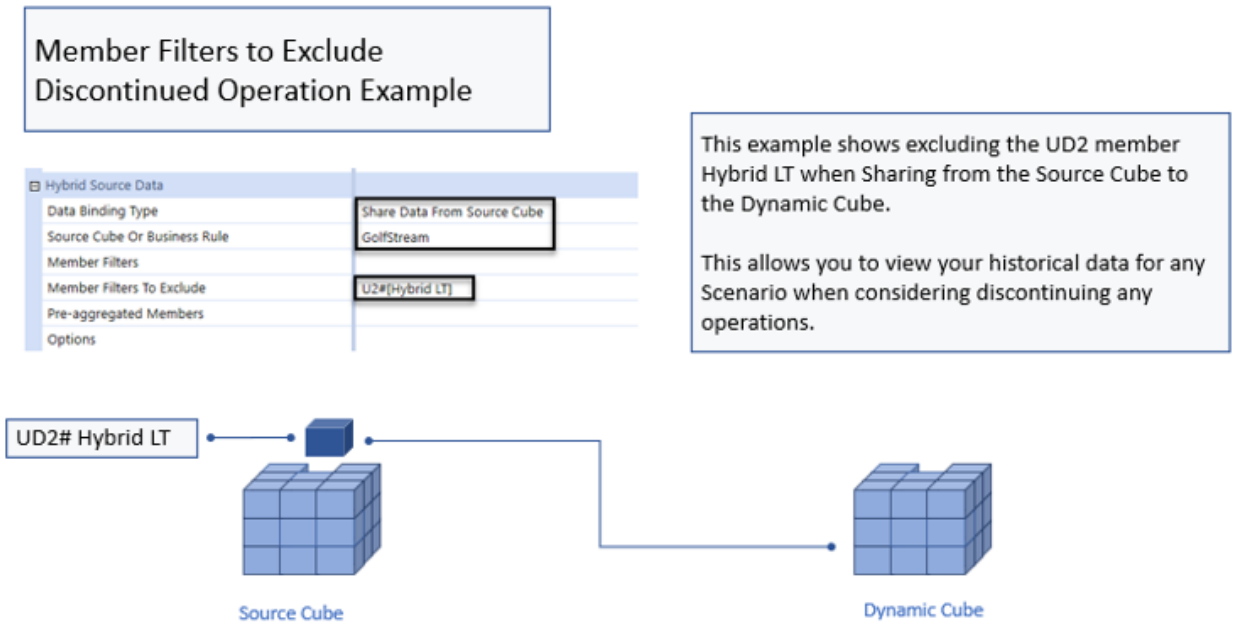
Obtaining data from external systems may introduce member names that already exist in the system. Depending on the configuration of the data sources, this means that you might have existing members with the same name, but with different meanings and could lead to misinterpretation of data sets. For example, you could have a product member by the name of 10000 with a description of Widgets. An external data source might also have a product member named 10000, but in this external system, the member descriptions are Pencils. This could occur with multiple external data sources.

Use a master data tool to manage all of this for you so that members have the same meaning across the enterprise.

Dynamic Dimension Use Cases

Dynamic Dimensions can be created from external sources that feed into dynamic cubes, or to define volatile structures with existing members.

Cube filters let you exclude a member, or multiple members and dimensions. This allows you to perform data analysis for various scenarios. For example, you can exclude a member when considering the discontinuation of an operation.



Dynamic Cube Use Cases

Large data units can impact application reporting performance. Dynamic Cubes can focus the data unit on filtered intersections of data. The focused data unit can return a smaller subset of data across all scenarios, as defined by cube properties.

Dynamic Cubes can also be defined using stage attributes, marketplace registers, and custom datatables; this can enable easy reporting of non-cube data using cube-based tools like the Excel Add-in, cube views, and so on.

Dynamic Cubes can be used to expose data produced by BI Blend as cubes, making it easier to mix with existing cube data. For best results, ensure the following:

- Align the blend unit with the entity to pre-aggregate results
- Leverage cube dimensions for transformation and aggregation points
- Build aggregation points as derivative rules for pre-aggregated values
- Use UD8 aggregation info in a dynamic dimension to generate metadata and aggregations
- Plan required calculations using derivative rules and dynamic dimension member formulas

Dynamic cubes used with external relational tables require Smart Integration Connector for non-public facing sources.

Dynamic Cube Services API Reference

This guide includes classes required for implementation of Dynamic Cube Services.

Common Arguments

The following argument types are common to multiple methods and services.

Name	Type	Description
si	SessionInfo	Generic information about the current user session, including the type of client. See the generic API Guide for details.
brGlobals	BRGlobals	In-memory store for basic data, which can be used to communicate between different methods of the same class as part of the execution flow. Note that its lifespan is short: as soon as a data request is fully satisfied (i.e. a dimension or dataunit has been created and returned), the object is destroyed.

Services

Dynamic Cube Services must be enabled via a Service Factory. See the section on Configuring a Service Factory for more details on the overall approach.

Dynamic Dimension Service

This type of service is responsible for creating and refreshing a Dynamic Dimension. It should be returned by the Service Factory when an instance of type WsAssemblyServiceType.DynamicDimension is requested.

It must implement interface IWsasDynamicDimensionV800, which requires the following methods:

ReadDynamicDimensionContentTimestamp

Return Type: DateTime

This method returns a timestamp representing the last time the dimension has changed. If the system receives a timestamp more recent than the one associated with the last execution of ReadDynamicDimensionInfo, that method is executed again to rebuild the dimension.

ReadDynamicDimensionContentTimestamp will be executed at periodic intervals, as specified by the DynamicDimensionInfo object returned by ReadDynamicDimensionInfo. In addition, different application servers might execute it at different times, and it might be triggered by a number of internal events. For this reason, it is recommended to keep its execution time as short as possible (i.e. very simple logic, few or no I/O calls, etc), to avoid impacting system performance.

Note that individual application servers maintain their memory separately, so they might execute this method at different times, potentially returning different timestamps and hence letting users work with different hierarchies. Cache coordination can be implemented with custom logic between this method and ReadDynamicDimensionInfo.

Specific Arguments

api As IWsasDynamicDimensionApiV800 - see the related section

args As DynamicDimensionArgs - see the related section

ReadDynamicDimensionInfo

Return Type: DynamicDimensionInfo

This method is responsible for creating and returning the DynamicDimensionInfo object representing the generated Dimension. It will be executed the first time the dimension is requested; the resulting object will be kept in the application server memory until its timestamp is found older than the one returned by a call to ReadDynamicDimensionContentTimestamp, at which point it will be discarded and replaced with a new execution of this method.

Note that individual application servers maintain their memory separately, so they might execute this method at different times, potentially producing different hierarchies. Cache coordination can be implemented with custom logic between this method and ReadDynamicDimensionContentTimestamp.

Specific Arguments

api As IWsasDynamicDimensionApiV800 - see the related section

args As DynamicDimensionArgs - see the related section

Service-specific arguments

Api as IWsasDynamicDimensionApiV800

This object provides various utilities for building and maintaining dynamic dimensions.

Property	Type	Description
DbConnFW	DbConnInfo	Open connection to the Framework database
DbConnApp	DbConnInfo	Open connection to the Application database

Property	Type	Description
BRGlobals	BRGlobals	In-memory store for basic data, which can be used to communicate between different methods of the same class as part of the execution flow. Note that its lifespan is short: as soon as a data request is fully satisfied (i.e. a dimension or dataunit has been created and returned), the object is destroyed.
Workspace	DashboardWorkspace	Reference to the Workspace containing the Assembly, including Text properties.

Methods

ReadDynamicDimensionInfo

Return Type: DynamicDimensionInfo

This method will build the entire hierarchy by executing an SQL query and mapping the resulting columns to member properties. It is not compulsory to use this function, but it is recommended whenever source member definitions are in SQL tables. Note that items can be further customized by manipulating the returned object.

Specific Arguments:

DateTime contentTimestamp

The Timestamp to associate with this execution. As long as it continues to be more recent than what returned by ReadDynamicDimensionContentTimestamp, the system will not refresh the dimension.

Int numSecondsBeforeReadingContentTimestamp

How often, in seconds, the server should call RadDynamicDimensionContentTimestamp to verify cache validityi

DbConnInfo dbConnExt

Open connection to the database where the query will be executed. Please consider creating it with a Using block, for safety.

string selectStatement

The SELECT statement to execute.
Note that WHERE and ORDER BY clauses should be omitted, and provided separately with their dedicated arguments listed below; this will allow for performance optimizations in the background.

List<DbWhere> dbWheres

WHERE clauses to be appended to the query, added to a List. For example a clause like "WHERE isEnabled = 'true' " should be expressed as:

```
Dim dbWheres as new List(Of DbWhere)dbWheres.Add(New DbWhere
("isEnabled", DbOperator.IsEqualTo, "true")
```

List<DbOrderBy> dbOrderBys

ORDER BY clauses to be appended to the query, added to a List. Ordering records will influence the position of Members in the generated dimension.

For example, a clause like "ORDER BY Name ASC" should be expressed as:

```
Dim dbOrderBys As New List(Of DbOrderBy)dbOrderBys.Add(New DbOrderBy
("Name", True))
```

string memberNameColumn

Name of the table field that contains member names.

string memberDescriptionColumn,

Name of the table field that contains member descriptions.

string externalMemberIdColumn,

Name of the table field that contains individual IDs for each member as present in the source system. OneStream will generate its own set of IDs, that will be automatically associated with the external ones in order to maintain a relationship with the source definitions.

string externalMemberNameColumn

name of the table field that contains member names as they appear in the source system. Note that it can be the same as memberNameColumn.

string parentMemberNameColumn

name of the table field that contains the names of the direct parent of a member. This will be used to build the hierarchy.

string parentExternalMemberIdColumn

name of the table field that contains the ID of the direct parent of a member, as it appears in the source system. If present, it will be used to build the hierarchy, taking precedence over parentMemberName and parentExternalMemberName.

string parentExternalMemberNameColumn

name of the table field that contains the name of the direct parent of a member, as it appears in the source system. If present, it will be used to build the hierarchy, taking precedence over parentMemberName.

string aggWeightColumn

name of the table field that will contain values to be used for the Aggregation Weight property of parent-child relationships.

bool aggWeightIsPercentageNotFraction

Boolean to specify whether the values found in aggWeightColumns are percentages (for example, 75) or fractions (for example, 0.75).

CreateAndAddDynamicMembers

This method can create several Dynamic Members starting from a source DataTable, adding them to a DynamicMemberCollection instance.

Specific arguments

DataTable dataTable

The DataTable object containing member data.

string memberNameColumn

Name of the table field that contains member names.

string memberDescriptionColumn,

Name of the table field that contains member descriptions.

string externalMemberIdColumn,

Name of the table field that contains individual IDs for each member as present in the source system. OneStream will generate its own set of IDs, that will be automatically associated with the external ones in order to maintain a relationship with the source definitions.

string externalMemberNameColumn

name of the table field that contains member names as they appear in the source system. Note that it can be the same as memberNameColumn.

DynamicMemberCollection dynamicMembers

The collection object that will receive newly-created members.

CreateAndAddDynamicRelationships

This method can create several DynamicRelationships and add them to a DynamicRelationshipCollection object, starting from a DataTable that expresses such relationships. Note that any Dynamic Member involved must already exist in a DynamicMemberCollection object. See the sections on DynamicRelationship and DynamicRelationshipCollection for other details.

Specific arguments

DataTable dataTable

The datatable object containing relationship information.

DynamicMemberCollection dynamicMembers

The collection object containing all DynamicMembers referenced in the relationships. See the relevant section.

string memberNameColumn

Name of the table field that contains member names.

string externalMemberIdColumn,

Name of the table field that contains individual IDs for each member as present in the source system. OneStream will generate its own set of IDs, that will be automatically associated with the external ones in order to maintain a relationship with the source definitions.

string externalMemberNameColumn

name of the table field that contains member names as they appear in the source system. Note that it can be the same as memberNameColumn.

string parentMemberNameColumn

name of the table field that contains the names of the direct parent of a member. This will be used to build the hierarchy.

string parentExternalMemberIdColumn

name of the table field that contains the ID of the direct parent of a member, as it appears in the source system. If present, it will be used to build the hierarchy, taking precedence over parentMemberName and parentExternalMemberName.

string parentExternalMemberNameColumn

name of the table field that contains the name of the direct parent of a member, as it appears in the source system. If present, it will be used to build the hierarchy, taking precedence over parentMemberName.

string aggWeightColumn

name of the table field that will contain values to be used for the Aggregation Weight property of parent-child relationships.

bool aggWeightIsPercentageNotFraction

Boolean to specify whether the values found in aggWeightColumns are percentages (for example, 75) or fractions (for example, 0.75).

DynamicRelationshipCollection dynamicRelationships

The collection object that will receive newly-created relationships. See the relevant section.

Args as DynamicDimensionArgs

This object provides context-specific properties of the dimension.

Properties

- Dim: Dim
    - Description: Generic properties for the dimension, including source configuration. Name Value Pairs defined on the Dimension configuration can be found in its DimMemberSourceNVPairs property1.
- CachedDynamicDimensionInfo: DynamicDimensionInfo
    - Description: The dimension structure as it currently exists in cache. Note that this object can be null (for example, on first execution on a given appserver)

Dynamic Data Service

This service is responsible for creating and refreshing a Dynamic DataUnit for a Dynamic Cube. It should be returned by the Service Factory when an instance of type WsAssemblyServiceType.DynamicData is requested.

It must implement interface IWsasDynamicDataV800, which requires the following methods:

ReadDynamicDataUnitContentTimestamp

Return Type: DateTime

This method returns a timestamp representing the last time the dimension has changed. If the system receives a timestamp more recent than the one associated with the last execution of ReadDynamicDataUnitData, that method is executed again to rebuild the dimension.

ReadDynamicDataUnitContentTimestamp will be executed at periodic intervals, as specified by the DynamicDataUnitData object returned by ReadDynamicDataUnitData. In addition, different application servers might execute it at different times, and it might be triggered by a number of internal events. For this reason, it is recommended to keep its execution time as short as possible (i.e. very simple logic, few or no I/O calls, etc), to avoid impacting system performance.

Note that individual application servers maintain their memory separately, so they might execute this method at different times, potentially returning different timestamps and hence letting users work with different data. Cache coordination can be implemented with custom logic between this method and ReadDynamicDataUnitData.

Specific Arguments

api As FinanceRulesApi - see related section

dynamicDataBindingInfo As DynamicDataBindingInfo - see related section

ReadDynamicDataUnitData

Return Type: DynamicDataUnitData

This method is responsible for creating and returning the DynamicDataUnitData object representing the generated DataUnit. It will be executed the first time data is requested; the resulting object will be kept in the application server memory until its timestamp is found older than the one returned by a call to ReadDynamicDataUnitContentTimestamp, at which point it will be discarded and replaced with a new execution of this method.

Note that individual application servers maintain their memory separately, so they might execute this method at different times, potentially producing different data. Cache coordination can be implemented with custom logic between this method and ReadDynamicDataUnitContentTimestamp.

Specific Arguments

api As FinanceRulesApi - see related section

dynamicDataBindingInfo As DynamicDataBindingInfo - see related section

SaveDynamicDataCells

This method can be used to receive data entered by users through a Cube View or Form, and save it to a source system or table.

IMPORTANT: DO NOT WRITE INTO INTERNAL ONESTREAM TABLES. Always save data in custom tables or external systems.

Specific Arguments

cube As Cube

Object representing Cube information and configuration.

scenario As Member

The Scenario containing modified cells.

dynamicDataBindingInfo As DynamicDataBindingInfo

see related section

dataCellExs As List(Of DataCellEx)

List of datacells containing submitted data.

Service-specific arguments

api As FinanceRulesApi

This object contains methods and properties typically used in Finance Business Rules, see the generic API Guide for details.

IMPORTANT: although the object contains several "Set" methods to save data into cubes (api.Data.SetDataCell etc), they should NEVER be invoked in this service.

dynamicDataBindingInfo As DynamicDataBindingInfo

This object contains information related to the configuration of the dynamic cube. Note that this is also available in Hybrid Scenario rules. All properties are Read-Only.

Properties

- **IsCubeNotScenarioDataBinding**: `Boolean`
  - Indicates whether the binding is used in a Dynamic Data Service (`True`) or in a Hybrid Scenario Business Rule (`False`).
- **DataBindingType**: `DynamicDataBindingType`
  - Represents the value of the "Data Binding Type" property.
- **SourceCubeOrScenarioOrBRName**: `String`
  - Contains the value of the "Source Cube or Workspace Assembly" property for cubes, or the "Source Scenario or Workspace Assembly" property for scenarios.
- **MemberFilters**: `String`
  - Description: Holds the value of the "Member Filters" property.
- MemberFiltersToExclude: String
  - Value of property "Member Filters to Exclude"
- PreAggregatedMembers: String
  - Value of property "Pre-aggregated Members"
- EndYear: int
  - In Hybrid Scenario Business Rules, the last Year containing data retrieved from Source
- Options: String

- Value of property "Options"

- OptionsNameValuePairs: IReadOnlyDictionary(Of String, String)
  - Value of property "Options", parsed as a collection of Name-Value Pairs

Related Classes

The following object types are necessary to most Dynamic Cube Services implementations.

> **NOTE:** In the descriptions below, basic methods and properties (for example, ToString) have been omitted.

> **NOTE:** Implementations typically require knowledge of other classes and BRApi functions, typically found in Finance Business Rule (Member, DataCell, etc) and Connector Business Rules (DbConnInfo, etc).

DataBufferCellPk Class

The DataBufferCellPk class is responsible for managing primary keys for data buffer cells, i.e. the combination of dimension members that points to an individual value. This is the same class typically found in advanced Finance Business Rules related to calculations.

Properties

AccountId: int	Gets or sets the account ID.
FlowId: int	Gets or sets the flow ID.

OriginId: int	Gets or sets the origin ID.
ICId: int	Gets or sets the intercompany ID.
UD1Id: int	Gets or sets the user-defined dimension 1 ID.
UD2Id: int	Gets or sets the user-defined dimension 2 ID.
UD3Id: int	Gets or sets the user-defined dimension 3 ID.
UD4Id: int	Gets or sets the user-defined dimension 4 ID.
UD5Id: int	Gets or sets the user-defined dimension 5 ID.
UD6Id: int	Gets or sets the user-defined dimension 6 ID.
UD7Id: int	Gets or sets the user-defined dimension 7 ID.
UD8Id: int	Gets or sets the user-defined dimension 8 ID.

Constructors

DataBufferCellPk()	Initializes a new instance of the DataBufferCellPk class.

DataBufferCellPk(DataBufferCellPk from)	Initializes a new instance of the DataBufferCellPk class by copying an existing instance.
DataBufferCellPk(DataCellPk dataCellPk)	Initializes a new instance of the DataBufferCellPk class by copying an existing DataCellPk instance.
DataBufferCellPk(DataRecordPk dataRecordPk)	Initializes a new instance of the DataBufferCellPk class by copying an existing DataRecordPk instance.
DataBufferCellPk(int initialMemberIdForAllDims)	Initializes a new instance of the DataBufferCellPk class with the same initial member ID for all dimensions.
DataBufferCellPk(int accountId, int flowId, int originId, int icId, int ud1Id, int ud2Id, int ud3Id, int ud4Id, int ud5Id, int ud6Id, int ud7Id, int ud8Id)	Initializes a new instance of the DataBufferCellPk class with specified IDs for all dimensions.

Methods

CreateUnknown(): static DataBufferCellPk	Creates an instance of DataBufferCellPk with unknown values.
AreAllMembersSetToAll(): bool	Checks if all members are set to "all".

DataCacheCell Class

The `DataCacheCell` class represents a DataCell kept in memory.

Properties

CellAmount: `decimal`	Gets or sets the cell amount.
CellStatus: `DataCellStatus`	Gets or sets the cell status.

Constructors

DataCacheCell()	Initializes a new instance of the `DataCacheCell` class.
DataCacheCell(DataCacheCell from)	Initializes a new instance of the `DataCacheCell` class by copying an existing instance.
DataCacheCell(DataCacheCell from, decimal aggWeight)	Initializes a new instance of the `DataCacheCell` class by copying an existing instance and applying an aggregation weight.

DataCacheCell(DataCellStorageType dataCellStorageType)	Initializes a new instance of the `DataCacheCell` class with the specified data cell storage type.
DataCacheCell(DataCellExistenceType dataCellExistenceType, DataCellStorageType dataCellStorageType)	Initializes a new instance of the `DataCacheCell` class with the specified data cell existence type and storage type.
DataCacheCell(decimal cellAmount, DataCellStatus cellStatus)	Initializes a new instance of the `DataCacheCell` class with the specified cell amount and cell status.

Methods

SetData(decimal cellAmount, DataCellExistenceType dataCellExistenceType, DataCellStorageType dataCellStorageType)	Sets the data for the cell in memory.
Aggregate(DataCacheCell sourceDataCacheCell, decimal aggWeight, bool changeDerivedDataToRealData)	Aggregates the data from the source data cache cell with the specified aggregation weight and changes data type from Derived to Real if specified.

AggregateDataCacheCells (DataCacheCell[] sourceDataCells, DataCacheCell[] resultDataCells): `static`	Aggregates data cache cells from the source array to the rest array.
ClearDataCacheCells(DataCacheCell[] dataCells, int lastPeriodIndexToClear): `static bool`	Clears the data cache cells up to the specified last period index.
AreDataCacheCellsEqual (DataCacheCell[] dataCells1, DataCacheCell[] dataCells2): `static bool`	Checks if two arrays of data cache cells are equal.

DataRecord Class

`DataRecord` objects represent a record in the DataUnit. Each record points to a specific combination of members for all dimensions, and contains all values of that intersection in a single year.

Properties

DataRecordPk: `DataRecordPk`	Gets or sets the data record primary key, which holds references to the dimension members defining the intersection.

DataCells: `DataCacheCell[]`	Gets or sets the data cells. Note that cells can also be accessed directly from the object, e.g. Dim myCell as DataCacheCell = myDataRecordObject(0).

Constructors

DataRecord()	Initializes a new instance of the `DataRecord` class.
DataRecord(int numDataCells)	Initializes a new instance of the `DataRecord` class with the specified number of data cells, which should typically be the same as the amount of periods required in a DataUnit. For example, 12 for a monthly application.
DataRecord (DataRecordPk dataRecordPk, int numDataCells)	Initializes a new instance of the `DataRecord` class with the specified data record primary key and number of data cells, which should typically be the same as the amount of periods required in a DataUnit. For example, 12 for a monthly application..
DataRecord (DataRecordPk dataRecordPk, DataCacheCell[] dataCells)	Initializes a new instance of the `DataRecord` class with the specified data record primary key and data cells.

DataRecord (DataRecord from)	Initializes a new instance of the `DataRecord` class by copying an existing instance.

Methods

Aggregate(DataRecord sourceDataRecord, decimal aggWeight)	Aggregates the data from the source data record with the specified aggregation weight.
Aggregate(DataRecord sourceDataRecord, decimal aggWeight, bool changeDerivedDataToRealData, int lastPeriodIndexToAggregate)	Aggregates the data from the source data record with the specified aggregation weight, changes derived data to real data if specified, and aggregates up to the specified last period index.
HasNonZeroCells(): `bool`	Checks if the data record has non-zero cells.
GetDataCellArrayLengthInYearForFrequency (SessionInfo si, TimeDimAppInfoEx timeDimAppInfo, int year, Frequency frequency): `static int`	Gets the length of the data cell array for a given year and frequency.

ResizeDataCellArrayIfNecessary(TimeDimType timeDimType, int newNumDataCells): `bool`	Resizes the data cell array if necessary.
ConvertToAnotherFrequency(SessionInfo si, DataRecord sourceDataRecord, TimeDimAppInfoEx timeDimAppInfo, int sourceFreqId, int destFreqId, ref bool foundDestCellsThatAreDerivedOrRealData): `static DataRecord`	Converts the data record to another frequency.
ConvertDerivedDataCellToRealDataIfNecessary (DataCacheCell[] sourceDataCells, int indexOfSourceCellUsedToCreatePreviousDestCell, int indexOfSourceCellUsedToCreateThisDestCell, DataCacheCell destDataCell, ref bool foundDestCellsThatAreDerivedOrRealData): `static`	Converts derived data cells to real data if necessary.
CreateNoDataDataRecord(DataRecordPk dataRecordPk, int numDataCells): `static DataRecord`	Creates a data record with no data.

DataRecordPk Class

The `DataRecordPk` class is responsible for managing primary keys for data records, which means holding references to the dimension members defining the intersections that the record will contain.

Properties

CubeId: `int`	Gets or sets the cube ID.
EntityId: `int`	Gets or sets the entity ID.
ParentId: `int`	Gets or sets the parent entity ID.
ConsId: `int`	Gets or sets the consolidation ID.
ScenarioId: `int`	Gets or sets the scenario ID.
YearId: `int`	Gets or sets the year ID.
AccountId: `int`	Gets or sets the account ID.
FlowId: `int`	Gets or sets the flow ID.
OriginId: `int`	Gets or sets the origin ID.
ICId: `int`	Gets or sets the intercompany ID.
UD1Id: `int`	Gets or sets the user-defined dimension 1 ID.
UD2Id: `int`	Gets or sets the user-defined dimension 2 ID.

UD3Id: `int`	Gets or sets the user-defined dimension 3 ID.
UD4Id: `int`	Gets or sets the user-defined dimension 4 ID.
UD5Id: `int`	Gets or sets the user-defined dimension 5 ID.
UD6Id: `int`	Gets or sets the user-defined dimension 6 ID.
UD7Id: `int`	Gets or sets the user-defined dimension 7 ID.
UD8Id: `int`	Gets or sets the user-defined dimension 8 ID.
this[int dimTypeId]: `int`	Returns the ID of the member for the dimension specified. Example: Dim myUd8MemberID as Integer = myDataRecordPK(DimType.UD8.Id)

Constructors

DataRecordPk()	Initializes a new instance of the `DataRecordPk` class.
DataRecordPk(DataRecordPk from)	Initializes a new instance of the `DataRecordPk` class by copying an existing instance.

DataRecordPk(int cubeId, int entityId, int parentId, int consId, int scenarioId, int yearId, int accountId, int flowId, int originId, int icId, int ud1Id, int ud2Id, int ud3Id, int ud4Id, int ud5Id, int ud6Id, int ud7Id, int ud8Id)	Initializes a new instance of the `DataRecordPk` class with specified IDs for all dimensions.
DataRecordPk(int cubeId, int entityId, int parentId, int consId, int scenarioId, int yearId, DataBufferCellPk dataBufferCellPk)	Initializes a new instance of the `DataRecordPk` class with specified IDs for DataUnit dimensions, retrieving IDs for the other dimensions from a provided `DataBufferCellPk` instance.
DataRecordPk(DataCellPk dataCellPk, int yearId)	Initializes a new instance of the `DataRecordPk` class for the provided year, retrieving member IDs from the provided `DataCellPk` instance.
DataRecordPk(DataUnitCachePk dataUnitCachePk, DataBufferCellPk dataBufferCellPk)	Initializes a new instance of the `DataRecordPk` class retrieving member IDs from the provided `DataUnitCachePk` and `DataBufferCellPk` instances.

DynamicDataUnitData

The `DynamicDataUnitData` class represents a DataUnit, which contains all records pointing to a specific Scenario/Entity/Year combination.

Properties

ContentTimestamp: `DateTime`	Gets or sets the content timestamp. See the section on ReadDynamicDataUnitContentTimestamp for more information.
NumSecondsBeforeReadingContentTimestamp: `int`	The number of seconds before the content timestamp will be re-checked.
NumTimePeriodsPerDataRecord: `int`	Gets the number of time periods per data record. This depends on the application configuration; for example, a monthly application will require 12 periods per record.

Constructors

DynamicDataUnitData(FinanceRulesApi api)	Initializes a new instance of the `DynamicDataUnitData` class using the specified `FinanceRulesApi` instance.
DynamicDataUnitData(FinanceRulesApi api, DateTime contentTimestamp, int numSecondsBeforeReadingContentTimestamp)	Initializes a new instance of the `DynamicDataUnitData` class with the specified `FinanceRulesApi` instance, content timestamp, and number of seconds before checking the content timestamp.

Methods

GetDataRecordCollection(): `DataRecordCollection`	Returns the collection of DataRecord. This is a special List-like class with utility methods, see the relevant section for details.

GetDataCellAnnotations(): `DynamicDataUnitAnnotationCollection`	Returns a collection of data cell annotations. This is a special List-like class with utility methods, see the relevant section for details.
SetDataRecords(SessionInfo si, IEnumerable dataRecords)	Assign to the DataUnit all records contained in the provided IEnumerable. For example, a List or Array..
SetDataRecord(SessionInfo si, DataRecord dataRecord)	Add a single data record to the DataUnit.
SetDataCell(SessionInfo si, int timeId, DataBufferCellPk dataBufferCellPk, decimal cellYTDAmount, DataCellExistenceType dataCellExistenceType, DataCellStorageType dataCellStorageType)	Set a single data cell into the DataUnit. Note that it's typically preferable to work with DataRecords for better performance.
SetDataCellUsingCellIndex(SessionInfo si, int zeroBasedCellIndex, DataBufferCellPk dataBufferCellPk, decimal cellYTDAmount, DataCellExistenceType dataCellExistenceType, DataCellStorageType dataCellStorageType)	Sets a data cell using the cell index, where an index of 0 is the first period in the year, 1 is the second period, etc. Note that it's typically preferable to work with DataRecords for better performance.

SetDataCellAnnotation(SessionInfo si, int timeId, int viewId, DataBufferCellPk dataBufferCellPk, string textValue)	Sets a data cell annotation.
SetDataCellAnnotationUsingCellIndex (SessionInfo si, int zeroBasedCellIndex, int viewId, DataBufferCellPk dataBufferCellPk, string textValue)	Sets a data cell annotation using the cell index, where an index of 0 is the first period in the year, 1 is the second period, etc.

DynamicDimensionInfo

The `DynamicDimensionInfo` class represents a generated dimension.

Properties

ContentTimestamp: `DateTime`	Gets or sets the content timestamp.
NumSecondsBeforeReadingContentTimestamp: `int`	Gets the number of seconds before reading the content timestamp.
Members: `DynamicMemberCollection`	Gets the collection of dynamic members.
Relationships: `DynamicRelationshipCollection`	Gets the collection of dynamic relationships.

Constructors

- **DynamicDimensionInfo(DateTime contentTimestamp, int numSecondsBeforeReadingContentTimestamp, DynamicMemberCollection members, DynamicRelationshipCollection relationships)**

Initializes a new instance of the `DynamicDimensionInfo` class with the specified content timestamp, number of seconds before reading the content timestamp, dynamic members, and dynamic relationships.

DynamicMember

The `DynamicMember` class represents a single member of a Dynamic Dimension. It is effectively a wrapper around a regular Member object. In most cases, developers should obtain DynamicMember instances from the utility methods of a DynamicMemberCollection object.

IMPORTANT: if creating a Member instance from scratch, you should generate a unique ID with the **GetOrCreateMemberId method from a DynamicMemberCollection object.**

Properties

Member: `Member`	Gets or sets the Member object.
VaryingMemberProperties: `VaryingMemberProperties`	Gets or sets the varying member properties.
MemberDescriptions: `Dictionary<string, MemberDescription>`	Gets or sets language-specific member descriptions.

Constructors

DynamicMember()	Initializes a new instance of the `DynamicMember` class.
DynamicMember(Member member, VaryingMemberProperties varyingMemberProperties, Dictionary<string, MemberDescription> memberDescriptions)	Initializes a new instance of the `DynamicMember` class with the specified member, varying member properties, and member descriptions.
DynamicMember(int dimTypeId, int memberId, string name, string description, int dimId)	Initializes a new instance of the `DynamicMember` class with the specified dimension type ID, member ID, name, description, and dimension ID.

DynamicMemberCollection

The `DynamicMemberCollection` class contains all members that will be available in a dynamic dimension. Optionally, it will maintain a set of mappings between members and their related information in an external system. The class also contains a number of utility methods to create Dynamic Members that can be added to the collection.

Properties

Count: `int`	Gets the number of members in the collection.

Constructors

DynamicMemberCollection()	Initializes a new instance of the `DynamicMemberCollection` class.

Methods

Add(SessionInfo si, DynamicMember member)	Adds a dynamic member to the collection.
CreateMember(SessionInfo si, Member member): `static DynamicMember`	Creates a dynamic member from a `Member` instance.
CreateAndAddMember(SessionInfo si, Member member): `DynamicMember`	Creates a dynamic member from a Member instance and adds it to the collection.
GetAllMembers(): `List<DynamicMember>`	Gets all dynamic members in the collection.
GetMember(int memberId): `DynamicMember`	Gets a dynamic member by its ID.

GetMember(string memberName): `DynamicMember`	Gets a dynamic member by its name.
GetMemberUsingExternalId(int externalMemberId): `DynamicMember`	Gets a dynamic member using its external ID.
GetMemberUsingExternalName(string externalMemberName): `DynamicMember`	Gets a dynamic member using its external name.
Clear()	Clears all dynamic members from the collection.
GetOrCreateMemberId(this DynamicMemberCollection thisDynamicMemberCollection, DbConnInfo dbConnApp, DynamicDimensionArgs args, ref string name): `static int`	Gets or creates a member ID, ensuring it doesn't overlap with member IDs already present in OneStream.
CreateMember(this DynamicMemberCollection thisDynamicMemberCollection, DbConnInfo dbConnApp, DynamicDimensionArgs args, string name, string description): `static DynamicMember`	Creates a dynamic member with the specified name and description.

CreateAndAddMember(this DynamicMemberCollection thisDynamicMemberCollection, DbConnInfo dbConnApp, DynamicDimensionArgs args, string name, string description): `static DynamicMember`	Creates a dynamic member with the specified name and description, and adds it to the collection.
CreateMember(this DynamicMemberCollection thisDynamicMemberCollection, DbConnInfo dbConnApp, DynamicDimensionArgs args, string name, string description, int externalMemberId, string externalMemberName, object tag): `static DynamicMember`	Creates a dynamic member with the specified name, description, external ID, external name, and tag.
CreateAndAddMember(this DynamicMemberCollection thisDynamicMemberCollection, DbConnInfo dbConnApp, DynamicDimensionArgs args, string name, string description, int externalMemberId, string externalMemberName, object tag): `static DynamicMember`	Creates and adds a dynamic member with the specified name, description, external ID, external name, and tag.

DynamicRelationship

The `DynamicRelationship` object represents a relationship between a parent DynamicMember and child DynamicMember, including the aggregation weight.

Properties

ParentMember: `DynamicMember`	Gets or sets the parent member.
ChildMember: `DynamicMember`	Gets or sets the child member.
AggregationWeight: `decimal`	Gets or sets the aggregation weight.

Constructors

DynamicRelationship()	Initializes a new instance of the `DynamicRelationship` class.
DynamicRelationship(DynamicMember parentMember, DynamicMember childMember, decimal aggregationWeight)	Initializes a new instance of the `DynamicRelationship` class with the specified parent member, child member, and aggregation weight.

DynamicRelationshipCollection

The DynamicRelationshipCollection class is responsible for managing all relationships in a Dynamic Dimension. It contains a number of utility methods for creating Relationships, which can be added to the actual collection.

Properties

Count: `int`	Gets the number of relationships in the collection.

Constructors

DynamicRelationshipCollection()	Initializes a new instance of the `DynamicRelationshipCollection` class.

Methods

Add(DynamicRelationship relationship)	Adds a dynamic relationship to the collection.
CreateRelationship(SessionInfo si, DynamicMember parentMember, DynamicMember childMember): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member and child member.
CreateRelationship(SessionInfo si, DynamicMember parentMember, DynamicMember childMember, long siblingSortOrder, decimal udAggWeight): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member, child member, sibling sort order, and aggregation weight.

CreateRelationship(SessionInfo si, DynamicMember parentMember, Member childMember): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member and child member.
CreateRelationship(SessionInfo si, DynamicMember parentMember, Member childMember, long siblingSortOrder, decimal udAggWeight): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member, child member, sibling sort order, and aggregation weight.
CreateRelationship(SessionInfo si, Member parentMember, DynamicMember childMember): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member and child member.
CreateRelationship(SessionInfo si, Member parentMember, DynamicMember childMember, long siblingSortOrder, decimal udAggWeight): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member, child member, sibling sort order, and aggregation weight.
CreateAndAddRelationship(SessionInfo si, DynamicMember parentMember, DynamicMember childMember): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member and child member, and adds it to the collection.

CreateAndAddRelationship(SessionInfo si, DynamicMember parentMember, DynamicMember childMember, long siblingSortOrder, decimal udAggWeight): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member, child member, sibling sort order, and aggregation weight, and adds it to the collection.
CreateAndAddRelationship(SessionInfo si, DynamicMember parentMember, Member childMember): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member and child member, and adds it to the collection.
CreateAndAddRelationship(SessionInfo si, DynamicMember parentMember, Member childMember, long siblingSortOrder, decimal udAggWeight): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member, child member, sibling sort order, and aggregation weight, and adds it to the collection.
CreateAndAddRelationship(SessionInfo si, Member parentMember, DynamicMember childMember): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member and child member, and adds it to the collection.
CreateAndAddRelationship(SessionInfo si, Member parentMember, DynamicMember childMember, long siblingSortOrder, decimal udAggWeight): `DynamicRelationship`	Creates a dynamic relationship with the specified parent member, child member, sibling sort order, and aggregation weight, and adds it to the collection.

GetAllRelationships(): `List<DynamicRelationship>`	Gets all dynamic relationships in the collection.
Clear()	Clears all dynamic relationships from the collection.

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*.

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 sample 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. As a cautionary note, 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 View. 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 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

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

Formulas

A formula is a set of calculation instructions to compute values. Formulas are written using Microsoft Visual Basic .NET procedures that use OneStream API function libraries and member script expressions. These combined capabilities provide a powerful programming environment delivering reliable compiled formula definitions.

Formula Structure

Microsoft Visual Basic.NET With OneStream API and Member Scripts

All formulas and business rules run as compiled VB.NET code. In a VB.NET function or subroutine, calls are made to specific API functions which enable the rule writer to interact with the Analytic Engine. Specific API functions are used to process member script expressions, and create calculated values in the analytic data model.

Common Finance APIs That Use Member Scripts

```
api.Data.GetDataCell("A#Cash")
```

This returns a single numeric cell value.

```
api.Data.GetDataCellEx("A#Cash")
```

This returns a single numeric or text cell value.

```
api.Data.SetDataCell("A#Cash", 5.00, False)
```

This saves/writes a single data cell value.

```
api.Data.Calculate("A#Cash = A#Sales * 0.10")
```

This saves multiple data cells as stored values.

Basic Dynamic Cell Calculation

The following return a single cell.

Calculate Current Ratio

```
Return api.Data.GetDataCell("A#15000:O#Top / A#22000:O#Top")
```

Formula Composition:

```
VB.NET     Language Keyword(s)
Return
```

OneStream API Function

```
api.Data.GetDataCell("Member Script”)
```

OneStream Member Script

```
A#15000:O#Top / A#22000:O#Top
```

> **TIP:** When using a nested function (YearPrior(|POVTime|) in a GetDataCell call, you must enclose the function in square brackets []. This is necessary because GetDataCell and the functions use “(“and “)” as indicators of the start/end points in the script. The inner functions must be wrapped in square brackets. GetDataCell("S#[|!ScenarioActual_GRT!|]:T#[YearPrior1(|CVTime|)][Period (|CVTime|)]"):Name("PYR Var")

Basic Stored Calculation

The following calculate and store multiple cells.

Pull Prior Period Retained Earnings

```
api.Data.Calculate("A#28000 = A#28999:T#POVPrior1")
```

Formula Composition

```
OneStream API    Function
api.Data.Calculate("Member Script”)
```

Member Script

```
A#28000=A#28999:T#POVPrior1
```

You can use formula variables in member scripts to significantly improve performance when the same formula is used for multiple members. When using formula variables, the formula text remains the same, so there is no need for continued parsing and evaluation.

Using variables can also improve performance if a member Id is used instead of a member name as the ID can also be used as the value in a formula variable. To use a formula variable in a member script, use a dollar sign $ instead of a pound # sign before the member name, and use the variable name after the dollar sign.

Example 1

```
api.Data.FormulaVariables.SetTextVariable("variableAccount", "8150")
api.Data.Calculate("A#8250=A$variableAccount * 10")
```

Example 2

```
Dim acctMember As Member = api.Members.GetMember(DimType.Account.Id, "8150")
api.Data.FormulaVariables.SetMemberVariable("variableAccount",acctMember)
api.Data.Calculate("A#8250= A$variableAccount * 100")
```

Example 3

```
Dim acctMember As Member = api.Members.GetMember(DimType.Account.Id, "8150")
Dim acctId As Integer = acctMember.MemberId
api.Data.FormulaVariables.SetIntegerVariable("myAccount",acctId)
api.Data.Calculate("A#8250 = A$myAccount * 1000")
```

Applying Formulas

An administrator can create a formula on a Dimension Member or in a Business Rule.

Member Formulas

The preferred and more common approach is to write formulas on Dimension Members. Member formulas are written using the Formula property on individual Scenario, Account, Flow, or User Defined Members in the Dimension Library. The primary reasons for writing formulas on Dimension Members are they provide intuitive formula organization and promote reusability of Dimension and Members with their associated calculation across multiple Cubes. They also enable parallel processing for performance optimizations using advanced multi-threading that executes multiple formulas at the same time. Finally, writing formulas on Dimension Members support drill down from a calculated amount to the amount used as the source for calculation and they support the ability to vary formulas by Scenario and Time. This is useful because changes can be made without affecting the calculation for older data or data in other Scenarios.

Finance Business Rules

Formulas can also be written using one or more Business Rule files. While this typically is not the preferred approach, it provides the ability to put all formulas in one location, and it is similar to the approach used by some older competitive products. The primary reasons for writing formulas in a Finance Business Rule are the formula requires extensive cross-dimensional dependencies and it is not clear on which Member an equivalent Member Formula should be written, or the formula requires complex sequential logic, variable, or conditional statements that affect multiple dependent calculations. Another reason for writing formulas in a Finance Business Rule is when the application requires custom algorithms for Currency Translation, Share, or Intercompany Eliminations.

> **TIP:** Due to the sophisticated built-in translation and consolidation algorithms, most applications only require Member Formulas for the Data Unit Calculation Sequence (DUCS) (i.e., Chart Logic). Custom Business Rules for Translation, Share, and Intercompany Eliminations are not typically needed.

> **NOTE:** Due to the way OneStream stores its data, Decimal should always be used instead of Double or Integer when declaring variables that return a number within a Business Rule or Member Formula.

```
Dim myNumber as Decimal = Api.Data.GetDataCell("A#Sales - A#CostOfSales").CellAmount
```

See Business Rules in Application Tools for more details.

Formula Types

Two types of formulas can be applied to dimensional members or members in a business rule file:

- Dynamic Cell Calculation (Dynamic Calc): An in-memory calculation that runs on demand when a cell containing a dynamic member formula is requested. A Dynamic Calc formula computes a value for a single cell and runs whenever the cell needs to be displayed without storing the result.
- Stored Calculation: A persisted calculation that runs as part of the Data Unit Calculation Sequence (DUCS). With these formulas you can calculate many cells simultaneously, such as data buffer.

Performance Considerations

Dynamic Cell Calculation (Dynamic Calc): Dynamic Cell Calculations enhance the consolidation process because the amount is calculated when requested for display and is not written to the database. For reporting however, performance may be impacted because data is calculated on demand. Dynamic calculations are usually used for ratio or percentage calculations.

Stored Calculation: Consolidation performance is directly impacted by the volume and complexity of stored calculations. Carefully consider each stored calculation since one poorly written rule can cause large amounts of data to be written to the cube, negatively impacting consolidation performance. If you use many member formulas or if data volumes are not considered in member formulas, over 100,000 stored numbers may be generated from just 1,000 initially loaded numbers. The quantity of stored numbers is a critical factor to consider to optimize consolidation performance.

Reference and Aggregation Considerations

Dynamic Cell Calculations: Dynamic cell calculations can reference other dynamic cell calculations and stored calculations but should not be used by stored calculations because they do not naturally aggregate parent members. For example, if a parent member in an Account, Flow, or User Defined dimension has a child member calculated by a dynamic calculation formula, the parent excludes the child member from the aggregated amount. Achieve aggregation by writing another dynamic calculation on the parent member.

Stored Calculations: Stored calculations can reference other stored calculations and parent members aggregate naturally.

> **TIP:** You do not need to write a stored calculation to add or subtract individual members. Instead, create an alternative member hierarchy and use the Aggregation Weight property, set to -1.0 to negate. This aggregated value is dynamic and supports drill-down.

Stored Calculation Evaluation Tools: You can access detailed process logging information. Run **Force Consolidate With Logging** and click **Task Activity** to analyze steps in the consolidation process to identify bottlenecks or errors for performance optimizations. Examples:

- Calculate with Logging
- Translate with Logging
- Consolidate with Logging

Formula Calculation Threshold Monitoring: Calculation time threshold values can be set in the Application Server Configuration File using the **NumSecondsBeforeLoggingSlowFormulas** setting to log formulas exceeding the specified threshold time.

Formula Execution

This section defines the types of calculations and calculation sequences used during different analytic processing routines.

Cube Processing

Calculation: This executes the standard calculation sequence for a single Data Unit. A Data Unit refers to a group of data cells for a specific Cube, Entity, Parent, Consolidation, Scenario, and Time Member. See Data Units in Workflow for more details on Data Units. Except for Dynamic Cell Calculations, all Member Formulas are written to execute as part of a Data Unit's Calculation Sequence.

Translation: This executes a currency translation that occurs when the data for an Entity's local currency needs to be translated to a foreign currency. The translation step executes after the system has run the Data Unit Calculation Sequence for an Entity's local currency. After this is completed, the default translation algorithms use Foreign Exchange (FX) rates to generate and store a corresponding translated Data Unit. Finally, the Data Unit Calculation Sequence on the translated Data Unit to produce the final translated amounts are run.

Consolidation: The Analytic Engine provides pre-built financial intelligence through a statutory Consolidation Dimension that defines a sequence of Data Unit calculations and aggregations which include currency translations, Parent-level adjustments, complex ownership computations, and Intercompany Eliminations. For more details on Consolidation, see Consolidation.

Dynamic Cell Calculation: Dynamic Cell Calculations are a special type of Member Formula for Account, Flow, or User Defined Members. They are used to generate amounts for a Member on the fly (i.e., the results are calculated on-demand and are not stored). Dynamic Cell Calculations are often used for metric accounts (e.g., ratios involving other accounts) and are appropriate when the result of the Dynamic Calculation is not needed as the source number for another Stored Calculation. The use of Dynamic Cell Calculations can result in improved Consolidation performance because they do not generate stored numbers and are typically only executed when a number needs to be displayed, not during the Consolidation process.

Data Units and Formula Execution

OneStream executes formulas at a specific unit of work call a Data Unit. This section details the order and combination of logical processes that execute for a Data Unit. See Using Data Units in Workflow.

Guidelines on Formula Passes: There are 16 Formula Passes and one Dynamic Calculation Formula Type available on each Member. Formulas that do not depend on one another can exist in the same formula pass and will be calculated in parallel using OneStream's advanced parallel formula processing engine.

As a basic guideline, customers should think about organizing formulas by account/collection type across OneStream's available formula passes using the following examples.

Formula Pass 1 - 8: Trial Balance

Formula Pass 5 - 7: Translation logic related to Cash Flow

Formula Pass 8 or 9: Balance Account and CTA account

Formula Pass 9 - 16: Non-trial Balance

All formulas in a pass are processed at the same time, so they cannot have dependencies on one another.

Calculation Sequence of a Single Data Unit: The items below detail the specific list of tasks executed for each Data Unit's calculation process. As an example, the following steps are executed for a single Data Unit when a user selects Calculate for a single Entity, Scenario, and Time period.

Data Unit Calculation Sequences (DUCS)

1. Clear previously calculated data for the Data Unit.
2. Run the Scenario's Member Formula, which is typically used for seeding a Scenario's data from another Scenario or from a prior year.
3. Run reverse translations by calculating Flow Members from other Alternate Currency Input Flow Members. This is part of the built-in ability for an Entity to accept input using multiple currencies.
4. Execute Business Rules (1 and 2). Up to 8 Business Rule files can be attached to each Cube.
5. Run Formula Passes (1 – 4) for the Cube's Account Dimension Members, then Flow Members, and then User Defined Members. The Formula Pass is specified using each Member's Formula Type property in the Dimension Library.
    - Account, Flow, UD1, UD2, … UD8 (Member Formula Execution)
6. Execute Business Rules (3 and 4).
7. Run Formula Passes (5 – 8).
    - Account, Flow, UD1, UD2, … UD8 (Member Formula Execution)

8. Execute Business Rules (5 and 6).
9. Run Formula Passes (9 – 12).
    - Account, Flow, UD1, UD2, … UD8 (Member Formula Execution)
10. Execute Business Rules (7 and 8).
11. Run Formula Passes (13 – 16).
    - Account, Flow, UD1, UD2, … UD8 (Member Formula Execution)

Every time a data cell is written to the database, information with the stored cell about how it was stored (e.g., manually entered, calculated, consolidated, etc.) is included. If a number was calculated and stored as a result of a formula, it will always get cleared regardless of the metadata settings. The AllowInput property specifies whether a cell can be written to, and therefore if a formula stored a number, there would be information if the cell was calculated. If AllowInput is set to True, and a new number is typed over the same cell, it is stored as manually entered instead of a calculated cell.

Formula Execution for Statutory Consolidation Sequence

The consolidation process is run for a hierarchy of Entities. See Consolidation.

Formula Level Controls

In addition to the Cube-level settings described above, VB.NET If statements are often used in a formula to have that formula execute only for certain types of Data Units. The most common usage is to contain checks for Base-Level Entities and local currencies. The following If statement will cause the formula to execute only for Base-Level Entities and the Entity's default currency. This is often used when the results of the calculation in the Base-Level Entity is intended to be consolidated to Parent-Level Entities.

```
If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyForEntity())) Then
End If
```

Writing Formulas

The following section provide information on writing formulas.

Writing Dynamic Cell Calculations

When writing a formula for a Dynamic Calc Account, Flow, or User Defined Member in either a Member Formula or a Business Rule (via FinanceFunctionType.GetDataCell), the goal is to return an amount for a single data cell. In that case, the system knows the full 18 Dimensions of the data cell it needs to display. Therefore, use any of the api methods that refer to a specific Account, Flow, Intercompany, User Defined Dimension, etc.

For example, a user selected a Cube View to view some numbers. The full 18 Dimensions for each cell the Cube View needs to display is determine that it needs to run a custom formula and it initializes the api, so it knows about all 18 Dimensions. The Member Formula then displays an amount on the Cube View.

Dynamic Calc Formula Examples

Return a Constant

The simplest Dynamic Calc formula is to return a constant (i.e., the same number regardless of the intersection). For example, select an Account (or a Flow or User Defined Member) in the Dimension Library, set the Formula Type property to DynamicCalc, and then enter the following line in the Member's Formula property for the Default Scenario Type and Default Time Member. After typing the code, press the Compile button on the Formula Editor's toolbar to make sure the VB.NET syntax is correct, press OK to close the dialogs, and then click Save.

```
Return 123.4
```

If a Cube View is used to display the result of the above calculation for the Dynamic Calc account for any numeric intersection (any Entity, Scenario, Time, UD, etc. for View set to YTD), the Cube View will always display 123.4. If the account's value is displayed for any text intersection (any Entity, Scenario, Time, UD, etc. for View set to Annotation), the Cube View will show an empty cell because the formula returned a numeric value, not a text value. To make the formula work for both numeric and text View Members, use an If statement to check which type of View Member is currently being requested. Then return a text value surrounded by double quotes when the View Member is an annotation type. After saving, run the Cube View using multiple View Members (YTD, Periodic, Annotation, Assumptions, etc.). The corresponding cell will display either of the two constants specified.

```
objViewMember As ViewMember = ViewMember.GetItem(api.Pov.View.MemberId)
If objViewMember.IsAnnotationType Then
   Return "My first OneStream Member Formula"
Else
    Return 123.4
End If
```

Return Types for Dynamic Calc Formulas

As shown in the example above, the return value for a numeric data cell can be a number such as a constant or a decimal variable, and the return type for a text intersection can be a text value such as words within double quotes or a String variable. However, when those types are returned, the underlying engine converts them to a DataCell or a DataCellEx object automatically. Therefore, if a DataCell or DataCellEx object was already contained by calling api.Data.GetDataCell, then it is recommended to return the full object instead of just the number contained within the DataCell object (i.e., which could be accessed using objDataCell.CellAmount). In order to specify status such as whether the cell is NoData, return a DataCell or DataCellEx and use its CellStatus property for those types of settings.

A DataCell object is a wrapper for a DataCellPk object that defines the cell's 18 Dimensional intersection, a decimal accessed using the CellAmount property to store the number, and a CellStatus containing other information about the cell such as NoData and Invalid Status.

A DataCellEx object is a wrapper for a DataCell object and a text property called DataCellAnnotation which is used for setting a string for an Annotation type View Member. It also contains some additional properties for CurrencyId and AccountTypeId filled in and can be ignored when creating a DataCellEx object in a Dynamic Calc Member Formula.

The following example accomplishes the exact same result as the example above, except this uses DataCell and DataCellEx objects to illustrate what to do if a return value containing cell status is needed.

```
Dim objViewMember As ViewMember = ViewMember.GetItem(api.Pov.View.MemberId)
If objViewMember.IsAnnotationType Then
   Dim objDataCellEx As DataCellEx = New DataCellEx()
   objDataCellEx.DataCell.CellStatus = New DataCellStatus(True)
   objDataCellEx.DataCellAnnotation = "My first OneStream Member Formula"
   Return objDataCellEx
Else
   Dim objDataCell As DataCell = New DataCell()
   objDataCell.CellStatus = New DataCellStatus(True)
   objDataCell.CellAmount = 123.5
   Return objDataCell
End If
```

Using Math in Dynamic Calc Formulas

OneStream's api.Data.GetDataCell function supports a powerful script parser that allows math equations to be written and operate on one or more data cells or constant amounts to calculate the values for a new data cell. For example, if a Dynamic Calc account needs to display Cash plus AcctsRec increased by 10%, this one-line Member Formula can do it.

```
Return api.Data.GetDataCell("(A#Cash + A#AcctsRec) * 1.10")
```

Notice the api.Data.GetDataCell function accepts one string Parameter in double quotes which represents a Member Script equation. Each operand of the equation (e.g., A#Cash) takes the unspecified Dimensions using the data cell currently being calculated. Therefore, every operand points to a specific data cell identified by using an 18 Dimension intersection.

If the current data cell being calculated is:

```
"E#CT:C#USD:S#Actual:#2013M1:V#YTD:A#NewAccount:F#None:O#Top:I#None:U1#None, …",
```

then the first operand is:

```
"E#CT:C#USD:S#Actual:#2013M1:V#YTD:A#Cash:F#None:O#Top:I#None:U1#None, …"
```

even though only "A#Cash" was specified.

When writing Dynamic Calc formulas, specify any or all the 18 Dimensions if necessary. (e.g., "A#Cash:U1#AllProducts + A#AcctsRec:U1#None")

Division in Dynamic Calc Formulas

Using Member Script equations in api.Data.GetDataCell as shown above is powerful for Dynamic Calc formulas, but there are occasions when all the math cannot be performed reliably in one line of script. For example, use this formula to divide by a data cell when the data cell being used as the denominator is zero or NoData (i.e., a number was never entered). Since dividing by zero results in infinity and is an invalid operation for computers, the formula needs some extra checking. To illustrate additional concepts, here are three different examples of performing division in a Dynamic Calc formula.

This first example uses the division operator (/) to calculate a data cell from one account divided by a data cell from another account. If the denominator (A#AcctsRec) is zero or NoData, it will automatically return a very large number (e.g., 9999999999999999999.0) as the result. This is because dividing by zero in mathematics results in infinity, and the large number to approximate infinity which allows subsequent functions or math operators that refer to the result to continue to be processed is used.

```
Return api.Data.GetDataCell("A#Cash / A#AcctsRec")
```

Although an extremely large number is the best mathematical approximation for infinity, it is typically not what administrators want to display in their financial system when source numbers are not available. A Divide function that produces a NoData cell if either the numerator or the denominator is NoData is available.

```
Return api.Data.GetDataCell("Divide(A#Cash, A#AcctsRec)")
```

The built-in Divide function is typically used when performing division in Dynamic Calc formulas. However, for completeness and to provide some insight about how to create more complex formulas, the following is an example of how to implement a formula that performs safe division. Notice that the Member Formula is performing division using two DataCell objects to create a resulting DataCell (Return numeratorDataCell / denominatorDataCell). This is a powerful capability that allows any type of math to be performed using any number of DataCell objects.

```
Dim numeratorDataCell As DataCell = api.Data.GetDataCell("A#Cash")
Dim denominatorDataCell As DataCell = api.Data.GetDataCell("A#AcctsRec")
If ((Not numeratorDataCell Is Nothing) And (Not denominatorDataCell Is Nothing)) Then
   If ((Not numeratorDataCell.CellStatus.IsNoData) And _
        (Not denominatorDataCell.CellStatus.IsNoData) And _
        (denominatorDataCell.CellAmount <> 0.0)) Then
     Return numeratorDataCell / denominatorDataCell
   End If
End If
Return Nothing
```

> **TIP:** When using VB.NET, use the underscore character at the end of a line in order to continue a statement on the next line. For example, refer to the underscore after And in the code above. This was done here because the full statement did not fit on one line in this document. However, in the Member Formula editor, the If statement would be kept on one long line.

Days Sales Outstanding

Days Sales Outstanding (DSO) is a common formula that is a required calculation for many applications. Consequently, OneStream has provided a pre-built function to encapsulate the logic required for this function.

```
Return api.Functions.GetDSODataCell("AcctsRec", "Sales")
```

The example below demonstrates a possible customized version of the DSO calculation.

```
Dim numDaysSum as Integer = 0
Dim currTimeId as Integer = api.Pov.Time.MemberPk.MemberId
Dim acctsRec as decimal = api.Data.GetDataCell("A#AcctsRec - A#SaleTax").CellAmount
If (acctsRec > 0.0) Then
    Dim salesSum as Decimal = 0.0
    Dim numPeriodsInYear = api.Time.GetNumPeriodsInYearForSameFrequency(currTimeId)
    For (numPeriodsToSubtract As Integer = 0 To numPeriodsInYear)
        Dim timeId as Integer
        If numPeriodsToSubtract = 0 Then
           timeId = currTimeId
        Else
           timeId = api.Time.AddTimePeriods(-1 * numPeriodsToSubtract, True)
        End If
        Dim timeName As String = api.Time.GetNameFromId(timeId)
        Dim numDaysInTimePeriod As Integer = api.Time.GetNumDaysInTimePeriod(timeId)

        Dim MemberscriptBldr = New MemberscriptBuilder("A#Sales:V#Periodic").SetTime(timeName)
        Dim Memberscript As String = MemberscriptBldr.GetMemberscript()
        Dim salesForTimePeriod as Decimal = api.Data.GetDataCell(Memberscript).CellAmount

        If (salesForTimePeriod + salesSum >= acctsRec) Then
            Dim ratio As Decimal = (acctsRec - salesSum) / salesForTimePeriod
            numDaysSum = numDaysSum + (ratio * numDaysInTimePeriod)
            'We are done
            Exit For
        Else
            numDaysSum = numDaysSum + numDaysInTimePeriod
```

```
            salesSum = salesSum + salesForTimePeriod
        End If
    Next
End If
Return api.Data.CreateDataCellObject(numDaysSum, False, False)
```

Dynamic Simulation of Consolidation: GetEntityAggregationDataCell

Use a function called GetEntityAggregationDataCell for pseudo/approximate consolidation of a data cell with a Cube View when requested for display. The intention is not to produce a value that would tie to a formally consolidated number if custom Business Rules for ownership or translation are in play for such a financial model, but this on-the-fly dynamic value is presented to the user instantly as a convenience, typically during data entry.

This function employs standard currency translation using the Entity in the cell's POV's local currency as the source and a parent Entity's target and standard Percent Consolidation on the Relationship Properties to calculate Share. It does not take intercompany elimination into account unless that value was already consolidated and stored.

```
api.Functions.GetEntityAggregationDataCell(memberScript as string,
 Optional useStoredAmountsWhereCalcStatusIsOK as Boolean, Optional
 fxRateTypeRevenueExpOverride as String, Optional
fxRuleTypeRevenueExpOverride as FxRuleType, Optional
 fxRateTypeAssetLiabOverride as String, Optional
fxRuleTypeAssetLiabOverride as FxRuleType)
```

When useStoredAmountsWhereCalcStatusIsOK is set to True (the default setting), the algorithm checks if CalcStatus is OK for the intersection and uses numbers that are already calculated, translated, ownership share calculated, eliminated, consolidated and stored. This setting is also useful when displaying multiple translated currencies for consolidated parent Entities dynamically. If the parent Entity's Local amount is OK and fully consolidated, the only dynamic calculation would be the last translation step.

The last four optional settings let you specify alternate named FX Rate Types (e.g. "AverageRate") and FX Rule Types (i.e. FxRuleType.Direct or FxRuleType.Periodic) to perform what-if simulations.

If any of these four are specified, useStoredAmountsWhereCalcStatusIsOK is ignored and is treated as False because consolidated amounts would not have been stored using the alternate FX rates.

Examples when used with a Dynamic Calc member (e.g. UD8):

```
Return api.Functions.GetEntityAggregationDataCell("A#NetIncome:S#Budget:UD8#None")
Return api.Functions.GetEntityAggregationDataCell("UD8#None", True, "AverageRate",
FxRuleType.Periodic)
Return api.Functions.GetEntityAggregationDataCell("UD8#None", True, "AverageRate",
FxRuleType.Periodic,
        "HistoricalRate", FxRuleType.Direct)
```

Example use of GetEntityAggregationDataCell to enter inputs and instantly see aggregated results upon saving a Form:

Create a UD8 Member named EntityAggregation and set its Formula Type to Dynamic Calc. Use this as the UD8 member's Formula:

```
Return api.Functions.GetEntityAggregationDataCell("UD8#None")
```

Create a Cube View

Set Cube View POV tab to have Scenario set to Budget or whatever is desired on the Form and UD8 Member of EntityAggregation.

Define rows that display a hierarchy of Entities.

Define columns that display a few key Accounts. Include a base-level Account that supports input for both UD8#EntityAggregation and for input. E.g. A#Sales, A#Sales:UD8#None:Name("Sales Input")

Change Cube View settings for General Settings / Common with Can Modify Data and Can Calculate set to True.

Associate this Cube View with a Form Template which is assigned to a Workflow Profile.

The user enters a number on the Form in the Account that accepts input and clicks Save. The user will see the dynamically aggregated Entity results without having to run a Consolidation.

Dynamic Calc Member Formula

This formula returns the name and description of the Entity's UD1 Default Member. Other UD8 formulas can be created for the Entity's UD2 Default, etc. After adding the UD8 Member, the Entity's Default UD1 Member name can be displayed using XFGetCell or in a Cube View by accessing the cell: E#MyEntity:V#Annotation:U8#EntityDefaultUD2Name.

This assumes that the UD8 Member with the formula is named EntityDefaultUD2Name. If XFGetCell is being used in Excel, use None or any valid Member for all the other Dimensions.

```
'Display the Member name of the entity's DefaultUD1 Member using the Annotation View Members.
If api.View.IsAnnotationType(api.Pov.View.MemberId) Then
   Dim text As String = String.Empty
   Dim udId As Integer = api.Entity.GetDefaultUDMemberId(api.Pov.Entity.MemberId, DimType.UD1.Id)
   If udId <> DimConstants.Unknown Then
Dim udMember As Member = api.Members.GetMember(DimType.UD1.Id, udId)
```

```
If Not udMember Is Nothing Then
text = udMember.NameAndDescription
End If
    End If
    Return text
End If
'If this is a numeric View Member (e.g., Periodic, YTD), display the number from the U8#None
Member.
Return api.Data.GetDataCell("U8#None")
```

Annotations on Dynamic Calc Members

To support the ability to calculate text or pull it from an external system to display in a Cube View, DynamicCalc or DynamicCalcTextInput formulas can be used. If the user wants the formula to calculate the annotations, use a DynamicCalc formula to display text such as Pass and Fail.

The DynamicCalcTextInput Formula Type works the same as a DynamicCalc formula, but it allows users to input annotations on Cube View cells without having to use the Data Attachment Dialog. When this formula is used, the user can make annotations on Dynamic Calc Members following the same method as a non-calculated Member.

Writing Stored Calculations

When writing a Member Formula or a Business Rule for a Stored Calculation, the new calculated numbers are being determined to store for that Cube, Entity, Parent, Cons, Scenario, and Time combination (i.e., a Data Unit).

Return is never seen in a Member Formula for Formula Pass. Instead of being returned, many numbers are being calculated and stored. When running a Calculation, Translation, or Consolidation, the Member Formula calls for an entire Data Unit. It does not tell with which Account, Flow, or User Defined the numbers are being saved, this is the responsibility of the user. Initially, this may be confusing because Member Formulas are often written in an account's Formula property, and administrators believe it will only allow that specific Member Formula to write to that specific account. However, putting a Member Formula in an account's Formula property is only for organizational purposes. When it calls that formula, it is currently calculating a Data Unit and it will initialize the api with only the Data Unit Dimensions.

Basic Stored Formula Examples

The formula examples in this section demonstrate how to calculate basic stored values driven by formulas consisting of OneStream Member Script expressions.

Copy Data from another Account

The following formula would be implemented as a Member Formula on the Sales1 account. It is executed as part of the DUCS during the Formula Pass that was specified in the account's Formula Type setting.

```
api.Data.Calculate("A#Sales1 = A#Sales2”)
```

Stored Formula passes use Data Buffer math, not the Data Cell math that occurs for the single cell Dynamic Calc formulas. Stored Formulas are multi-Dimensional. For example, the formula is executed for an entire Data Unit (e.g., Location1 Entity, USD Consolidation Member, Actual Scenario, January 2013 time period). That Data Unit is a portion of a Cube where the UD1 Dimension could contain 1,000 products to keep track of sales by product. Therefore, the data for the Sales2 account could contain a separate number for every Product (i.e., UD1), or if the Location1 Entity only sells some of the products, there might be 200 numbers for Sales2 and the other 800 products for Sales2 are NoData. That set of 200 numbers is called a Data Buffer. Data Buffers can get much larger and more complicated when multiple Dimensions are used for detailed analyses. However, since the same concepts still apply, it is easier to think about a smaller set of Dimensions as in this example.

The formula “A#Sales1 = A#Sales2” is equivalent to saying, “Take the 200 numbers stored in the Sales2 Data Buffer and copy them to a new Data Buffer, but change the account to Sales1, and then store the new Sales1 Data Buffer in the database.” That one-line formula calculated and stored an additional 200 numbers that did not exist before the formula was executed.

The formula below reads the Data Buffer for the Sales2 account (200 numbers) and then adds 50.0 to each of those 200 numbers to create a new Data Buffer that also contains 200 numbers. The account for each of the 200 numbers in the new Data Buffer is changed to Sales1 and it is then stored in the database.

```
api.Data.Calculate("A#Sales1 = A#Sales2 + 50.0”)
```

The newly modified formula below uses three accounts. That formula reads the Data Buffer for the Sales2 account (200 numbers) and then reads the Data Buffer for the Sales3 account. For example, the Sales3 account contains 100 numbers broken out by product in UD1, and 25 of those 100 numbers use the same UD1 Members as some of the numbers from Sales2. The other 75 Sales3 numbers are for other products not used by the Sales2 account. OneStream automatically combines the numbers from Sales2 and Sales3 and adds the Sales2 and Sales3 numbers that share a common intersection and also adds the additional non-common intersections. The result is a new Data Buffer containing 275 numbers stored in the database for the Sales1 account.

```
api.Data.Calculate("A#Sales1 = A#Sales2 + A#Sales3”)
```

Copy Data from Another Scenario

The following formula would be implemented as a Member Formula on the Forecast Scenario. It copies all the data from the Actual Scenario and stores the results in the Forecast Scenario. The If statement in this example causes the data to be copied only for Base-Level Entities and each Entity’s local currency. This is because the example application wants the numbers for Parent Entities and foreign currencies to be determined using OneStream’s Consolidation and Translation algorithms, not by copying directly from another Scenario. Limit the formula to only copy data for certain Dimension Members by adding specific Members in the formula’s Member Script. For example, if the Forecast Scenario needs to copy only the Import Members, add O#Import to both sides of the equation. If the Forecast Scenario needs to start with the sum of the Actual Scenario’s Import plus Forms plus AdjInput data and copy that into the Forecast Scenario’s O#Import Member, then use “S#Forecast:O#Import = S#Actual:O#Top”. When writing Stored Formulas, any Data Unit Dimension not explicitly specified uses the Dimension Member for the Data Unit currently being calculated. For any Account, Flow, Origin, IC, or User Defined Dimensions not explicitly specified, OneStream will use #All which is the syntax that represents all existing data for that Dimension.

```
If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyForEntity()))
Then
    api.Data.Calculate("S#Forecast = S#Actual")
End If
```

To copy data from another Cube or Scenario that uses different Dimensionality, an example of the Scenario formula would be as follows:

```
'Convert dimensionality
Dim destinationInfo As ExpressionDestinationInfo = api.Data.GetExpressionDestinationInfo("")
Dim sourceDataBuffer As DataBuffer = api.Data.GetDataBuffer
(DataApiScriptMethodType.Calculate,
"Cb#AnotherCube:S#AnotherScenario",destinationInfo)
Dim convertedDataBuffer As DataBuffer = api.Data.ConvertDataBufferExtendedMembers
("AnotherCube",
"AnotherScenario", sourceDataBuffer)
api.Data.SetDataBuffer(convertedDataBuffer, destinationInfo)
```

To drill down on this formula, use the following example in the Scenario's Formula for Calculation Drill Down setting:

```
If api.Pov.Cube.Name.XFEqualsIgnoreCase("TheDestCube") Then
    Dim result As New DrillDownFormulaResult()
    result.Explanation = "Pseudo-formula:
Cb#TheDestCube:S#TheDestScenario=Cb#AnotherCube:S#AnotherScenario”
    result.SourceDataCells.Add("Cb#AnotherCube:S#AnotherScenario")
    Return result
End If
Return Nothing
```

Out-Of-Balance

The following formula would be implemented as a Member Formula on the Balance account. It stores the difference of two other accounts.

```
If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyForEntity())) Then
    api.Data.Calculate("A#Balance = A#2899 - A#5999")
End If
```

CTA Account Formula Examples

This is essentially the same formula as Out-Of-Balance, but only runs on translated Data Units.

```
If ((Not api.Entity.HasChildren()) And (api.Cons.IsForeignCurrencyForEntity())) Then
    api.Data.Calculate("A#CTA = A#2899 - A#5999")
End if
```

Reading a Specific Data Cell in a Stored Formula

Stored Formulas are executed for an entire Data Unit, so unlike Dynamic Calc formulas, there is no context about a specific View, Account, Flow, Origin, IC, or User Defined Member within the Data Unit. Therefore, to read the value for a Data Cell using api.Data.GetDataCell, a Member needs to be explicitly specified for all non-Data Unit Dimensions. This is different than Dynamic Calc Member Formulas where the default setting for every Dimension Member comes from the Data Cell currently being displayed.

Therefore, the following is incorrect when trying to read a specific Data Cell inside a Stored Formula:

```
Dim objDataCell As DataCell = api.Data.GetDataCell("A#Cash")
```

Instead, all non-Data Unit Dimensions need to be specified. If one or more of those Dimensions needs to be based on the other data stored in the Data Unit, then use Eval (for details on Eval see Advanced Stored Formulas using Eval below). Otherwise, the formula will look like this:

```
Dim objDataCell As DataCell = api.Data.GetDataCell("V#YTD:A#Cash:F#None:O#Import:I#None
    :U1#None:U2#None:U3#None:U4#None:U5#None:U6#None:U7#None:U8#None")
```

The syntax above is accurate but creating that long string for many Dimensions is tedious and error prone especially when using functions to determine what the Member names should be, and then concatenating multiple strings. Instead it is recommended to use the MemberscriptBuilder class when creating Member Scripts.

```
Dim MemberscriptBldr = New MemberscriptBuilder("V#YTD:A#Cash")
MemberscriptBldr.SetFlow("None").SetOrigin("Import").SetIC("None").SetAllUDsToNone()
Dim Memberscript As String = MemberscriptBldr.GetMemberscript()
Dim objDataCell As DataCell = api.Data.GetDataCell(Memberscript)
```

Advanced Stored Formulas using Eval

As described in the examples above, OneStream's Data Buffer math is extremely powerful and can process hundreds or thousands of numbers with just one simple equation. Without Data Buffer math or an equivalent scripting capability, a large multi-Dimensional financial application would not be feasible because every intersection would need to be considered separately. There is a consequence when processing data using Data Buffers instead of individual Data Cells. Additional capabilities are needed when wanting to perform math differently based on the individual data cell amounts.

Fortunately, OneStream accommodates that pattern using Eval. When implementing api.Data.Calculate functions, Eval has an advanced capability that provides the ability to get at the individual Data Cells in any Data Unit created while processing an api.Data.Calculate script. It even allows Eval() to be wrapped around a subset of the formula's math in order to evaluate the Data Buffer that was just created by running that math.

As an example for Eval, start with this formula:

```
api.Data.Calculate("A#Sales1 = A#Sales2″)
```

The Sales2 numbers need to be copied to Sales1 for “green” Products. In this fictitious example, there is a special tax situation for green products and the sales numbers for those products need to be isolated into the special Sales1 account. The application uses the UD1 Member’s Text1 property to keep track of which products are green.

The first thought might be to do something like the following (incorrect):

```
Dim ud1Id As Integer = api.Pov.UD1.MemberId
Dim text1 As String = api.UD1.Text(ud1Id, 1)
If (text1.Equals("green", StringComparison.InvariantCultureIgnoreCase)) Then
    api.Data.Calculate("A#Sales1 = A#Sales2")
End If
```

However, this would not work because Stored Formulas are executed for an entire Data Unit. A Data Unit represents all data for a Cube, Scenario, Entity, Parent, Cons, and Time Member. Since there is no single product (i.e., UD1 Member) for the Data Unit currently being calculated, the first line above does not make sense. A Data Unit cannot be asked what the UD1 MemberId is because a Data Unit has data for multiple UD1 Members (200 different products in the Sales2 example).

The solution is using Eval to evaluate the individual Data Cells in a Data Buffer. Put the Eval keyword around any portion of the api.Data.Calculate function including math statements. After OneStream reads or calculates the DataBuffer defined within the Eval statement, it executes the Eval function to give the opportunity to filter the list of Data Cells in the Data Buffer, or to completely change the list of Data Cells in the Data Buffer. After the Eval function is completed, OneStream uses the modified Data Buffer to perform the remaining part of the api.Data.Calculate function.

The example formula needs to be modified by adding the Eval keyword around the A#Sales2 Data Buffer, a helper function, typically the name OnEvalDataBuffer, needs to be implemented allowing an inspection, filter, and/or change to the Data Cells in the Data Buffer. The helper function Loops over each of the Sales2 Data Cells (200 in this example). If the Data Cell's UD1 Text1 setting says green, add that Data Cell to a new list of result cells. Otherwise, ignore the Data Cell causing it to be skipped. The result is a new modified Data Buffer containing only the Data Cells for green products (i.e., fewer than 200 Data Cells).

```
api.Data.Calculate("A#Sales1 = Eval(A#Sales2)", AddressOf OnEvalDataBuffer)

Private Sub OnEvalDataBuffer(ByVal api As FinanceRulesApi, ByVal evalName As String, _ByVal
eventArgs As EvalDataBufferEventArgs)
    'Filter to list of Sales2 source numbers to only include numbers
    'for "green" Products using each dataCell's UD1 Text1 setting.
    'The final list of resultCells is what will be assigned to Sales1 by api.Data.Calculate.
    Dim resultCells As New Dictionary(Of DataBufferCellPk, DataBufferCell)
    For Each sourceCell As DataBufferCell
In      eventArgs.DataBuffer1.DataBufferCells.Values
        If (Not sourceCell.CellStatus.IsNoData) Then
            Dim ud1Id As Integer = sourceCell.DataBufferCellPk.UD1Id
            Dim text1 As String = api.UD1.Text(ud1Id, 1)
            If (text1.Equals("green", StringComparison.InvariantCultureIgnoreCase)) Then
            'Add this dataCell to the new list.
            resultCells(sourceCell.DataBufferCellPk) = sourceCell
            End If
        End If
    Next
    'Assign the new list of DataCells to the result.
    eventArgs.DataBufferResult.DataBufferCells = resultCells
End Sub
```

NOTE: If using Eval for NoData and ZeroCells, refer to the Remove Functions in Formulas section for alternative performance enhancing solutions.

Advanced Stored Formulas using GetDataBuffer and SetDataBuffer

Most of the advanced stored formulas that need to process multiple Data Cells should use Eval. Eval allows the user to get at the individual Data Cells in any Data Unit while processing the Member Script in an api.Data.Calculate function. However, in some rare occasions, an appropriate Member Script may not be defined for the api.Data.Calculate function because multiple Data Cells that seem completely unrelated to each other are being processed and none of the Dimension Members are constant.

For those rare situations, use the GetDataBuffer and SetDataBuffer functions directly. GetDataBuffer and SetDataBuffer are more fundamental then Eval. They are part of the internal implementation of the Eval functionality. They allow the user to read some numbers using a Member Script, process or modify each cell in the result, and then save the changes.

In the following example, the UD2 and UD3 Dimensions are being used to analyze data based on each UD1 Member's default settings for UD2 and UD3. All data is initially loaded to the U2#Input:U3#Input Members, but that loaded data needs to be copied to the U2#DefaultUD1:U3#DefaultUD1 Members. GetDataBuffer needs to be used in order to read the loaded data because the destination UD2 and UD3 Members can be different for every Data Cell based on its UD1 Member's settings. Loop over each Data Cell and use the UD1 Member to get its default UD2 and default UD3 settings. Then, change the UD2 and UD3 Member IDs for the Data Cell in the Data Buffer. Finally, after Looping, call SetDataBuffer to save the new numbers.

When using api.Data.Calculate functions with or without Eval, it is important to know to which Member a formula is being attached. For example, if the formula starts with api.Data.Calculate ("A#Sales1 = …"), put the formula in the Sales1 account Member's Formula setting. However, the formula in this example is not writing to a specific Member. Every Data Cell being saved is possibly written to a different UD2 and UD3 Member.

Technically speaking, the formula can be put in any Member's Formula property even a seemingly unrelated Member. If the Formula Pass is set correctly, the formula executes before any other dependent formulas. Assigning stored formulas to Members is for organizational purposes only. The Member does not restrict what the formula can do. However, if a formula is attached to an unrelated Member, it will make the application difficult to maintain and understand. Therefore, decide to attach a formula like this to the Scenario's Member which means this formula needs to be processed before most other formula passes. If there are numerous Scenarios, this formula should be put in a Business Rule file, and the Business Rule file should be added to the Cube(s). This can be done under
Application Tab|Cube|Cubes.
See the completed formula using GetDataBuffer and SetDataBuffer below:

```
'Copy all "U2#Input:U3#Input" numbers for this dataUnit to the corresponding UD1 default
Members for UD2 and UD3.
Dim destinationInfo As ExpressionDestinationInfo = api.Data.GetExpressionDestinationInfo("")
Dim sourceDataBuffer As DataBuffer = api.Data.GetDataBuffer
(DataApiScriptMethodType.Calculate, _"U2#Input:U3#Input", destinationInfo)
If Not sourceDataBuffer Is Nothing Then
    Dim resultDataBuffer As DataBuffer = New DataBuffer()
    For Each cell As DataBufferCell In sourceDataBuffer.DataBufferCells.Values
        If (Not cell.CellStatus.IsNoData) Then
            Dim ud1Id As Integer = cell.DataBufferCellPk.UD1Id
            cell.DataBufferCellPk.UD2Id = api.UD1.GetDefaultUDMemberId(ud1Id,
DimType.UD2.Id)
            cell.DataBufferCellPk.UD3Id = api.UD1.GetDefaultUDMemberId(ud1Id,
DimType.UD3.Id)
            resultDataBuffer.SetCell(api.DbConnApp.SI, cell)
        End If
    Next
    api.Data.SetDataBuffer(resultDataBuffer, destinationInfo)
End If
```

Comparing Two DataBuffers Using Eval2

Eval2 is the same as Eval except two Members Scripts are specified to define two DataBuffers. When finished, the OnEvalDataBuffer function can compare all the numbers in the two DataBuffers. The example provided evaluates two separate Flow Members to see if they contain the same value.

```
'Use Eval2 to compare the numbers in 2 DataBuffers and store a value in F#USDOverride_Check
for each pair of numbers that don't match.
If ((api.Cons.IsLocalCurrencyForEntity()) And (Not api.Entity.HasChildren())) Then
    api.Data.Calculate("V#YTD:F#USDOverride_Check:O#Forms = Eval2(V#YTD:F#Local_Change_
Validation:O#Top,V#YTD:F#USDOverride_Change_Validation:O#Top)", AddressOf OnEvalDataBuffer)
End If

Private Sub OnEvalDataBuffer(ByVal api As FinanceRulesApi, ByVal evalName As String, ByVal
eventArgs As EvalDataBufferEventArgs)

'This function compares the numbers in 2 DataBuffers and returns a new DataBuffer that has a
value for each pair of numbers that don't match.
    eventArgs.DataBufferResult.DataBufferCells.Clear()
        If Not eventArgs.DataBuffer1 Is Nothing And Not eventArgs.DataBuffer2
        Is Nothing Then
'For each cell in DataBuffer1, try to find a number for the same intersection in
DataBuffer2.
 For Each cell1 As DataBufferCell In eventArgs.DataBuffer1.DataBufferCells.Values
            If (Not cell1.CellStatus.IsNoData) Then
            Dim cell2 As DataBufferCell = eventArgs.DataBuffer2.GetCell
(api.SI,cell1.DataBufferCellPk)
            If Not cell2 Is Nothing Then
                If (cell1.CellAmount <> cell2.CellAmount) Then
                'Since the numbers don't match, add a cell to the result DataBuffer.
                 Dim resultCell As New DataBufferCell(cell1)
                resultCell.CellAmount = 1.0
                eventArgs.DataBufferResult.SetCell(api.SI, resultCell, False)
                End If
            Else
            'A number exists in the 1st DataBuffer, but it doesn't exist in the
2ndvDataBuffer.
            'Therefore, add a cell to the result DataBuffer.
                Dim resultCell As New DataBufferCell(cell1)
                resultCell.CellAmount = 1.0
                eventArgs.DataBufferResult.SetCell(api.SI, resultCell, False)
            End If
        End If
Next
'Now, for each cell in DataBuffer2, try to find a number for the same intersection in
DataBuffer1 (the opposite of the loop above).
'Create a cell in the result DataBuffer for each DataBuffer2 cell that doesn't exist in
DataBuffer1.
For Each cell2 As DataBufferCell In eventArgs.DataBuffer2.DataBufferCells.Values
        If (Not cell2.CellStatus.IsNoData) Then
            Dim cell1 As DataBufferCell = eventArgs.DataBuffer1.GetCell
(api.SI,cell2.DataBufferCellPk)
                If cell1 is Nothing Then
                'A number exists in the 2nd DataBuffer, but it doesn't exist in the 1st
DataBuffer.
                'Therefore, add a cell to the result DataBuffer.
                    Dim resultCell As New DataBufferCell(cell2)
                    resultCell.CellAmount = 1.0
                    eventArgs.DataBufferResult.SetCell(api.SI, resultCell, False)
                End If
            End If
        Next
```

```
    End If
End Sub
```

Referencing a Business Rule from a Member Formula or Business Rule

Finance Business Rules from Member Formulas or other Business Rules can be called. This is helpful when the same code must be copied to multiple Member Formulas and instead of using the same complicated code, a Public Function with two lines of code written in a Business Rule can be called.

First, create a new Finance Business Rule (in this example, a Finance Business Rule called SharedFinanceFunctions was created) and then set the Contains Global Functions for Formulas property to True. If the Business Rule is only being used to hold Shared Functions, delete most of the content in the Main function. However, a Main function is still needed even if it is empty.

Next, create a Public Function or Sub in the Business Rule. See below for an example. If any edits to the Business Rule impact Calculation Status, assign the Shared Business Rule to the Cube under
Application Tab|Cube|Cubes. This is recommended.

Use the Business Rule in a Member Formula by creating an instance of the Business Rule and assigning it to a variable. Then, any of the Business Rule's Public Functions or Sub can be called.

Sample Member Formula Code:

Dim sharedFinanceBR As New
OneStream.BusinessRule.Finance.SharedFinanceFunctions.MainClass
Dim myResult As String = sharedFinanceBR.Test(si, api, args)

Sample Business Rule:

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

Namespace OneStream.BusinessRule.Finance.SharedFinanceFunctions
    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
                Return Nothing
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        Public Function Test(ByVal si As SessionInfo, ByVal api As FinanceRulesApi, ByVal
args As FinanceRulesArgs) As String
            Try
                Return "This is the result of my Test function!"
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
    End Class
End Namespace
```

Conditionally Apply Formulas to Entities and Consolidation Members

Conditionally apply formulas to run only when needed. For example, calculate headcount only at Base Entities and Local currency and the consolidation engine will do the rest.

```
If (api.cons.IsLocalCurrencyForEntity And (Not api.entity.HasChildren)) Then
    api.data.calculate("A#10999=A#25999-A#25986")
End If
```

Focus Consolidation to Calculate Only When Needed

There are up to seven calculation operations per Entity in the Consolidation process. Add Conditional Statements to formulas to limit which Consolidation calculation processes will run for a particular formula.

Base Entities Only

If Not api.entity.HasChildren Then

Local Currency Only

If Api.Cons.IsLocalCurrencyforEntity Then

Translated Currency Only

```
If api.Cons.IsForeignCurrencyForEntity Then
```

Parent-Child Relationships

If api.Cons.IsForeignCurrencyForEntity Then returns True if the current calculated Consolidation Member also depends on the Parent Entity (i.e. OwnerPreAdj, Share, Elimination, OwnerPostAdj, Top). If there are two different Parent Entities for the same Entity, then there are two different sets of numbers stored for those Consolidation Members.

At Specific Level of Consolidation

If (Not api.Entity.HasChildren()) And (api.Pov.Cons.Name.XFEqualsIgnoreCase("Elimination")) Then would run if the Entity is a parent and also if the member of the Consolidation dimension being processed in the Data Unit is Elimination. Note that this is the preferred function to use rather than the formerly supported api.Pov.Cons.ScriptName.

Formulas for Calculation Drill Down

To drill down on calculated Members, a formula must be entered in the Formula for Calculation Drill Down property. This allows drilling to occur on calculated Account, Flow, User Defined or Scenario Members.

Drill down can occur on data cells copied from one Scenario to another via formula or Data Management Sequence. Before displaying the drill results, every cell's Formula for Calculation DrillDown Scenario Property is executed. The result determines whether the Scenario Member will appear as drillable or not. Therefore, use If Statements in the formula to narrow in on the cell's Storage Type and/or the POV Members associated with the data copy, so cells do not appear drillable when they are not. The example below copies data from the Actual Scenario Type to Budget:

```
Dim result As New DrillDownFormulaResult()
If args.DrillDownArgs.RequestedDataCell.CellStatus.StorageType =
DataCellStorageType.Calculation Then
    'Use this to drill down to data that was copied using a Scenario Formula.
    result.Explanation = "Formula Definition: Actual = Budget"
    result.SourceDataCells.Add("Cb#Houston:E#Houston:S#Actual")
Else If args.DrillDownArgs.RequestedDataCell.CellStatus.StorageType =
DataCellStorageType.Input Then
    'Use this to drill down to data that was copied using Data Management.
    'result.Explanation = "Data Management Defintion: Actual = Budget"
    'result.SourceDataCells.Add("Cb#Houston:E#Houston:S#Actual")
End If
Return Result
```

Global Entity for Driver Storage

A Global Entity can be created to store information that is unrelated to the company's data such as switches, drivers, or values. To do this within the application, create the Entity and set the Is Consolidated property to False. The Consolidation process will then skip this Entity allowing companies to use this for grouping purposes only.

Avoiding Data Explosion

In some cases, many data intersections unintentionally explode in the Cube with unintended values. This can happen with multi-dimensional data structures.

For example, an application has one Entity, four Accounts, 100 Products (UD1) and 100 Customers (UD2). The user has entered three numbers in the system.

```
(A#Sales:U1#Prod1:U2#Cust1=22.0,   A#Sales:U1#Prod5:U2#Cust1=33.0,
A#Sales:U1#Prod7:U2#Cust100=55.0).
```

The following cases explain what happens when writing the following Formulas.

Case 1: No Explosion

Use the following statement to set the value of a data intersection equal to another or a constant value.

```
api.Data.Calculate("A#Cash1 = A#RestrictedCash")
```

Cash1 Account will equal what was in the Restricted Cash Account. The only UD1 and UD2 Members populated would be None Members because it is not typical to delineate Cash by Product, or Customer.

Case 2: No Explosion

```
api.Data.Calculate("A#Profit:U2#Cust1 = A#Sales * 1.05")
```

Two of the three numbers are multiplied by 1.05 and copied from the Sales Account to the Profit Account which are only the source Sales numbers for Customer 1.

Case 3: Some Explosion

```
api.Data.Calculate("A#Profit = A#Sales:U2#Cust1")
```

The user specified a Customer for source data but did not specify which customer to use when writing to the destination. This means answers are written to every base-level member of the Customer dimension, resulting in some data explosion. The Profit account has a number for every Customer (UD2) using the same Product (UD1) member as the source numbers. Profit account will have 300 numbers, but because there were three different source products, there will be some data explosion.

Not specifying a Member for a Dimension is the same as specifying all for a Member. If All is specified on the left only, data explosion occurs. Some level of data explosion occurs if:

- The right side of an equation identifies a specific member or amount.
- The left side of an equation specifies All for one or more dimensions.

Case 4: Large Scale Explosion When Setting to a Constant

```
api.Data.Calculate("A#Profit = 2.0")
```

Profit will have 1000 numbers stored that are all Products by all Customers, causing large scale explosion. However, this example is overly simple. If all Dimensions are used, trillions of numbers may be stored. A constant is the same as not specifying All for every Dimension on the right side of the equation.

Writing Formulas to Avoid Data Explosion

If a member is specified for a dimension on the right side of the equation, explicitly specify a value for that Dimension on the left side.

The examples below are the same as Cases 3 and 4 from above, but this time the formulas are written to avoid data explosion.

Case 3: Specify a target on the left side of the equation

api.Data.Calculate("A#Profit = A#Sales:U2#Cust1") would be better written as api.Data.Calculate ("A#Profit:U2#Cust1 = A#Sales:U2#Cust1") if that is what is intended. If the user intended on copying Cust1's Sales figures to every Customer intersection under the Profit Account, the formula would need to be written as follows:

```
api.Data.Calculate("A#Profit:U2#All = A#Sales:U2#Cust1"
```

Case 4:

api.Data.Calculate("A#Profit = 2.0") will fail since the constant of 2.0 implies All Members from each Dimension. In order for this Formula to execute, it needs to be api.Data.Calculate ("A#Profit:F#All:O#All:I#All:U1#All:U2#All:U3#All:U4#All:U5#All:U6#All:U7#All:U8# All = 2.0"), although the user may choose to do otherwise.

Avoiding Data Explosion in Stored Formulas

When writing stored calculations, the Member Script equations in api.Data.Calculate end up reading or calculating one or more Data Buffers for the purpose of saving a resulting Data Buffer. As described earlier, thousands of Data Cells could be processed using a seemingly simple one-line formula.

Data Explosion can occur when a formula is inadvertently written to read or calculate a Data Buffer and then copy all Data Cells in that Data Buffer multiple times to every base-level Member of a Dimension using the resulting Data Buffer (often causing hundreds of thousands of new numbers to be saved). We only apply calculation to intersections where data exists. Fortunately, OneStream protects the user from writing formulas that could result in data explosion, but the concepts are important to understand because it is possible to circumvent those protections.

Consider the formula from an earlier example:

```
api.Data.Calculate("A#Sales1 = A#Sales2 + A#Sales3”)
```

Now, the formula is changed, so it only copies the sales data for specific customers with UD2 being the Customer Dimension.

```
api.Data.Calculate("A#Sales1:U2#None = A#Sales2:U2#CustomerX + A#Sales3:U2#CustomerY”)
```

The above formula will not result in data explosion because there is the same level of detail (i.e., the same Dimensions) specified in the destination as in every source operand. The example now reads the Sales2 data for CustomerX, adds it to the Sales3 data for CustomerY, and saves the results in the Sales1 account and the UD2 None Member.

The following formula is written to cause Data Explosion:

```
api.Data.Calculate("A#Sales1 = A#Sales2:U2#CustomerX + A#Sales3:U2#CustomerY”)
```

The UD2 Dimension is specified for the source operands which is the right-hand side of the equals sign, but UD2 is not specified for the destination which is the left-hand side of the equals sign.

When processing this formula, OneStream will read the two Data Buffers and add them together correctly as before. However, it then needs to assign the combined Data Buffer to the Sales1 Account. However, the system cannot use U2#CustomerX or U2#CustomerY because Data Cells were created by adding those together and it cannot arbitrarily choose one over the other. The system could also have defaulted to use the U2#None Member, but history suggests that this type of rule is more often written in error, and the customer did not intend the results to be stored in the U2#None Member. From a maintenance perspective, OneStream feels that it is better to explicitly specify U2#None if that is the intended destination Member.

If something like this were to happen, OneStream will provide an error message notifying the user that data explosion will occur when trying to execute the above formula. Otherwise, the formula will copy the source Data Buffer to every base-level UD2 Member because #All is the default setting for each unspecified Member.

To circumvent the error message and force data explosion (please do not do this), explicitly specify U2#All in the destination as shown below. This should be avoided and #All should never be used in Member Scripts for stored formulas. However, the capability is provided for extremely rare circumstances where that functionality was relied upon using an older product. In this case, the consultant carefully analyzed the quantity of data and metadata settings to ensure the data explosion resulted in a manageable number of Data Cells.

> **CAUTION:** This causes Data Explosion! Do not ever use #All explicitly in stored Member Formulas.

```
api.Data.Calculate("A#Sales1:U2#All = A#Sales2:U2#CustomerX + A#Sales3:U2#CustomerY")
```

Key Functions

The list below contains the most commonly used functions; however, this is not the complete list of all available functions. See the *API Overview Guide*.

Account

Name	Function	Description
Account Type	Public Function GetAccountType (MemberId As Integer) As AccountType myAccountType = api.Account.GetAccountType (MemberId)	Retrieves the Account type for the Member.

Name	Function	Description
Get Cell Account Type	GetCellLevelAccountType	Retrieves the Account type of the data cell based on its Account and flow settings.
Get Formula Type	Public Function GetFormulaType (MemberId As Integer) As FormulaType myFormulaType = api.Account.GetFormulaType (MemberId)	Returns the Formula Type if the Account is calculated.
Get Plug Account	Public Function GetPlugAccount (MemberId As Integer) As Member myMember = api.Account.GetPlugAccount (MemberId)	Retrieves the plug Account.

Consolidation

Name	Function	Description
Calculate	Public Sub Calculate(Formula As String, Optional onBeforeSetDataBuffer As BeforeSetDataBufferDelegate, Optional userState As Object, Optional arg0 As String, Optional arg1 As String, Optional arg2 As String, Optional arg3 As String, Optional arg4 As String, Optional arg5 As String, Optional arg6 As String, Optional arg7 As String) api.Data.Calculate(Formula, onBeforeSetDataBuffer, userState, arg0, arg1, arg2, arg3, arg4, arg5, arg6, arg7)	Executes a calculation for a specifically qualified Point of View.
Execute Default Elimination	Public Sub ExecuteDefaultElimination() api.ExecuteDefaultElimination()	Puts data into the Elimination Member of the Consolidation Dimension.

Name	Function	Description
Second Pass Eliminations	Boolean argument for use in FinanceFunctionType.Calculate rules Dim bValue As Boolean = args.CalculateArgs.IsSecondPassEliminationCalc	Used to calculate Data Units where Entity members are sibling members. Used to ensure source Entities are fully calculated at Eliminations
Execute Default Share	Public Sub ExecuteDefaultShare() api.ExecuteDefaultShare()	Puts data into the Share Member of the Consolidation Dimension.

Name	Function	Description
Execute Default Translation	Public Sub ExecuteDefaultTranslation (fxRuleTypeRevenueExp As FxRuleType, rateForRevenueExp As Decimal, fxRuleTypeAssetLiab As FxRuleType, rateForAssetLiab As Decimal) api.ExecuteDefaultTranslation (fxRuleTypeRevenueExp, rateForRevenueExp, fxRuleTypeAssetLiab, rateForAssetLiab) or api.ExecuteDefaultTranslation()	Calculates translation by bypassing consolidation and Business Rules.
Is Consolidated	Public Function IsConsolidated(MemberId As Integer) As TriStateBool myTriStateBool = api.Account.IsConsolidated (MemberId)	Reveals if the Account is consolidated.
Percent Consolidated	Public Function PercentConsolidation(Optional EntityId As Integer, Optional ParentId As Integer, Optional varyByScenarioTypeId As Integer, Optional varyByTimeId As Integer) As Decimal myDecimal = api.Entity.PercentConsolidation (EntityId, ParentId, varyByScenarioTypeId, varyByTimeId)	The percent that an Entity contributes to a Relationship.

Name	Function	Description
Translated Currency Member	Public Function IsForeignCurrencyForEntity (Optional EntityId As Integer, Optional consId As Integer) As Boolean myBoolean = api.Cons.IsForeignCurrencyForEntity (EntityId, consId)	Returns if the Consolidation Dimension Member is a translated currency Member. Used in translation rules.

Data

Name	Function	Description
Allocation	Use Journals for allocations	Allocates data across Dimensions (Entities, User Defined Dimensions, Accounts, etc.) with configurable weighting, all through Journals that can be previewed, are generated, are posted and can be unposted.

Name	Function	Description
Convert Data Buffer	api.Data.ConvertDataBuffer	Modifies Dimension Members for the cells in a Data Buffer using mapping.
Convert Data Buffer Extended Members	api.Data.ConvertDataBufferExtendedMembers	Automatically aggregates the data for extended Members in order to create data cells for Parent Members that are Base-Level Members in the destination Dimensions. This is used when copying data from a source Data Buffer created in another Cube or Scenario where one or more Dimensions have been extended.

Name	Function	Description
Get Data Buffer (working with Data Units)	Public Function GetDataBuffer (scriptMethodType As DataApiScriptMethodType, sourceDataBufferScript As String, expressionDestinationInfo As ExpressionDestinationInfo) As DataBuffer myDataBuffer = api.Data.GetDataBuffer (scriptMethodType, sourceDataBufferScript, expressionDestinationInfo)	Retrieves a Data Unit's values during a consolidation, calculation, or translation.
Get Data Buffer Using Formula	Dim myDataBufer As DataBuffer = api.Data.GetDataBufferUsingFormula ("A#Sales-A#Costs)	Use an entire math expression to calculate a final data buffer.

Name	Function	Description
Get Stored and Dynamically Calculated Values	Public Function GetDataCell(Formula As String, Optional arg0 As String, Optional arg1 As String, Optional arg2 As String, Optional arg3 As String, Optional arg4 As String, Optional arg5 As String, Optional arg6 As String, Optional arg7 As String) As DataCell myDataCell = api.Data.GetDataCell (Formula, arg0, arg1, arg2, arg3, arg4, arg5, arg6, arg7)	Retrieves the data contained in an intersection of the Cube. If this is a dynamically calculated value, the calculation will be run on the fly before returning the value.

Name	Function	Description
Has Cell Notes or Data Attachments	api.Data.GetDataCell(Formula, Parameters).CellStatus.HasAttachment	Indicates whether the data cell has notes or file attachments. (Boolean) To retrieve the data attachment, query the appropriate Member of the View Dimension (Annotation, AuditComment, Footnote, VarianceExplanation, Assumptions).
Has No Data	api.Data.GetDataCell(Formula, Parameters).CellStatus.IsNoData	Indicates if the cell has no data. (Boolean)
Has Valid Data	api.Data.GetDataCell (Formula,Parameters). CellStatus.DataIsTimeLogicDerived	Indicates if the data in a cell contains valid data that was derived from time logic. (Boolean)

Name	Function	Description
Get Data Buffer for Custom Share Calculation	Public Function GetDataBufferForCustomShareCalculation (Optional cubeId As Integer, Optional entityId As Integer, Optional ParentId As Integer, Optional scenarioId As Integer, Optional timeId As Integer, Optional viewId As Integer) As DataBuffer myDataBuffer = api.Data. GetDataBufferForCustomShareCalculation (cubeId, entityId, ParentId, scenarioId, timeId, viewId)	Use this function to assist in Custom Consolidations

Name	Function	Description
Get Data Buffer for Custom Elim Calculation	Public Function GetDataBufferForForCustomElimCalculation (Optional includeICNone As Boolean, Optional includeICPartners As Boolean, Optional combineImportFormsAndAdjConsolidatedInto Elim As Boolean, Optional cubeId As Integer, Optional entityId As Integer, Optional ParentId As Integer, Optional scenarioId As Integer, Optional timeId As Integer, Optional viewId As Integer) As DataBuffer myDataBuffer = api.Data. GetDataBufferForForCustomElimCalculation (includeICNone, includeICPartners, combineImportFormsAndAdjConsolidatedInto Elim, cubeId, entityId, ParentId, scenarioId, timeId, viewId)	Use this function to assist in Custom Calculations
Rounding	Use the VB.NET function for rounding: Round(expression[, numdecimalplaces])	Controls the level of rounding that is used.

Name	Function	Description
Set and Clear Data	Public Sub SetDataCell(Memberscript As String, amount As Decimal, isNoData As Boolean) api.Data.SetDataCell(Memberscript, amount, isNoData)	Sets data to a certain value.
Show Cell Text	Data Attachment Members in the View Dimension: Annotation, Assumptions, AuditComment, Footnote and VarianceExplanation	Use these View Members to display data attachment text within a Cube View, which can be used to edit text or display on Reports.
Variance	((A-B)/Abs(B))	Returns a Variance, but does not consider Account Type
BetterWorse Difference	Revenue Accounts:((A-B/Abs(B))) Expense Accounts: ((-1*A)-(-1*B))	Returns a Variance based on the Account Type.

Name	Function	Description
VariancePercent	((A-B) / Abs(B)) * 100	Returns a Variance Percent but does not consider Account Type.
BetterWorse Percent	Revenue Accounts: ((A-B) / Abs(B)) * 100 Expense Accounts: (((-1 * A) – (-1 * B))/Abs(B)) * 100	Returns a Variance Percentage based on the Account Type.
Unbalanced Math: Add, Subtract, Multiply, and Divide	Api.Data.Calculate(“A#TargetAccount = AddUnbalanced (A#60000,A#41000:O#Top, O#Top)”)	Required to perform math using two Data Buffers where additional dimensionality must be specified for the second Data Buffer.

Entity

Name	Function	Description
Default Currency	Public Function GetLocalCurrency(Optional EntityId As Integer) As Currency myCurrency = api.Entity.GetLocalCurrency (EntityId)	Retrieves the assigned Currency for the Entity or Parent.
Is Descendant	Public Function IsDescendent(dimPk As DimPk, ancestorMemberId As Integer, descendentMemberId As Integer, dimDisplayOptions As DimDisplayOptions) As Boolean myBoolean = api.Members.IsDescendent(dimPk, ancestorMemberId, descendentMemberId, dimDisplayOptions)	Returns if the Member is a Descendant of another Member. (Boolean)
Is Intercompany	Public Function IsIC(Optional EntityId As Integer) As Boolean myBoolean = api.Entity.IsIC(EntityId)	Returns if the Entity or Account Member is an intercompany Member.

Flow

Name	Function	Description
Switch Sign	Public Function SwitchSign(MemberId As Integer) As Boolean myBoolean = api.Flow.SwitchSign (MemberId)	Flow Dimension only. Responds as to whether credits are switched to debits for the specified Member for Revenue / Expense Accounts. (Boolean)
Switch Type	Public Function SwitchType(MemberId As Integer) As Boolean myBoolean = api.Flow.SwitchType (MemberId)	Flow Dimension only. Responds as to whether Account types are switched for the current or specified Member. This can drive translating this Member by a different FX Rate Type. (Boolean)

FX

Name	Function	Description
Currency Type	Public Function GetDefaultCurrencyId(Optional CubeId As Integer) As Integer myInteger = api.Cubes.GetDefaultCurrencyId(CubeId) or Public Function GetCurrency(currencyName As String) As Currency myCurrency = api.Cons.GetCurrency(currencyName)	Retrieves the currency type for the Cube or the Consolidation Dimension Member.
Current Exchange Rate	Public Function GetStoredFxRate(fxRateType As FxRateType, Optional timeId As Integer, Optional sourceCurrencyId As Integer, Optional destCurrencyId As Integer) As FxRate myFxRate = api.FxRates.GetStoredFxRate (fxRateType, timeId, sourceCurrencyId, destCurrencyId)	Retrieves the current exchange rate for the specified Entity.
Exchange Rate Calculated	Public Function GetCalculatedFxRate(fxRateType As FxRateType, timeId As Integer) As Decimal myDecimal = api.FxRates.GetCalculatedFxRate (fxRateType, timeId)	Calculates the exchange rate from the default currency to another.

Name	Function	Description
Get FX Rate Type for Asset / Liability Accounts	Public Function GetFxRateTypeForAssetLiability (Optional CubeId As Integer, Optional ScenarioId As Integer) As FxRateType myFxRateType = api.FxRates.GetFxRateTypeForAssetLiability(CubeId, ScenarioId)	Retrieves the default Rate Type for Asset and Liability Accounts in this Cube or Scenario (overrides Cube value).
Get FX Rate Type for Revenue/ Expense Accounts	Public Function GetFxRateTypeForRevenueExp (Optional CubeId As Integer, Optional ScenarioId As Integer) As FxRateType myFxRateType = api.FxRates.GetFxRateTypeForRevenueExp(CubeId, ScenarioId)	Retrieves the default Rate Type for Revenue and Expense Accounts in this Cube or Scenario (overrides Cube value).

Name	Function	Description
Get FX Rule Type for Asset / Liability Accounts	Public Function GetFxRuleTypeForAssetLiability (Optional CubeId As Integer, Optional ScenarioId As Integer) As FxRuleType myFxRuleType = api.FxRates.GetFxRuleTypeForAssetLiability(CubeId, ScenarioId)	Retrieves the default translation Rule Type for Asset and Liability Accounts in this Cube or Scenario (overrides Cube value).
Get FX Rule Type for Revenue/ Expense Accounts	Public Function GetFxRuleTypeForRevenueExp (Optional CubeId As Integer, Optional ScenarioId As Integer) As FxRuleType myFxRuleType = api.FxRates.GetFxRuleTypeForRevenueExp(CubeId, ScenarioId)	Retrieves the default translation Rule Type for Revenue and Expense Accounts in this Cube or Scenario (overrides Cube value).

Name	Function	Description
Translate	Public Sub Translate(sourceDataBufferScript As String, destDataBufferScript As String, fxRuleType As FxRuleType, rate As Decimal) api.Data.Translate(sourceDataBufferScript, destDataBufferScript, fxRuleType, rate)	Performs the translation method assigned to the Cube or Scenario (Periodic or Direct).

Journals

Name	Function	Description
Journal Postings Allowed	Public Function AllowAdjustments(Optional EntityId As Integer, Optional varyByScenarioTypeId As Integer, Optional varyByTimeId As Integer) As Boolean myBoolean = api.Entity.AllowAdjustments(EntityId, varyByScenarioTypeId, varyByTimeId)	Results determine if Journal postings are allowed for the Member.

Name	Function	Description
Journal Postings from Children Allowed	Public Function AllowAdjustmentsFromChildren (Optional EntityId As Integer, Optional varyByScenarioTypeId As Integer, Optional varyByTimeId As Integer) As Boolean myBoolean = api.Entity.AllowAdjustmentsFromChildren(EntityId, varyByScenarioTypeId, varyByTimeId)	Results determine if Journal postings from children are allowed for this Member.

Member

Name	Function	Description
Base Members	HasChildren = false, e.g. A#Root.Children (HasChildren=False)	Determines if the Member is a base Member.
Get Member ID	api.Members.GetMember(dimTypeId, MemberName).MemberPk.MemberID or api.POV.Dimension.Memberpk.Memberid	Retrieves the name for the selected Member.
Get Member Name	api.Members.GetMember(dimTypeId, MemberName).Name	Retrieves the name for the selected Member.

Name	Function	Description
Member Lists	See section of documentation on Creating Member Lists	Retrieves the Members from a named list stored in a Business Rule.
Member Name	api.POV.AccountDim, api.POV.EntityDim, etc	Retrieves the Member name.
Member Name or ID	Get Member name from Member ID: Public Function GetMember(dimTypeId As Integer, MemberId As Integer) As Member myMember = api.Members.GetMember(dimTypeId, MemberId) Get Member ID from Member name: Public Function GetMember(dimTypeId As Integer, MemberName As String) As Member myMember = api.Members.GetMember(dimTypeId, MemberName)	Retrieves the Member for the specified ID number or name.

Name	Function	Description
Top Member	api.Account.GetTopMemberForDimType (AccountMemberId, dimTypeForTopMember)	Retrieves the top Member of the selected Dimension.
Is Base	Public Function IsBase(dimPk As DimPk, ancestorMemberId As Integer, baseMemberId As Integer, Optional dimDisplayOptions As DimDisplayOptions) As Boolean myBoolean = api.Members.IsBase(dimPk, ancestorMemberId, baseMemberId, dimDisplayOptions)	Determines whether the POV Member is the Base of a defined Member.
Get First Common Parent	Public Function GetFirstCommonParent(dimPk As DimPk, topMostMemberId As Integer, MemberIdA As Integer, MemberIdB As Integer, Optional dimDisplayOptions As DimDisplayOptions) As Member myMember = api.Members.GetFirstCommonParent (dimPk, topMostMemberId, MemberIdA, MemberIdB, dimDisplayOptions)	Returns the first common Parent between multiple Members.

Scenario

Name	Function	Description
Scenario Consolidation View	Public Function GetConsolidationView (Optional ScenarioId As Integer) As ViewMember myViewMember = api.Scenario.GetConsolidationView (ScenarioId)	Determines if the Scenario's Consolidation View is set to YTD or Periodic.
Scenario Default View	Public Function GetDefaultView (Optional ScenarioId As Integer) As ViewMember myViewMember = api.Scenario.GetDefaultView (ScenarioId)	Retrieves the Scenario's Default View.
Scenario Input Frequency	Public Function GetInputFrequency (Optional ScenarioId As Integer) As Frequency myFrequency = api.Scenario.GetInputFrequency (ScenarioId)	Retrieves the Scenario's Input Frequency.

Status

Name	Function	Description
Calc Status	Public Function GetCalcStatus(Optional CubeId As Integer, Optional EntityId As Integer, Optional ParentId As Integer, Optional consId As Integer, Optional ScenarioId As Integer, Optional timeId As Integer) As CalcStatus myCalcStatus = api.CalcStatus.GetCalcStatus(CubeId, EntityId, ParentId, consId, ScenarioId, timeId)	Retrieves the calculation status for the data intersection.
Impact Calc Status	Public Sub ImpactCalcStatus(Optional CubeId As Integer, Optional EntityId As Integer, Optional ParentId As Integer, Optional consId As Integer, Optional ScenarioId As Integer, Optional timeId As Integer) api.CalcStatus.ImpactCalcStatus(CubeId, EntityId, ParentId, consId, ScenarioId, timeId)	Changes the status of the specified Data Unit to impact.

Time

Name	Function	Description
MemberId for the Month to which a Week belongs	Dim timeIdForMonth As Integer = BRApi.Finance.Time.ConvertIdToClosestIdUsingAnotherFrequency (si, timeIdForWeek, Frequency.Monthly)	Weekly applications only: This determines to which month a specific week belongs. Used within a Finance Business Rule.

Functions

Name	Function	Description
Days Sales Outstanding	Dim cell As DataCell = api.Functions.GetDSODataCell (acctsReceivableMember, salesMember)	Calculates Days Sales Outstanding (see below).

Name	Function	Description
Dynamic Simulation of Consolidation	api.Functions.GetEntityAggregationDataCell (memberScript as string, Optional useStoredAmountsWhereCalcStatusIsOK as Boolean, Optional fxRateTypeRevenueExpOverride as String, Optional fxRuleTypeRevenueExpOverride as String, Optional fxRateTypeAssetLiabOverride as String, Optional fxRuleTypeAssetLiabOverride as String)	Pseudo/approximate consolidation of a data cell (see Dynamic Simulation of Consolidation)
GetStage or GetCustom	Download the OneStream API Overview Guide from Solution Exchange for examples: GetStageBlendTextUsingCurrentPOV GetStageBlendText GetStageBlendNumberUsingCurrentPOV GetStageBlendNumber GetStageBlendDataTableUsingCurrentPOV GetStageBlendDataTable GetCustomBlendDataTableUsingCurrentPOV GetCustomBlendDataTable	Several functions that retrieve text and calculate values by reading values from the OneStream Stage or a custom relational table.

Examples of Key Functions in Use

Calculate Data

The most common function used is api.data.calculate, which sets the value of one or more values (left side of Formula) equal to another (right side). A final argument (optional) can be added as True or False as to whether to use a data cell Storage Type of Durable. Durable data will not be cleared automatically when a Data Unit is re-calculated. It can only be cleared by calling api.Data.ClearCalculatedData with the clearDurableCalculatedData boolean property set to True.

```
api.Data.Calculate(formula, isDurableCalculatedData)
```

For example, the following Stat Account is used to calculate Total Cost of Sales for three months and is not set to be a Durable storage method (optional argument):

```
api.data.calculate("A#TOT_COS_LAST3:V#YTD = A#TOT_COS:V#Periodic + A#TOT_
COS:T#POVPrior1:V#Periodic +
       A#TOT_COS:
       T#POVPrior2:V#Periodic", False)
```

An alternative to this overloaded function is to provide Member Filters (all optional) that can be used to filter the results before saving them to the target to affect fewer intersections, such as only to be applied to certain Flow members:

```
api.Data.Calculate(formula, accountFilter, flowFilter, originFilter, icFilter, ud1Filter,
ud2Filter,
       ud3Filter, ud4Filter,
       ud5Filter, ud6Filter, ud7Filter, ud8Filter, onEvalDataBuffer, userState,
isDurableCalculatedData)
```

Another alternative allows the use of arguments to be applied:

```
api.Data.Calculate(Formula, onBeforeSetDataBuffer, userState, arg0, arg1, arg2, arg3, arg4, arg5,
arg6,
       arg7)
```

Clear Calculated Data

Clearing calculated data is performed when calculated Members need to be reset to NoData. This will result in the Data Units requiring a calculation. Note that the final argument must be true to clear any data with a Storage Type of Durable. See Calculate Data for how to set a calculation to store data as Durable.

```
api.Data.ClearCalculatedData(dataBufferScript, clearCalculateData, clearTranslationData,
       clearConsolidationData,
       clearDurableCalculatedData)

api.Data.ClearCalculatedData("A#[cash deposits]:ud1#Production", true, true, true, false)
```

DataBuffer

The DataBuffer log helps debugging and troubleshooting. It allows datastring results to be written to any target and multiple string results can be merged. The output is formatted as Tab delimited for copy/paste to Excel. Results include the DataBuffer formula, the size and number of records and displays the XFCommon member PK. It also contains a default record limit to guard against log explosion.

Examples of DataBuffer outputs:

- Error Log
- Send to SQL
- Send to a File
- Email
- FTP

When setting a value equal to another value, the item on the left side of the expression is the value being set, and the item on the right side is the value being queried or calculated to set the left side. Example: F#BeginBalance = F#EndingBalance.T#POVPrior1 would set the beginning balance in the Flow Dimension to the prior period's ending balance. In a Business Rule, the DestinationInfo is the left side of the equation while a GetDataBuffer is the right side of the equation.

```
Dim destinationInfo As ExpressionDestinationInfo = api.Data.GetExpressionDestinationInfo
       ("A#EBITDA:UD1#Tires")
Dim sales As DataBuffer = api.Data.GetDataBuffer("A#Sales:UD1#Tires", destinationInfo)
Dim operatingExpenses As DataBuffer = api.Data.GetDataBuffer("A#OperatingExpenses:UD1#Tires",
       destinationInfo)
Dim ebitda As DataBuffer = (sales – operatingExpenses)
api.Data.SetDataBuffer(ebitda, destinationInfo)
```

This translates to the following equation:

```
A#EBITDA:UD1#Tires=A#Sales:UD1#Tires - A#OperatingExpenses:UD1#Tires
```

GetDataBufferUsingFormula

Use an entire math expression to calculate a final data buffer. Api.Data.GetDataBufferUsingFormula can perform the same data buffer math as api.Data.Calculate, but the result is assigned to a variable where api.Data.Calculate saves the calculated data.

Example

```
Dim myDataBuffer As DataBuffer = api.Data.GetDataBufferUsingFormula ("A#Sales-A#Costs")
```

Loop over the contents of myDataBuffer to conditionally change each data cell.

Formula Variables

There is additional capability to using Formula Variables to achieve the same level of flexibility and integration as using Evals. After creating a data buffer variable, name it as a Formula Variable and reference it inside api.Data.Calculate or other calls to api.Data.GetDataBufferUsingFormula. This provides flexibility and can improve performance because the Data Buffer is calculated once and the variable is re-used multiple times.

Example

```
Dim myDataBuffer As DataBuffer = api.Data.GetDataBufferUsingFormula ("A#Sales-A#Costs")
api.Data.FormulaVariables.SetDataBufferVariable("myDataBuffer",myDataBuffer,False)
api.Data.Calculate("A#Profit=A#5000 + $myDataBuffer
```

Use api.Data.FormulaVariables.SetDataBufferVariable to name the data buffer. Pass in any name followed by the data buffer variable. Enter a True/False value for Uses Indexes to Optimize Repeat Filtering. Using True will re-use the same data buffer using FilterMembers and improve performance. After naming the data buffer, use a dollar sign and the name when referencing it in a script.

FilterMembers

Use this inside of an api.Data.Calculate or api.Data.GetDataBufferUsingFormula script.

Example

```
Dim myDataBuffer As DataBuffer
Api.Data.GetDataBufferUsingFormula("FilterMembers(A#All,A#6000,[A#[3600].Base])")
```

Change a data buffer and only include numbers for the specified Dimensions. The first parameter is the starting data buffer. This can be a variable name or an entire math equation in parentheses. There can be as many parameters as needed to specify Member Filters and different Member Filters can be used for multiple Dimension types. The resulting filtered data buffer will only contain numbers that match the Members in the filters.

RemoveMembers

This uses the same syntax as FilterMembers, but it takes the data cells away for the specified Members instead of keeping them.

Retrieving Member Names from IDs

This retrieves the Member names from the IDs when looping over the cells in a data buffer.

Example

```
For Each sourceCell As DataBufferCell In myDataBuffer.DataBufferCells.Values
     Dim accountName As String = sourceCell.DataBufferCellPk.GetAccountName(api)
     Dim ud1Name As String = sourceCell.DataBufferCellPk.GetUD1Name(api)
     If ud1Name = "None" Then BRApi.ErrorLog.LogMessage(si, "UD1"& ud1Name, Nothing)
Next
```

Logging Contents of a Data Buffer

Log the contents of a Data Buffer to the Error Log when writing Business Rules in order to make corrections and troubleshoot along the way.

Example

myDataBuffer.LogDataBuffer(api,”MyDataBufferOutput”,1000)

The third Parameters (1000) indicates the maximum number of cells to include in the log and displays what is in the data buffer.

Example api

api.LogMessage(XFErrorLevel.Information, "MyDataBuffer As a CSV String For Excel", myDataBuffer.GetCSVString(api, False, 1000))

The false Parameter specifies whether to include Member IDs in the output. Member names are always included.

Remove Functions in Formulas

Remove Zeros

The RemoveZeros function evaluates a source data buffer and removes all cells where the amount is zero whether it is flagged as a NoData cell or not. Therefore, it is not necessary to use both RemoveZeros and RemoveNoData together since RemoveZeros handles both situations. This can be identified on individual cells as the Cell Amount under Cell Status.

This function is important to use for performance purposes. Use the RemoveZeros function in calculations where there is a substantial amount of No Data or 0.00 cells in Data Units. This can be determined by looking at the Data Unit Statistics when right-clicking on a cell in a Cube View.

⊟ General	
Total Number of Stored Records	1637
⊟ NODATA Status	
Number of NODATA Cells	4
Number of Zero Cells	13
Number of Real Cells	1617
Number of Derived Cells	3

Example using api.Data.Calculate:

```
'Run for base Entities and local currency only
    If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then
    api.Data.Calculate("A#CashCalc=RemoveZeros(A#10000)")
End If
```

Example using GetDataBufferUsingFormula:

```
If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then

    'Get Data Buffer by using GetDataBufferUsingFormula
    Dim dataBufferExample As DataBuffer = api.Data.GetDataBufferUsingFormula("RemoveZeros(A#100000)")
    'Set Data Buffer Variable to pass into api.Data.Calculate formula.  Can be used for multiple instances of api.Data.Calculate
    'Create a unique name to name the Data Buffer as a Formula Variable
    api.Data.FormulaVariables.SetDataBufferVariable("dataBufferExample", dataBufferExample, False)
    'Pass Formula Variable into api.Data.Calculate using a $
    api.Data.Calculate("A#CashCalc = $dataBufferExample")

End If
```

Remove NoData

The RemoveNoData function evaluates a source data buffer and removes data cells that have a cell amount of NoData.

This function is important to use for performance purposes. Use the RemoveNoData function in calculations where there is a substantial number of cells with a Cell Status of NoData in Data Units. This can be determined by looking at the Data Unit Statistics when right-clicking on a cell in a Cube View.

⊟ General	
Total Number of Stored Records	1637
⊟ NODATA Status	
Number of NODATA Cells	4
Number of Zero Cells	13
Number of Real Cells	1617
Number of Derived Cells	3

Example using api.Data.Calculate:

```
'Run for base Entities and local currency only
    If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then
    api.Data.Calculate("A#CashCalc=RemoveNoData(A#10000)")
End If
```

Example using GetDataBufferUsingFormula:

```
If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then

    'Get Data Buffer by using GetDataBufferUsingFormula to do the math
    Dim dataBufferExample As DataBuffer = api.Data.GetDataBufferUsingFormula("RemoveNoData(A#10000)")
    'Set Data Buffer Variable to pass into api.Data.Calculate formula.  Can be used for multiple instances of api.Data.Calculate
    'Create a unique name to name the Data Buffer as a Formula Variable
    api.Data.FormulaVariables.SetDataBufferVariable("dataBufferExample", dataBufferExample, False)
    'Pass variable into api.Data.Calculate using a $
    api.Data.Calculate("A#CashCalc = $dataBufferExample")

End If
```

Performance Note: In the calculation performance testing for a single Data Unit, the use of RemoveZeros and RemoveNoData using api.Data.Calculate and/or GetDataBufferUsingFormula rendered a significant performance advantage. This is testing the before and after calculation time of a single formula change for a Data Unit. Times may vary from Data Unit to Data Unit and Application to Application. However, the use of RemoveZeros and RemoveNoData is highly recommended in formulas where Cube and Dimensionality designs lend to sparse data models. This is not limited to sparse data models as this can be identified in dense data models as well.

Dynamic Calc Using GetDataCell

A Dynamic Calc returns an Account value on demand without storing the data in the Cube. These Members cannot be referenced by other Members in calculating their values. Dynamic Calcs are best used in Members required for reporting only. One ideal use is the GetDataCell function in ratio-style statistical Accounts (e.g. Current Ratio).

```
Return api.Data.GetDataCell("Divide(A#[CurrentAssets], A#[CurrentLiabilities]")
```

Dynamic Filters and Aliases

Create Dynamic Member Filters to assign to rows or columns within Cube Views. A common example of a Variance column that subtracts one Member from another is shown below. Adding the :Name(“Membername”) option at the end gives the ability to assign a display name to the header of the Cube View. The double quotes in the Name() function are optional.

```
Cube View Filter: GetDataCell(“Variance(T#POV,T#Prior12)”):Name("Variance")
```

Error Traps

```
Try
  if api.POV.Cons.name =api.POV.GetEntityCurrency() then
     api.Data.Calculate("A#Cash1=A#[Restricted Cash] + 70000")
  End If
catch ex as exception
api.LogError(ex)
End Try
```

Math Functions

Use the math functions built into VB.NET.

Examples:

Absolute Value - math.abs()

Rounding - math.round()

Several other supported math functions and examples are listed here:

http://msdn.microsoft.com/en-us/library/thc0a116%28v=VS.90%29.aspx

Member List Functions

When a Cube filter is run and the filter value does not match a Member, the system looks in all Business Rules for the MemberListProperty property to find a matching Member list name. Users should create MemberLists in Business Rules that are not assigned to Cubes because the entire Business Rule will be run if the system finds a matching Member list name. Reference MemberLists like this: E#Root.PartnerList.

The following example first declares the two MemberListHeader names of Sample Member List and PartnerList. It then addresses each MemberList. Sample Member List shows a way to get the children of an Entity. E#Texas.[Sample Member List] is just another way of saying E#Texas.Children. In the PartnerList example, E#Root.PartnerList will generate a list that includes Paris and Nice.

```
Case Is = FinanceFunctionType.MemberListHeaders
' Additional logic that defines the names of the two custom Member Lists
   Dim myMemberListHeaders = New List(Of MemberListHeader)myMemberListHeaders.Add(new
   MemberListHeader("Sample Member List"))myMemberListHeaders.Add(new
   MemberListHeader("PartnerList"))
Return myMemberListHeaders

Case Is = FinanceFunctionType.MemberList
   If args.MemberListArgs.MemberListName = "Sample Member List" Then
            Dim myMemberListHeader = new MemberListHeader(args.MemberListArgs.MemberListName)
            Dim myMembers = new List(Of Member)()
            Dim myMemberList = New MemberList(myMemberListHeader,
            myMembers)myMembers.AddRange(api.Members.GetChildren(args.MemberListArgs.DimPk,
           args.MemberListArgs.TopMember.MemberPk.MemberId,args.MemberListArgs.DimDisplayOptions))
           Return myMemberList
   Else If args.MemberListArgs.MemberListName = "PartnerList" Then
           Dim myMemberListHeader = new MemberListHeader(args.MemberListArgs.MemberListName)

                Dim myMembers = new List(Of Member)()
           Dim myMemberList = New MemberList(myMemberListHeader, myMembers)
         myMembers.AddRange(api.Members.GetBaseMembers(args.MemberListArgs.DimPk,

         api.Members.GetMember(args.MemberListArgs.DimPk.dimtypeid,"Paris").Memberpk.Memberid,

         args.MemberListArgs.DimDisplayOptions))
         myMembers.AddRange(api.Members.GetBaseMembers(args.MemberListArgs.DimPk,
         api.Members.GetMember(args.MemberListArgs.DimPk.dimtypeid,"Nice").Memberpk.Memberid,
         args.MemberListArgs.DimDisplayOptions))
         Return myMemberList
End If
```

POV Object

In Business Rules, only Data Unit Dimensions are valid in the POV object, not Account or User Defined Members, so the code below only works in Business Rules:

```
 Dim AcctID as Integer=api.POV.Account.MemberPk.MemberID


Dim AcctID as Integer=api.POV.GetDataCellPk.AccountID
```

Retrieve Time Varying Property of Relationship's % Consolidated Entity Attribute

```
'Get Entity ID, Parent Entity ID, Scenario ID, Scenario Type ID and Time ID
dim EntityID as integer= api.POV.Entity.Memberpk.Memberid
dim ParentID as integer= api.POV.Parent.Memberpk.Memberid
dim ScenarioID as integer=api.POV.Scenario.Memberpk.Memberid
dim ScenarioTypeID as integer = api.Scenario.GetScenarioType(ScenarioID).id
dim TimeID as integer = api.Time.GetIdFromName("2013M1")
'Get time varying node % consolidation property value for current Scenario POV and 2013M1
dim myDecimal as decimal= api.Entity.PercentConsolidation(EntityId, ParentId, ScenarioTypeId,
TimeId)
'Log the time varying property
api.logerror(xfErrorLevel.Information,myDecimal & " " & api.POV.Entity.name & "." &
api.POV.Parent.name)
```

> **TIP:** API.POV.Parent object is only relevant for relationship level Consolidation Members

SetDataCell

Use this to set a value equal to another value. Any dimensions listed on the right side must have a member specified. You don't need to have all dimension on the right side but if they are not specified, the current POV is used.

```
api.Data.SetDataCell(Memberscript, amount, isNoData)
api.Data.SetDataCell("A#[Restricted Cash]:O#Forms:F#None:IC#None:U1#None:U2#None:U3#None:U4#None:
       U5#None:U6#None:U7#None:
       U8#None", 50, False)
```

Translate

The api.data.translate function is the same as api.data.calculate, but aggregates AdjInput data into the AdjConsolidated Member.

```
'GetParentCurrency only returns a value when running a translate.
If api.Parameters.FunctionType = FinanceFunctionType.Translate Then if api.POV.Cons.Name =
   api.POV.GetParentCurrency.Name Then
   api.Data.Translate("A#[Restricted Cash]=A#[Restricted Cash]:C#[USD]*10")end if
End If
```

Unbalanced Math

The Unbalanced math functions are required when performing math with two Data Buffers where the second Data Buffer needs to specify additional dimensionality. The term Unbalanced is used because the script for the second Data Buffer can represent a different set of Dimensions from the other Data Buffer in the api.Data.Calculate text. These functions prevent data explosion.

In the examples below, the first two parameters represent the first and second Data Buffers on which to perform the function. The third parameter represents the Members to use from the second Data Buffer when performing math with every intersection in the first Data Buffer. The math favors the intersections in the first Data Buffer without creating additional intersections.

It is key that the dimensionality of the Target (left side of the equation) matches the dimensionality of the first data buffer on the right side of the equation (argument 1).

AddUnbalanced

```
api.Data.Calculate("A#TargetAccount = AddUnbalanced(A#OperatingSales, A#DriverAccount:U2#Global,
       U2#Global)")
```

SubtractUnbalanced

```
api.Data.Calculate("A#TargetAccount = SubtractUnbalanced(A#OperatingSales,
A#DriverAccount:U2#Global,
       U2#Global)")
```

DivideUnbalanced

```
api.Data.Calculate("A#TargetAccount =DivideUnbalanced (A#OperatingSales,
A#DriverAccount:U2#Global,
       U2#Global)")
```

MultiplyUnbalanced

```
api.Data.Calculate("A#TargetAccount =MultiplyUnbalanced (A#OperatingSales,
A#DriverAccount:U2#Global,
       U2#Global)")
```

Consider this example. A#OperatingSales has 100 stored records in January for a single Entity.

Because A#OperatingSales has a total of 100 stored values, A#TargetAccount will end up with 100 stored numbers and the amounts would be the values from A#OperatingSales plus/minus/multiplied/divided by whatever was found at A#DriverAccount:U2#Global for each of those 100 intersections.

This means that if there was no data in A#OperatingSales:U2#Widgets, then even though the UD2 Dimension is unspecified in the target and in the first Data Buffer expression, no record would be created, hence avoiding data explosion. The most common use case would be applying a driver for some of the Dimensions.

ConvertUnbalanced

This function is related to the Unbalanced Math functions (see Unbalanced Math later in this section) and used to convert a data buffer so that it is balanced with an api.Data.Calculate script where unbalanced math does not apply. This is necessary when using an api.Data.GetDataBufferUsingFormula to calculate a data buffer where the script was not balanced to match a formula in another script where a data buffer variable needs to be used.

In the example below, a myDataBuffer was created to have data for all stored accounts, but the subsequent api.Data.Calculate scripts expects each operand to use a specific account. The ConvertUnbalanced function filters the data buffer to only include the specified account name and it also converts the data buffer to make it balanced and consistent with the destination. The same data buffer can be re-used multiple times.

Example

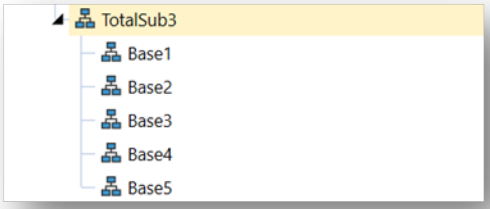
```
Dim myDataBuffer As DataBuffer = api.Data.GetDataBufferUsingFormula("A#All")
api.Data.FormulaVariables.SetDataBufferVariable("myDataBuffer ", myDataBuffer, True)
api.Data.Calculate("A#6050 = ConvertUnbalanced($myDataBuffer, A#6000) +
ConvertUnbalanced($myDataBuffer, A#3000)")
```

Finance Function Types

Use Finance Function Types, except for Custom Calculate, to run logic in certain calculation sequences during the calculation sequence while a Function Type is in process. There are thirteen Function Types:

Calculate
Additional logic during calculation of Entity, Consolidation Scenario and Time. This sets the value of one or more values (left side of Formula) equal to another (right side). It then executes a calculation for a specifically qualified Point of View. This is the most common function used.

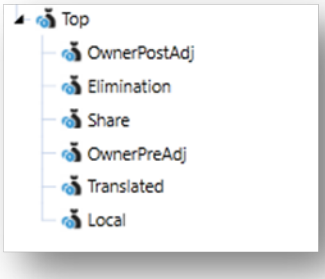
There are situations where the Entity being processed must access another Entity's data. In situations involving pulling Consolidation dimension Elimination results from other Entities as siblings, the multi-thread processing of the calculations requires an additional function to ensure the calculations are complete. Below is an example of this sibling relationship. The Entities Base1, Base2, etc. are siblings that would be calculated simultaneously during a consolidation:



For this purpose, the Calculate Finance Function Type supports the argument IsSecondPassEliminationCalc.

```
Case Is = FinanceFunctionType.Calculate

    If (api.Pov.Cons.MemberId = DimConstants.ConsElimination) And (args.CalculateArgs.IsSecondPassEliminationCalc) Then
```

During the Calculate process, this function allows Business Rules to execute after the Sibling Entities have calculated results to the Consolidation Elimination member. Below is a reference to the Consolidation dimension:



Once all the sibling Entity members are calculated to Elimination, the Business Rules within the IsSecondPassEliminationCalc will be executed.

Translate

Additional logic that uses custom translation.

FXRate

Custom logic used to determine Foreign Exchange rates for any intersection.

Consolidate Share

Additional logic used during the custom calculation of the Share Member.

Consolidate Elimination

Additional logic used during the custom calculation of the Elimination Member.

Custom Calculate

A CustomCalculate Finance Function Type can be used in order to execute a single year custom calculation via a Dashboard Parameter Component Server Task Action. This is considered a partial calculation and does not store the calculated data or run the calculation during a consolidation. Running a custom calculation from a Dashboard will impact calculation status for the affected data unit even if the data does not change. See Parameter Components in Presenting Data With Books, Cube Views and Dashboards for more details on how to assign this type of Finance Rule to a Dashboard. See Data Management for details on creating this type of Data Management Step. See Parameter Components for more information on passing arguments to a Custom Calculate function.

Example:

```
Select Case api.FunctionType
  Case Is = FinanceFunctionType.CustomCalculate
If args.CustomCalculateArgs.FunctionName.XFEqualsIgnoreCase("FunctionName") Then
Me.CalcTest(si, globals, api, args)
api.Data.Calculate("A#TFS2903 = A#10000 + A#69000)
End If
End Select
```

Calculation Drill Down Member Formula

Provides custom drill down results.

Conditional Input Rule

Conditional Input Rules make data cells read-only. While the settings for this can be done directly on the Cube, using a Conditional Input Business Rule offers more flexibility and still allows the use of the Cube settings. This rule can return the following: ConditionalInputResultType.Default, ConditionalInputResultType.NoInput, ConditionalInputResultType.NoInputAllowCellDetail, and ConditionalInputResultType.NoCellDetailAllowInput.

The following Business Rule example will make all cells for the Account 6000 read-only. This should be added to a Business Rule attached to a Cube.

```
Case Is = FinanceFunctionType.ConditionalInput
    If api.Pov.Account.Name.XFEqualsIgnoreCase("6000") Then
        Return ConditionalInputResultType.NoInput
    End If
    Return ConditionalInputResultType.Default
```

Confirmation Rule

Special logic that runs with Confirmation Rules.

Data Cell

Named GetDataCell calculations that can be reused such as a Better/Worse calculation in Cube Views.

Dynamic Calc Account

Special logic to use in Dynamic Calc members.

Member List and Member List Headers

A custom list of members for use in Cube Views and other areas. See Commonly Used Member Filter Functions in Cubes for more details on using custom lists in a Cube View.

Select Case api.FunctionType is the expression used when a certain process needs to be isolated and run special logic. See examples below:

```
Select Case api.FunctionType
Case Is = FinanceFunctionType.Calculate
    ' Additional logic to run with every calculation.
```

```
   api.Data.Calculate("A#DACash1= A#Cash4 + 1")
    api.Data.Calculate("A#2150:F#Movement = V#Periodic:A#5750:F#None") if api.metaData.Cons.name
="USD"
       then
    api.Data.Calculate("A#Cash1=A#[Restricted Cash]")
End If
Case Is = FinanceFunctionType.Translate
   ' Additional logic to run with every translation. If api.ExecuteDefaultTranslation is not

   included, then the standard translation process will not run.
    api.ExecuteDefaultTranslation
    dim Cubeid as integer = api.Cubes.GetCubeInfo("GolfStream").Cube.Cubeid
dim fxRule as fxRuleType=api.Cubes.GetFxRuleTypeForAssetLiability(CubeId)
api.Data.Translate("A#[Restricted Cash]:C#[USD]","A#[Restricted Cash]",fxRule,.5)
Case Is = FinanceFunctionType.ConsolidateShare then
    ' Additional logic to run when the Share Member of the Consolidation Dimension is
being calculated.
    api.ExecuteDefaultShare()
    api.data.calculate("A#[Restricted Cash]=A#[Restricted Cash]*.8")
    Case Is = FinanceFunctionType.ConsolidateElimination then
    ' Additional logic to run while the Elimination Member of the Consolidation Dimension
is being
       calculated.
    api.ExecuteDefaultElimination
    api.Data.Calculate("A#[IC AR]=A#[IC AR] * 1.1")
Case Is = FinanceFunctionType.MemberListHeaders
    ' Additional logic that defines the names of custom Member Lists
    Dim myMemberListHeaders = New List(Of MemberListHeader)myMemberListHeaders.Add(new
MemberListHeader
("Sample Member List"))
    Return myMemberListHeaders
Case Is = FinanceFunctionType.MemberList
    ' Additional logic that defines the Members within custom Member List that are included in
Business
       Rules.
```

```
 See the “MemberList Functions” section in the proceeding pages.
    'Example: "A#Root.[Sample MemberList]"
    If args.MemberListArgs.MemberListName = "Sample Member List" Then Dim myMemberListHeader =
new MemberListHeader(args.MemberListArgs.MemberListName)
    Dim myMembers = new List(Of Member)()
    Dim myMemberList = New MemberList(myMemberListHeader, myMembers)myMembers.AddRange
(api.Members.
       GetChildren
(args.MemberListArgs.DimPk,
args.MemberListArgs.TopMember.MemberPk.MemberId,args.MemberListArgs.DimDisplayOptions))
Return myMemberList
End If
End Select
```

Ultimate Ownership

This function calculates and stores Ultimate Ownership results for every ancestor/child Entity relationship. The current Entity being calculated is the ancestor Entity, and when storing the results, the Members in the Intercompany Dimension are used to represent each descendant Entity. This function assumes the source Direct Ownership numbers are weights and are typically entered by a user (e.g., numbers between 0.0 and 1.0).

C#OwnerPreAdj:O#AdjInput is using an account that does not consolidate and accepts manual entry for adjustments in those Parent/Child relationships. It cascades the user-entered amounts up the Entity tree and multiplies them when a Parent owns part of a Child and that Child owns part of a Grandchild.

sourceDirectOwnershipMemberscript represents the data cells where the user-entered Direct Ownership amounts are found for Parent/Child Entity combinations.

destUltimateOwnershipMemberscript represents the intersection for the resulting calculations. All results are stored using the current Data Unit being calculated.

This function is intended to run only for the Entity's Local Consolidation Member, and it uses IC Members to store the results for each descendant Entity.

Example: Use the following as a Member Formula in the UltimateOwnership account:

```
If api.Cons.IsLocalCurrencyForEntity() Then
api.Data.CalculateUltimateOwnership("C#OwnerPreAdj:A#DirectOwnership:O#AdjInput",
       "A#UltimateOwnership:O#Forms", 1.0)
End If
```

Here are the steps to set up Ownership entry and calculations in an application. This functionality is intended to be used in a separate Control Cube accessed via the main Financial Cube:

Create Two Accounts

DirectOwnership

This is for the user to enter Direct Ownership for each Parent/Child Entity combination.

AccountType = BalanceRecurring, IsConsolidated = False, Adjustment Type = Data Entry

UltimateOwnership

This stores the calculated results in Parent Entities using IC Members for every descendant Entity.

AccountType = BalanceRecurring, FormulaType = Formula Pass 1, Allow Input = False, IsConsolidated = False, Is IC Account = Conditional

Formula:

```
If api.Cons.IsLocalCurrencyForEntity() Then
api.Data.CalculateUltimateOwnership("C#OwnerPreAdj:A#DirectOwnership:O#AdjInput",
       "A#UltimateOwnership:O#Forms", 1.0)
End If
```

Create a Cube View for entering Direct Ownership Amounts

Cube View POV

Consolidation = OwnerPreAdj, Origin = AdjInput, Flow, IC and UDs = None

Column

A#DirectOwnership

Row

E#Root.TreeDescendants

Create a Cube View for Running a Consolidation and Viewing the Ultimate Ownership Results

Cube View POV

Consolidation = Local, Account = UltimateOwnership, Origin = Forms, Flow IC and UDs = None

Column

IC#ICEntities.Base

Row

E#Root.TreeDescendants

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.

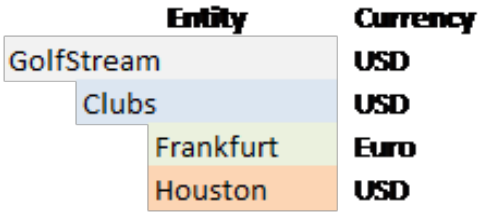
Translation algorithms (Use default or implement FinanceFunctionType.Translate in a BR)

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



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 vs. Consolidate.

Calculation Status State 1: Actual Scenario, Loading for December

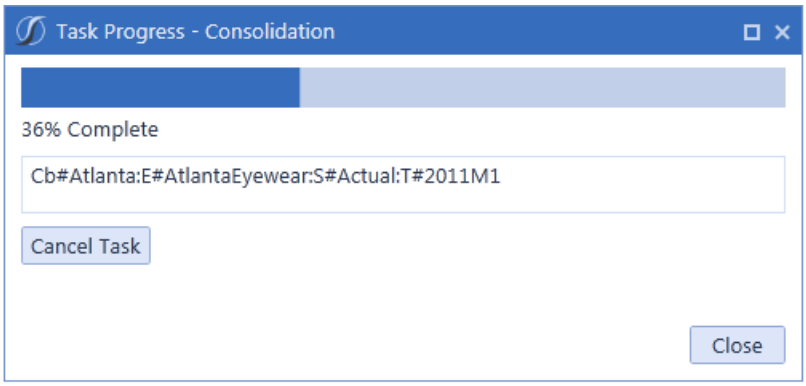
In this case Consolidate will perform better than Force Consolidate because every month prior to December has an OK Calculation Status. This means the data for that month has not changed since the last consolidation and only the month of December needs to be consolidated. If a Force Consolidate was used, 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 Point of View 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 in Cubes for additional details.

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 screen shot 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.

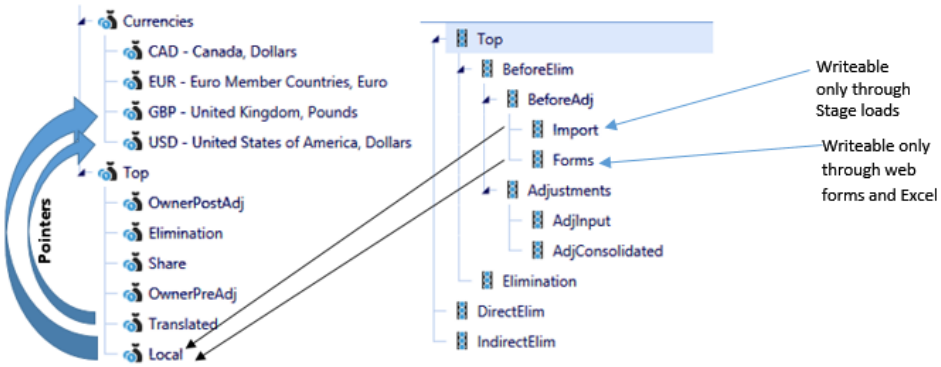
The consolidation process starts at Local, which is the same data as one of the currencies under the Currencies Member (based on the Entity's default currency setting). If the Entity is base-level, the data can be loaded into the Local Consolidation Member using the Import Origin Member, and data can 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



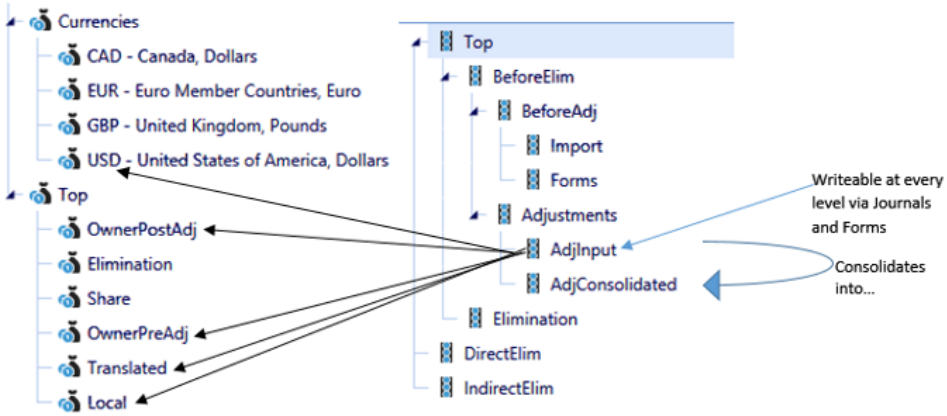
Adjustments

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

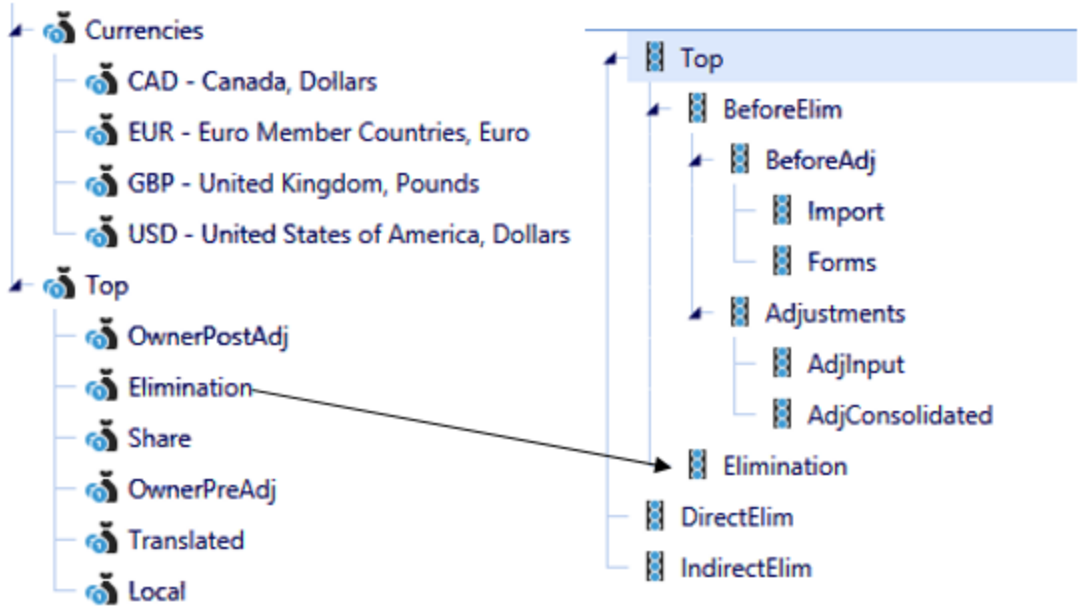


Eliminations

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.

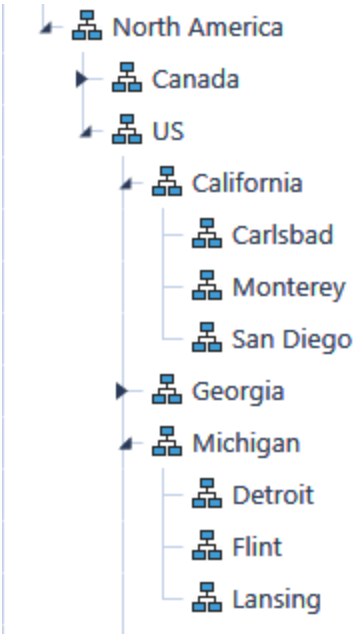
Users can drill down into the O#Elimination Member in a Parent to show the elimination entries made in both the Parent and child Entities.

Consolidation Dimension Origin Dimension



Intercompany Elimination

Intercompany Elimination is the process of cancelling out account balances for intercompany partners for intercompany accounts with any unresolved balance being placed in a Plug Account.

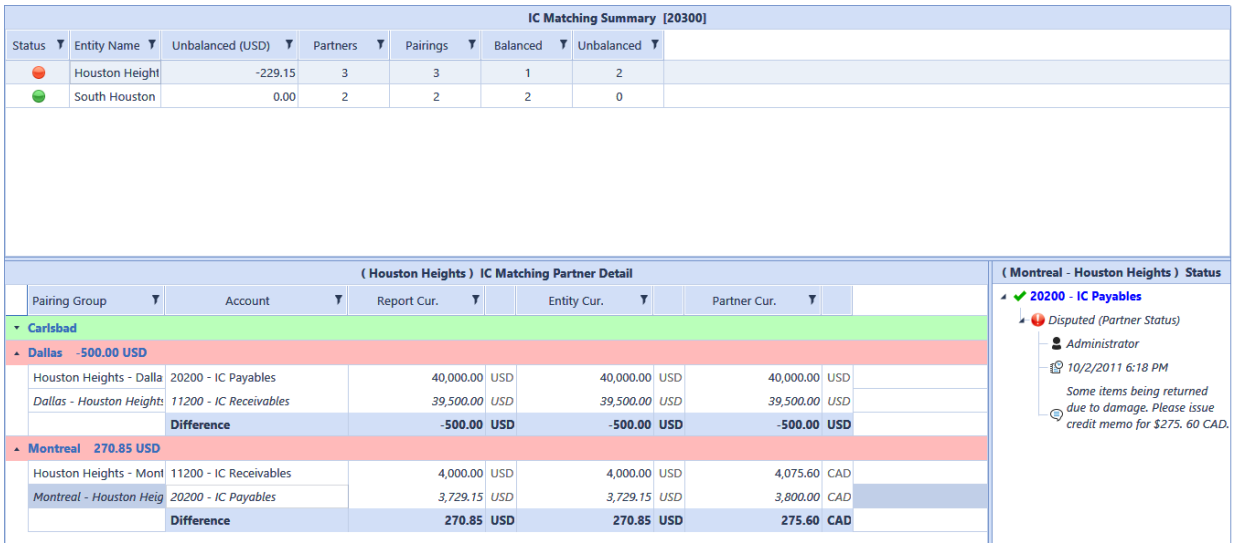


The Entity structure above belongs to GolfStream, a fictitious golf manufacturer. If the Detroit Entity sold golf club shafts to Monterey who assembles the final club product, this would be an intercompany transaction. The following prerequisites must exist for the transaction to eliminate.

- Monterey and Detroit must have there Is IC Entity property set to True.
- The Accounts 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.*

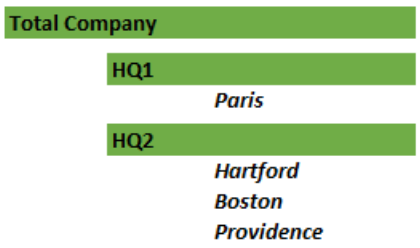
Houston Heights is shown in red because there is a $229.51 discrepancy. Details on the discrepancy are shown in the lower part of the screen. Houston Heights' discrepancy with Montreal is shown at the bottom in the application reporting currency (USD), Houston Heights' currency (USD), and the partner 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 Workflow for more details on how to set up Intercompany Matching via Workflow.

> **NOTE:** Using Extensible Dimensionality to extend Accounts used within the Intercompany Matching process is not a recommended practice.

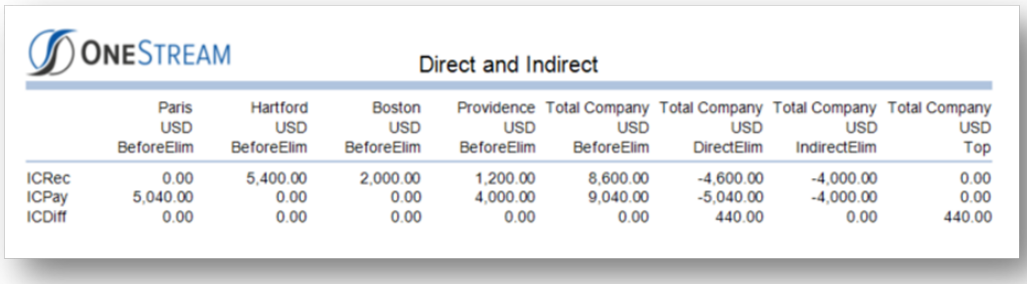
Direct and Indirect Eliminations

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 aparent 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 Cubes in Calculation section for 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 Business Rule section on applying Business Rule logic by Finance Function Type to discover what you can customize.
- Formulas 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).

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.

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.

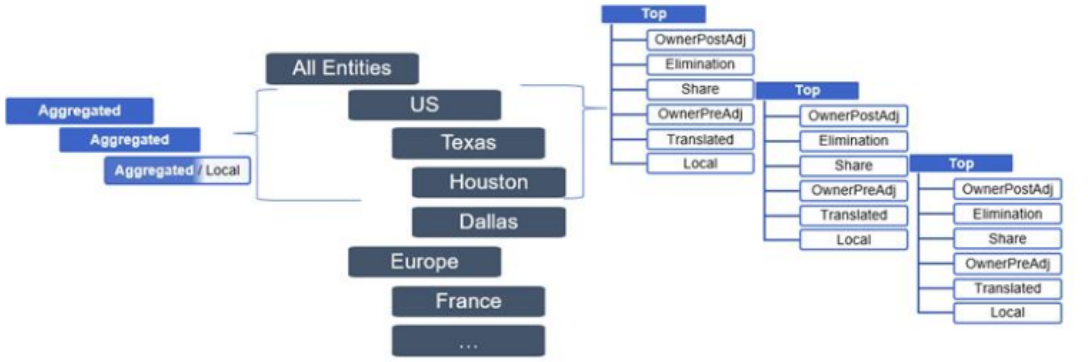
For base level entities, the Aggregated member displays the data that is stored in the “Local” member.

For parent entities, the Aggregated member stores the results of the Entity Aggregation process that occurs on its children.

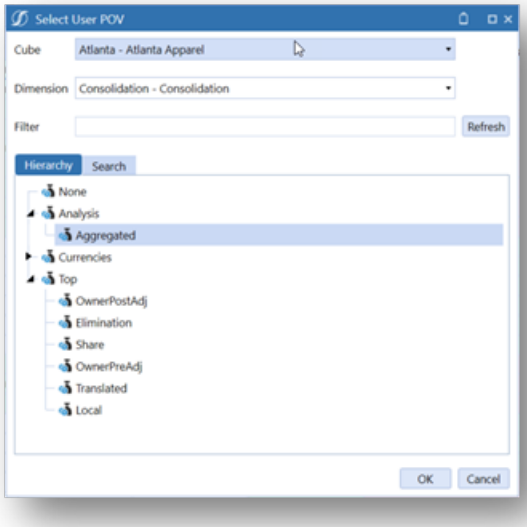
Aggregation Algorithm:

- 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.

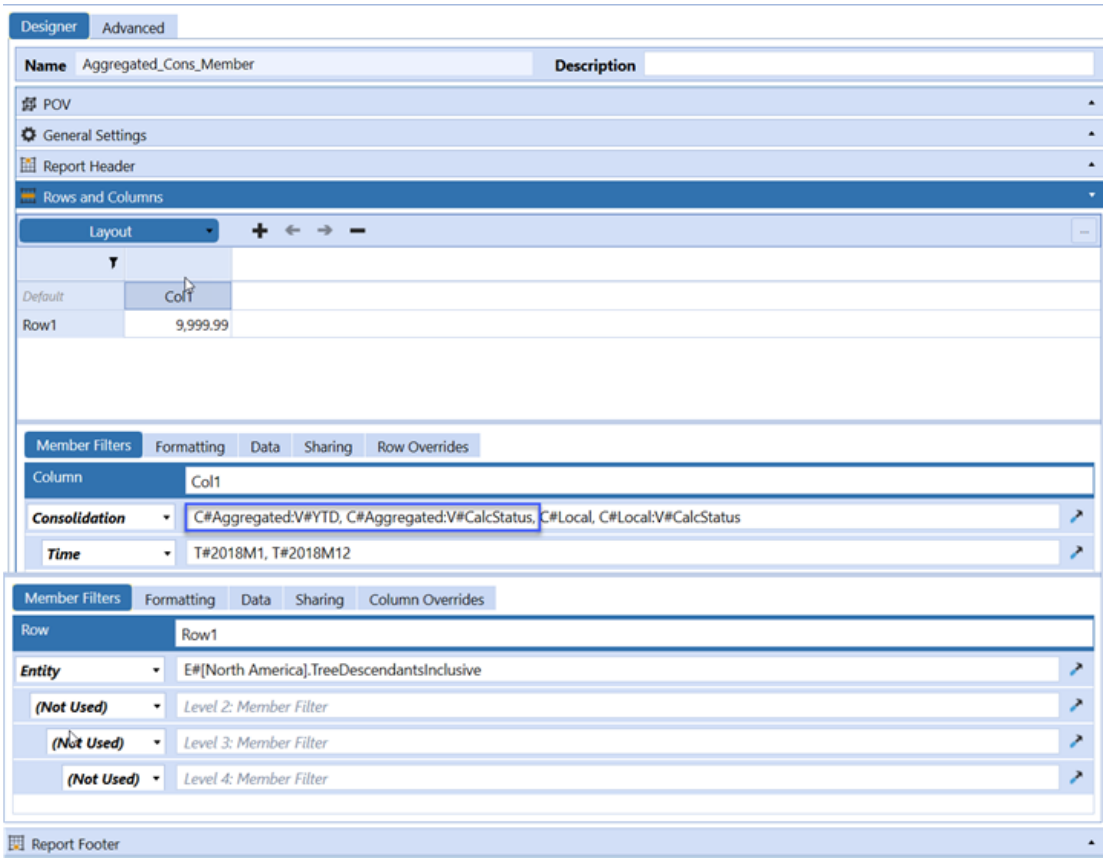
Example of an Entity Aggregation



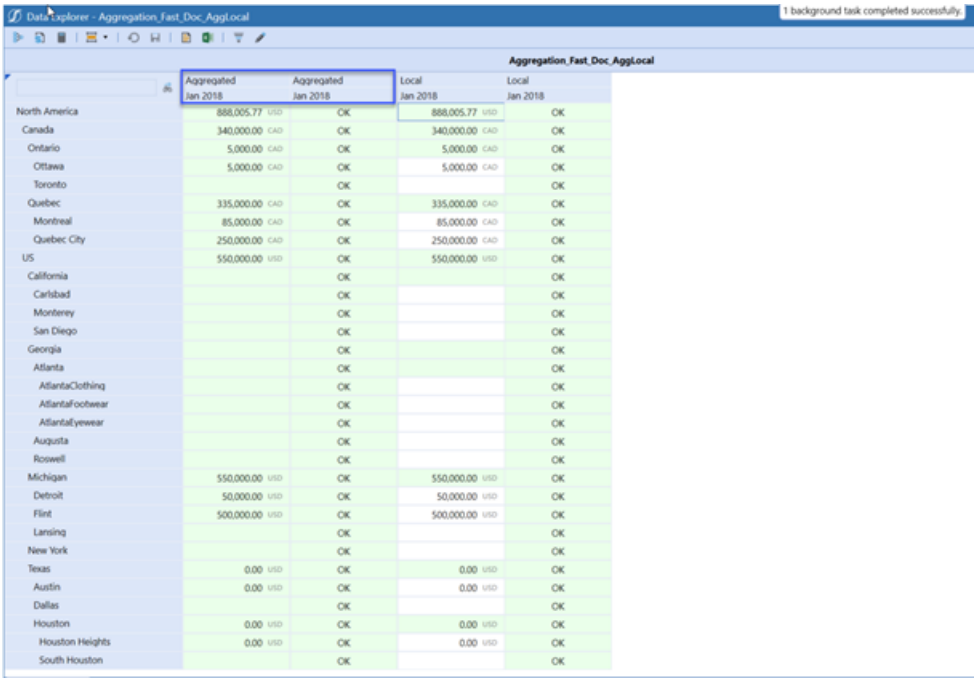
Aggregated Dimension



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 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.

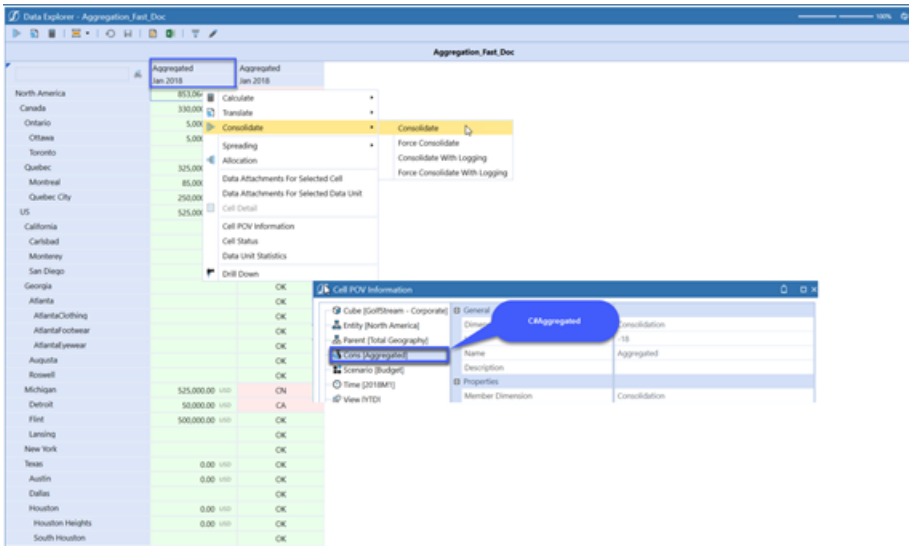
Launching an Aggregation

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 clicking Consolidate.

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 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 versions 7.4 and higher.

Enable Entity Aggregation Input

You must first enable aggregation input on the scenario you are using. Perform the steps below or watch the video.

1. On the Application tab, click **Dimensions**.
2. Expand **Scenario Dimensions**.
3. Expand the appropriate dimension, then select **Scenarios**.
4. On the Members tab, expand **Hierarchy** and click on the aggregated member to select it.
5. Click the **Member Properties** tab.
6. Under Settings, set **Allow Input into the Aggregated Consolidation Member** to TRUE.
7. Click **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

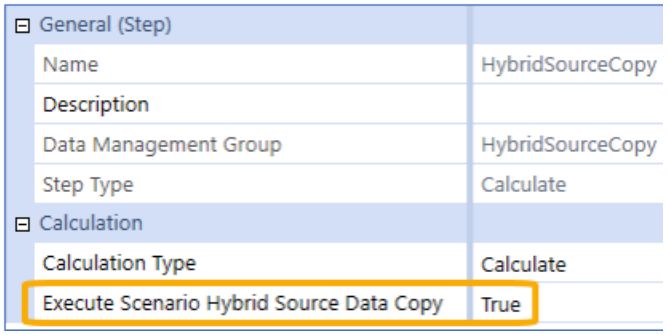
Data bindings determine if data is shared or copied from the source scenario member to the target scenario member. Sharing data is best if you need to analyze smaller data sets from a large data unit.

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 - 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.

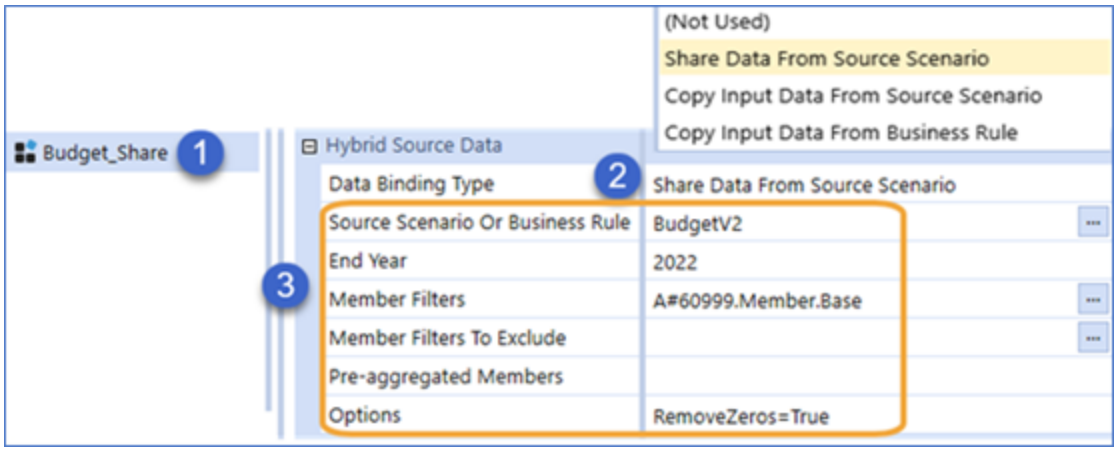
Once the target scenario copy finishes, you can modify and adjust data, but the target scenario is still bound to the source. The data copy occurs whenever the scenario is calculated based on the scenario setup. Hybrid copies can be chained, so results in a target scenario can be used as the source in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

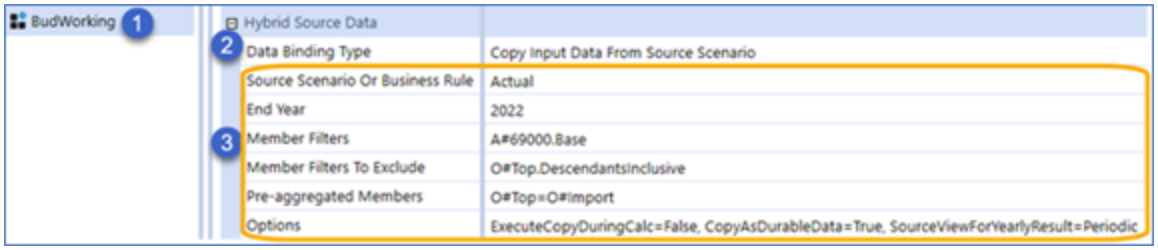
Share Data From the Source Scenario



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.

Navigation

You can access OneStream through a web browser or via OneStream Windows App through an application shortcut deployed via a OneStream website, or from a version installed directly on a computer. These options are available for both administrators and end users.

The layout is intuitive and easy to use. OnePlace will highlight all the major touch points as well as provide information on the navigation tips.

> **TIP:** Hover over any of the icons and a tool tip will temporarily display.

About the OneStream Windows Application

Using the Windows App is helpful because:

- The Windows app automatically updates whenever the application server version is updated.
- You do not need admin rights to download or use the application.
- It offers robust spreadsheet functionality, so you may not need to install the Excel Add-in.

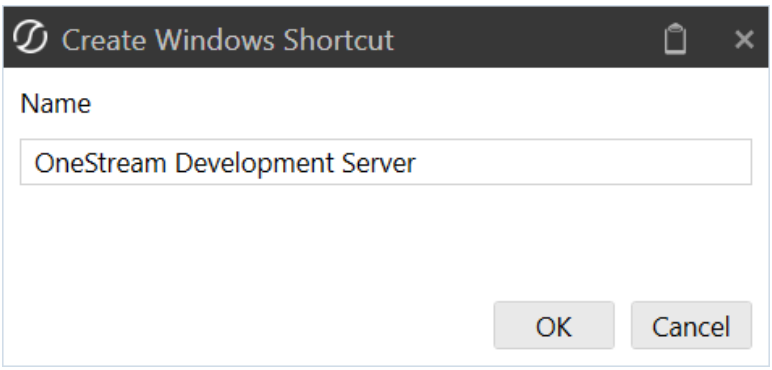
To launch the Windows App:

1. Click the launch icon in the upper right corner of the window from any 4.2 or greater web-based instance.
2. If prompted, click **Run**.

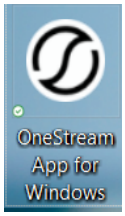
You can also launch the Windows App via Microsoft Edge with the URL associated with the latest version of the server. First, enable Edge Chromium support for Click Once.
See https://docs.microsoft.com/en-us/deployedge/microsoft-edge-policies#clickonceenabled.

To create a desktop shortcut:

1. Click The Create Windows Shortcut window displays.



2. Modify the shortcut name as needed and click **OK**.



This icon can now be used to launch the Windows App from your desktop.

Launching the OneStream Windows App

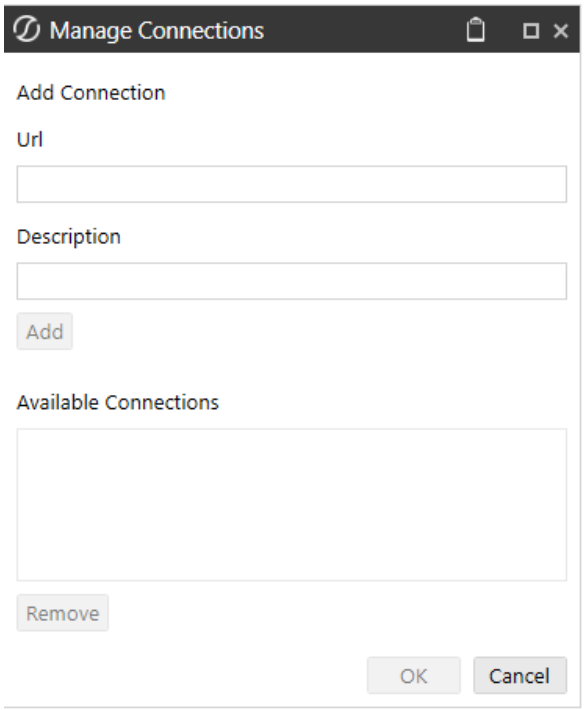
The Windows App is installed locally and can access different servers. To launch the app, click, **Start** and locate the OneStream Desktop app.

Defining Server Connections

Before you can log in and open an application, you must define server connections:

1. On the **Logon** page, click ... in **Server Address** to add the URL of the server to connect to.

   The Manage Connections window displays.



2. Enter the server URL and a description.
3. Click **Add** to include the server in the list of **Available Connections**.
4. In **Available Connections**, select the server and click **OK**.

Logging In

If you are in a OneStream-hosted environment, see the *Identity and Access Management Guide* for information about authentication with OneStream IdentityServer.

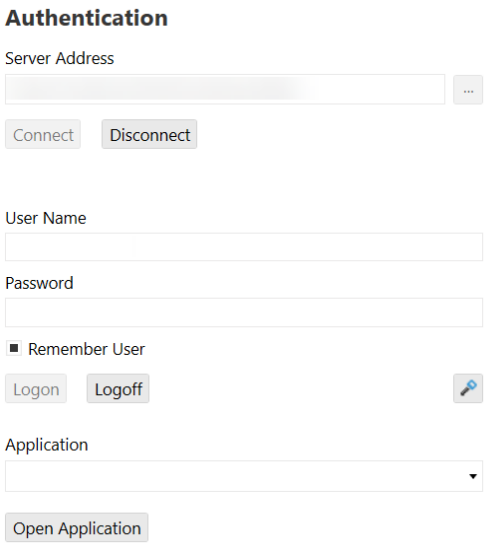
If you are in a self-hosted environment, to log in with the browser, see the *Modern Browser Experience Guide*.

If you are in a self-hosted environment, to log in with the desktop application, follow these steps.

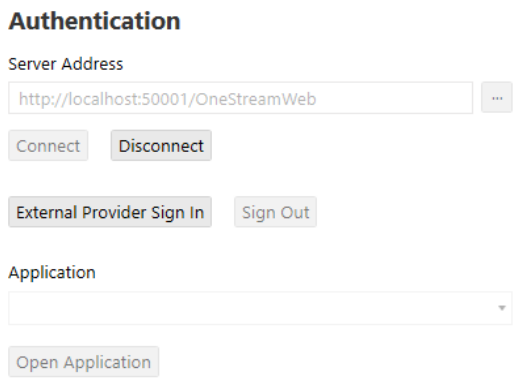
1. On the OneStream **Logon** screen, for **Server Address**, specify the URL or client connection. See Defining Server Connections.
2. Click the **Connect** button.

> **NOTE:** The Logon screen will look different depending on how your environment is configured (native authentication, external identity provider, or both native authentication and an external identity provider).

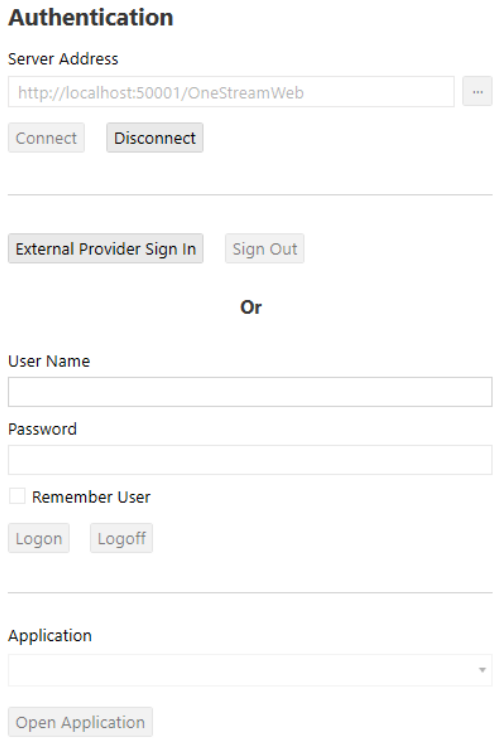
Native Authentication Only



External Identity Provider Only



Native Authentication and an External Identity Provider



3. Follow the steps for your authentication configuration:

- For native authentication:

    a. Enter your user name and password.

    b. Click the **Logon** button.

    > **NOTE:** This process for native authentication is also used for authentication with MSAD and LDAP identity providers. Enter your user name and password set up in MSAD or LDAP and click the **Logon** button.

- For an external identity provider:

    a. Click the **External Provider Sign In** button.

    b. Enter your external identity provider login credentials.

    > **NOTE:** Single sign-on with an external identity provider can be configured to require a one-time verification code for authentication. If the Require Verification Code setting in the Web Server Configuration File is enabled, you will be provided with a one-time verification code to enter in the application. For information about the verification code, see the *Installation and Configuration Guide*.

4. On the OneStream **Logon** screen, select an application from the drop-down menu.
5. Click the **Open Application** button.

For instructions on configuring authentication in a self-hosted environment, see the *Installation and Configuration Guide*.

OnePlace Layout

1. Navigation Pane - This section covers the three tabs that are available: System, Application and OnePlace.

> **NOTE:** Tab visibility is determined by users' security settings.

Each bar can be displayed by pinning it to the screen, or by Auto Hide. Additional details can be found further down in this section.

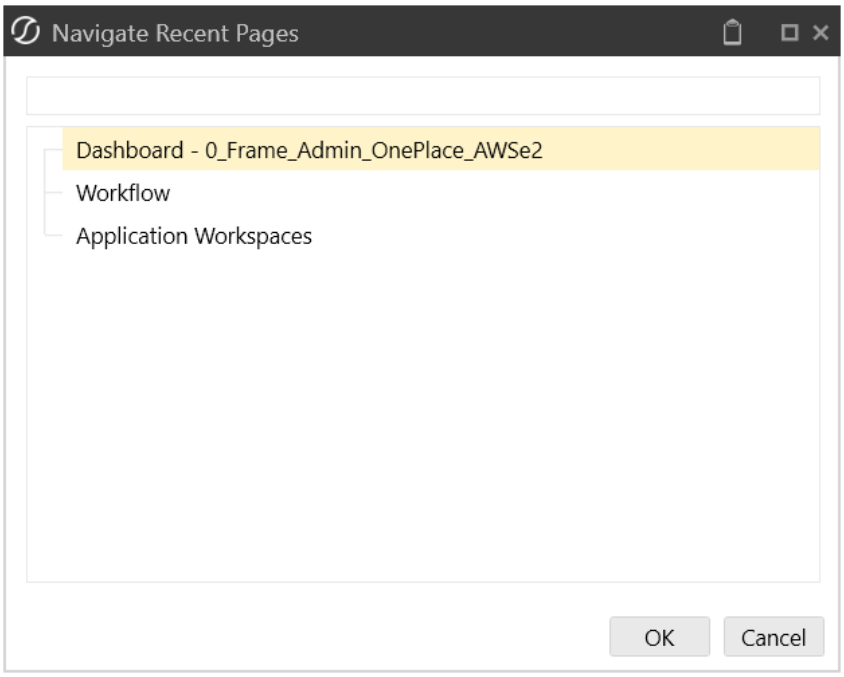
2. Home - Click the large OneStream icon to navigate to the user's set home screen. See Page Setting Options below for more information on setting home screens.
3. Title Bar - This displays the OneStream Logo.
4. Application Tray

Hamburger Menu (Navigation Pane)

Clicking (not hovering) this icon will hide or unhide the Application, System, OnePlace Tabs on the left-hand side of the OnePlace screen. If the Navigate Pane is unpinned, clicking (not hovering) this icon will hide the Application, System, OnePlace Tabs.

Navigate Recent Pages

This opens a navigation dialog allowing the user to select and go back to a recent page.

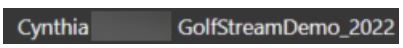


File Explorer

This opens the File Explorer dialog allowing users to access public folders, documents and the File Share.

Environment Name

This is a customized environment name which can be made different across environments (e.g. Development, Test and Production). Specify an environment name and color in the application's application server configuration file. See Installation and Configuration.

 User ID and Application

Logon/Logoff Icon

Upon selecting the Logoff icon, the user will be prompted to End Session or Change Application. End Session logs the user off and removes the saved password from the logon screen. Change Application keeps the user logged in and allows him/her to select a new application from the drop-down screen.

Task Activity

This displays all tasks performed within the application. See Task Activity in Logging for more details on this feature.

Refresh Application

Refreshes the Application and checks the first open tab. If it is an Application tab, the view will change to that tab. If it is not an Application tab, the view will stay on the selected tab but will change the main active tab to Application.

Pin or Unpin Navigation Pane or POV Pane

This will hide or unhide the Application, System, OnePlace Tabs on the left-hand side of the OnePlace screen. In the POV Pane, this will hide or unhide the POV on the right side of the OnePlace screen.

Clipboard

Drag and drop items such as data cells, text, rule scripts to the clipboard in order to reuse them in other areas. Users can store up to ten items on a clipboard.

Create Windows Shortcut

This creates a desktop shortcut for the application. See About the OneStream Windows Application .

Help

This opens OneStream documentation for Platform and OneStream Solution.

Hamburger Menu (POV Pane)

Clicking (not hovering) this icon will hide or unhide the POV on the right side of the OnePlace screen. If the Navigate Pane is unpinned, clicking (not hovering) this icon will hide the POV on the right-hand side of the OnePlace screen.

5. Page Setting Options
   Click the small OneStream icon to use the following page settings:
   a. Refresh Page: This will refresh the current page.
   b. Close Page: This will close the current page.
   c. Create Shortcut: This will create a shortcut for the current page and store it in the user's Favorites Folder. When this shortcut is selected from the user's folder, it will navigate to that page. This can be used for specific Cube Views, Dashboards, or Application/System pages.
   d. Set Current Page as Home Page: This setting controls the default settings for both the page display as well as the pinning of the Navigation and Point-of-View panels when you sign on. Click the OneStream icon in the application tray in order to navigate to the home page from any other screen. This functionality works in both the browser and the OneStream Windows App versions. Changes made here within one environment carry over to the other.

This also controls the Pinning of the Navigation Bar and the POV Bar as well.

This generates a UserAppSettings.XML file that has the following pin options:

```
<SLHomePagePinNavPane>TrueValue</SLHomePagePinNavPane>

<SLHomePagePinPovPane>FalseValue</SLHomePagePinPovPane>
```

These control if the user's navigation bar is pinned by default when logging into the application.

1. Clear Home Page Setting
   This will remove the current home page setting. This functionality works in both the browser and the OneStream Windows App versions. Changes made here within one environment carry over to the other.
2. Save Home Page Setting As Default For New Users
   This will save a home page as the default home page for any new user logging in for the first time. This functionality works in both the browser and the OneStream Windows App versions. Changes made here within one environment carry over to the other.
3. Close All Pages
   This closes all open pages.
4. Close All Pages Except Current Page
   This will close all open pages except the one currently displayed.
5. Workflow Bar
   This section displays exactly where the user is in the Workflow process. Based on the Workflow Profile, this can be configured as a Certifier or as a Data Loader. The example above is configured as a Data Loader on the Validate task of the Workflow. The color green indicates a completed task, blue indicates incomplete tasks. The white OneStream icon indicates what task is currently in view.
6. Page Refresh
   This section covers the local refresh and the ability to close a page.

Refresh Page

This refreshes the active page.

Toggle Page Size

This changes the page size in order to fit the screen.

Close Page

This closes the active page.

7. Toolbar
   Similar to the Workflow bar, this displays the items for a Data Loader or for a Reviewer / Certifier.
8. Context Pane
   This bar is where the Point of View is set. This is an important concept because it determines to which Dimensions the users will have access for this application view. Additional details can be found further down in this section.
9. Grid
   This displays the active window's contents for the functionality being executed.
   - Pages: This allows the ability to navigate through the pages either by choosing the page, by clicking on the next page, or the first / last page.
   - Tabs: Now that the tab is open, simply click on one of them to save time in navigating. Clicking the New tab allows two or more like tabs to be opened. For example, two Cube Views tabs can be opened at once.

10. Pages

> **NOTE:** Right-click on any of these opened tabs for more page setting options.

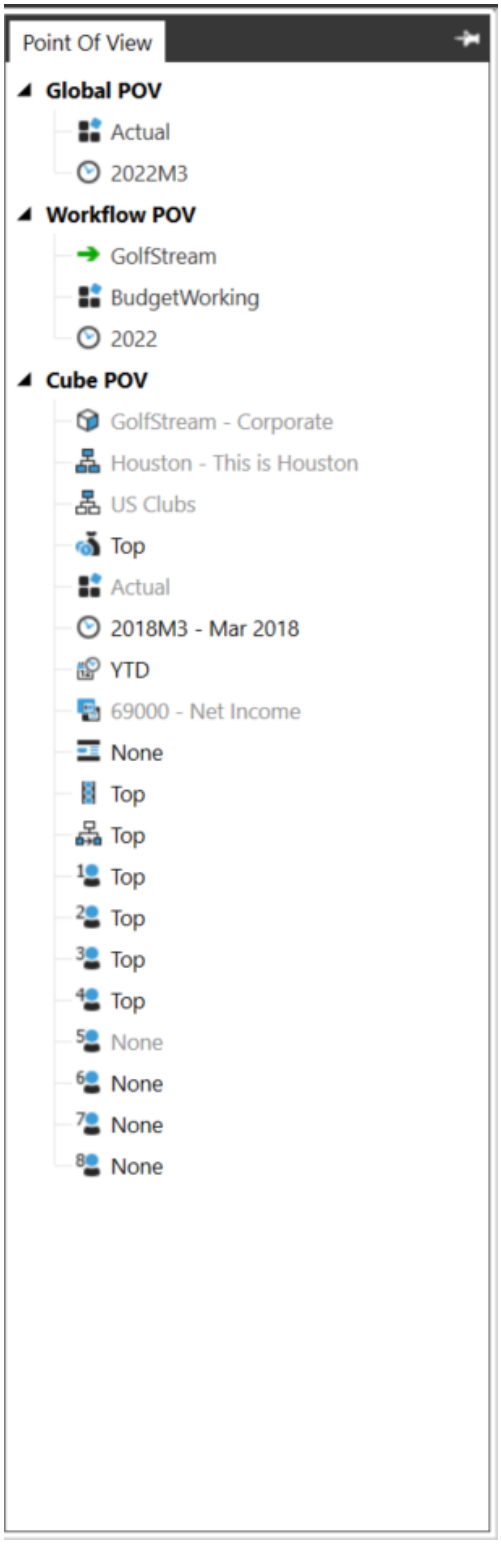
The following icons are only located in the OneStream Windows App.

Zoom Options

This controls the zoom settings when working in the OneStream Windows App.

Point of View

The Point of View panels located on the right side of the application. This tab can be docked by clicking the pin button, otherwise, it will disappear when clicking anywhere in the main page.


Point Of View
Global POV
Actual
2022M3
Workflow POV
GolfStream
BudgetWorking
2022
Cube POV
GolfStream - Corporate
Houston - This is Houston
US Clubs
Top
Actual
2018M3 - Mar 2018
YTD
69000 - Net Income
None
Top
Top
Top
Top
Top
Top
None
None
None
None

There are three primary sections defined under the Point of View.

Global POV

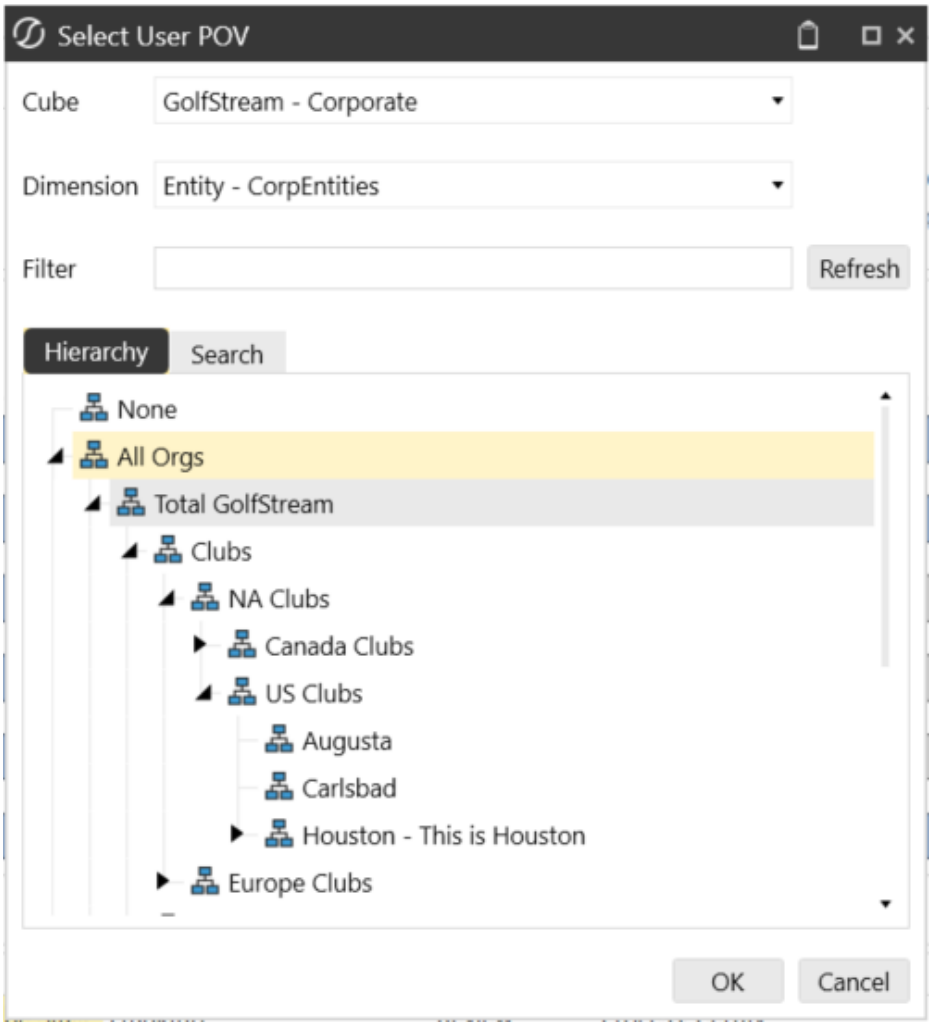
This is the Point of View for the whole application. This is set by the administrator and will not be active for the end user to update. This includes Scenario and Time.

Workflow POV

This has the same configuration as the Workflow area of the OnePlace Tab. It will display the active Point of View; however it will not be active for the end user to update. This includes Workflow, Scenario, and Time. The Time displayed is based on the Time Dimension Profile associated with the Cube assigned to the Workflow Profile.

Cube POV

This is active and available to be updated by the end user. Each Dimension will need to be set based on the information or activity a user needs to perform. Hover over any of the Dimensions and a tool tip will display the Dimension type. To update a Dimension, select one and a Select User POV box will appear. This box will give the user an opportunity to pick the Cube, Dimension, and the ability to apply a Member Filter and search.

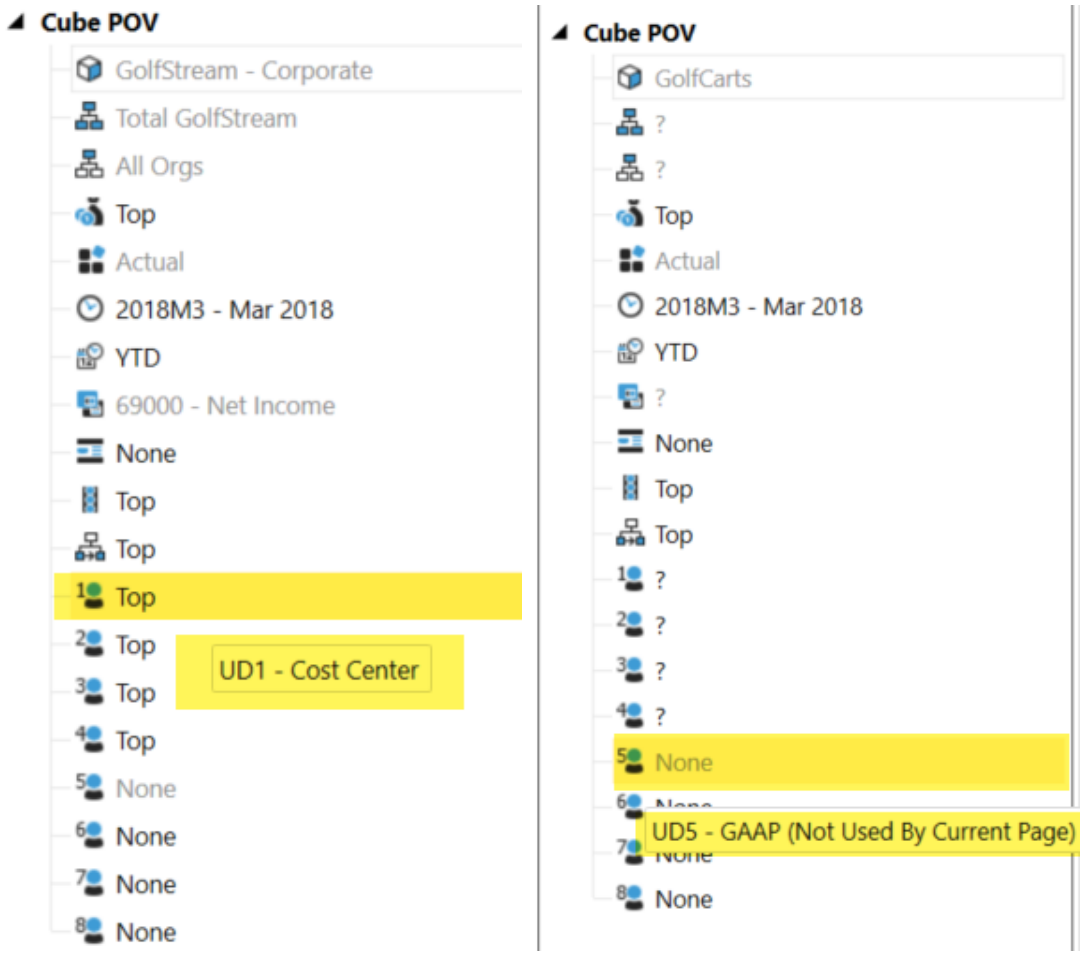


TIP: Save a commonly used POV by right clicking on the Cube POV and selecting Save Cube POV to Favorites. This saves the POV under **Application**|Documents|Users >|(User Name)|Favorites allowing it to be used on any Cube View, grid, or Dashboard.

User Defined Description – Point of View

The Point -of -View panel supports the new User Defined Descriptions. Hovering over a selectable point-of-view member will display the defined description. Dimensions which are fixed, not selectable, will display the defined description and append “Not Used by Current Page”. See Application Properties and then User Defined Dimensions (Descriptions).

The Point-of-View panel supports the new User Defined Descriptions. Hovering over a selectable point-of-view member will display the defined description. Dimensions which are fixed, not selectable, will display the defined description and append “Not Used by Current Page”. See Application Properties and then User Defined Dimensions (Descriptions).


Cube POV
GolfStream - Corporate
Total GolfStream
All Orgs
Top
Actual
2018M3 - Mar 2018
YTD
69000 - Net Income
None
Top
Top
Top
Top
Top
Top
None
None
None
None
UD1 - Cost Center
Cube POV
GolfCarts
?
?
Top
Actual
2018M3 - Mar 2018
YTD
?
None
Top
Top
?
?
?
?
None
None
UD5 - GAAP (Not Used By Current Page)

Docking

Docking improves your experience in OneStream by enabling you to click, drag, and drop OneStream tabs into specific areas.

Benefits

You can use docking in several ways:

- You can click and drag one tab to a separate screen or monitor while keeping other tabs open on the original screen.
- You can dock pages side-by-side and make changes as an administrator in one page and quickly see the results in the other page upon refresh.
- You can dock multiple pages within existing docking areas to create your own individualized layouts.

Docking provides the following benefits for viewing, building, editing, and creating in the OneStream application:

Task	Role	Benefits
Building a Dashboard	Application Developer	Instead of navigating between tabs in the application, you can dock multiple tabs and have them each display simultaneously.

Task	Role	Benefits
Data Analysis	End User	You can dock multiple Cube Views on several screens . For example, Window 1 has Income Statement docked, Window 2 has Revenue docked, and Window 3 has Expenses docked. This allows you to manage multiple items simultaneously.
Editing a Cube View	End User	You can view and edit Cube Views more easily with Data Explorer and Designer side-by-side docked.

See Docking Use Cases.

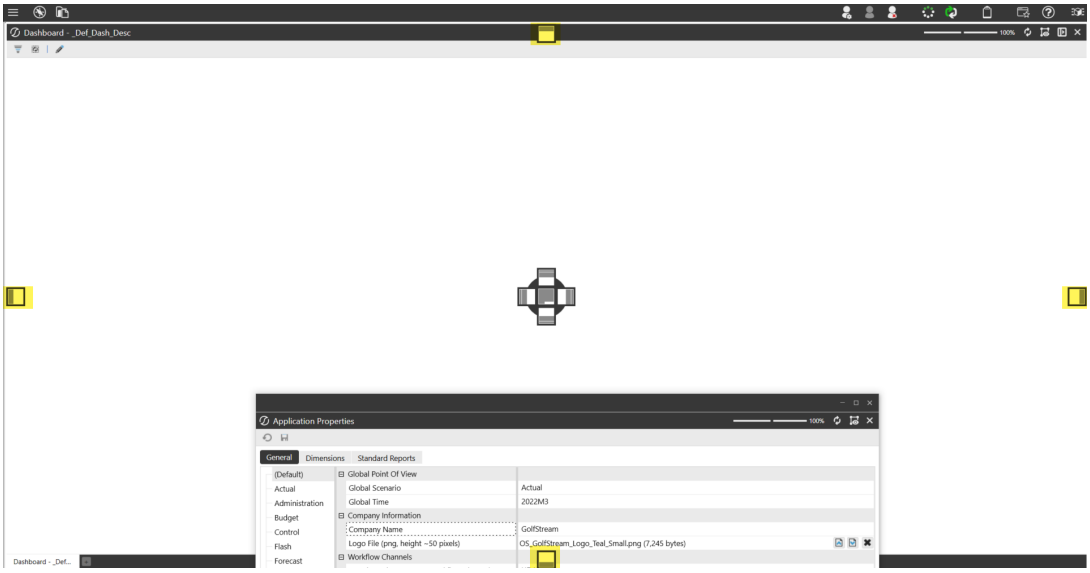
Docking Areas

You can create new docking areas and gain the ability to have tabs open simultaneously and multitask by clicking with the left mouse button, dragging, and then dropping tabs into specific docking areas.

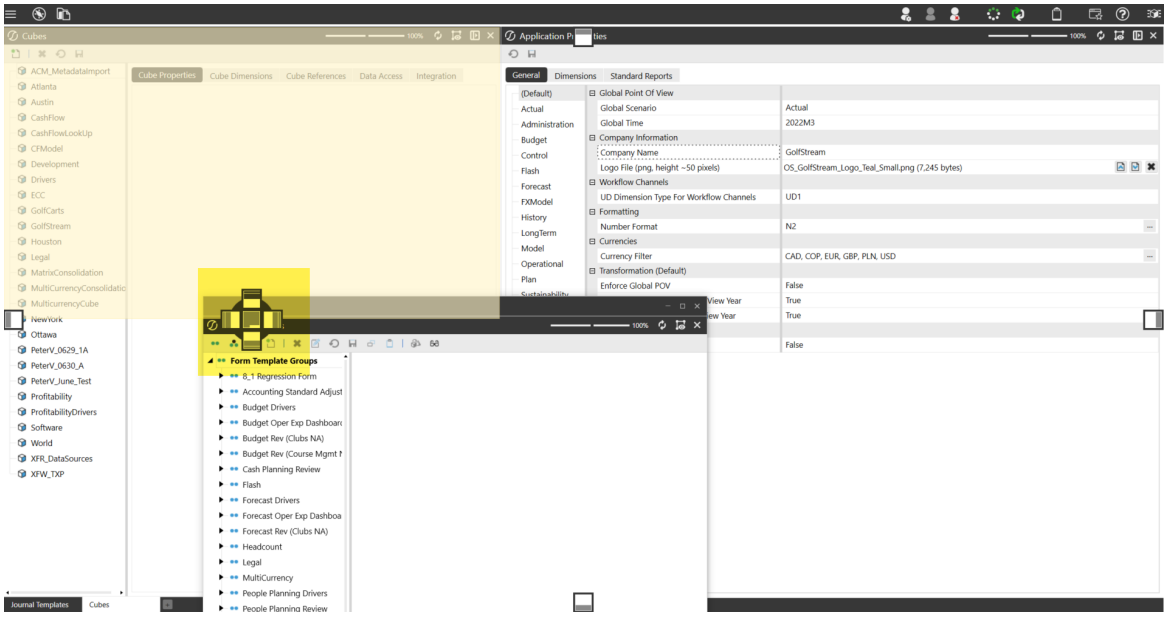
> **NOTE:** The Toggle Page Size icon will not display in a floating window.

Layouts

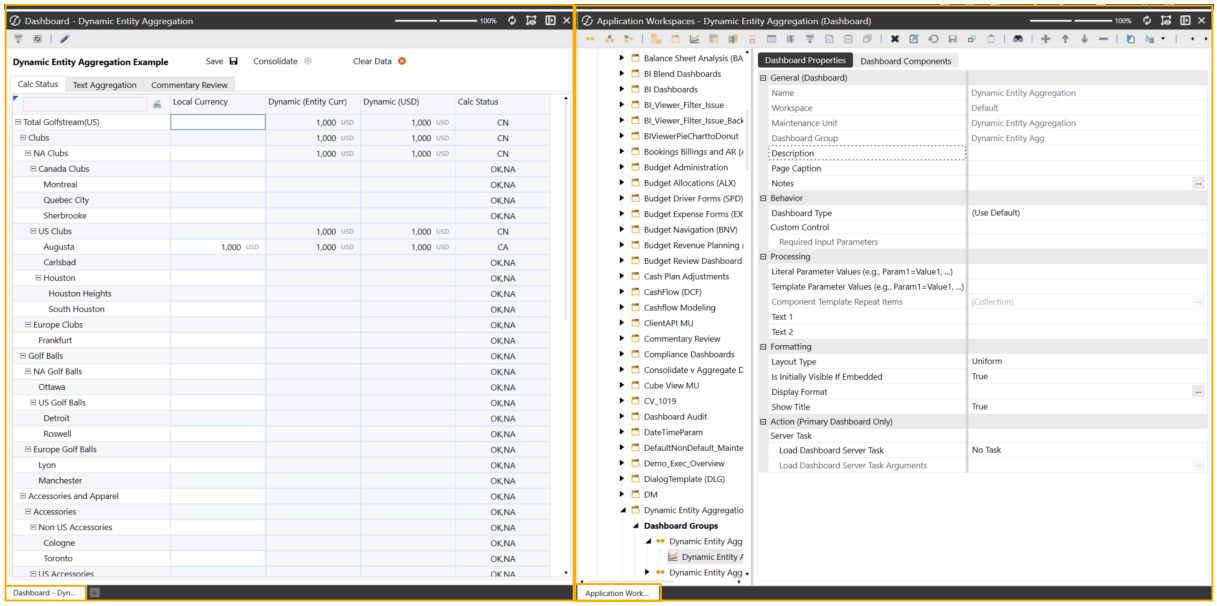
Each docking area allows you to move content in the following layouts: Left, Right, Top, and Bottom. This will move content to the layout direction of the document group.



There are also inner components you can move content into for each individual window. You can move content into the following inner layouts: Left, Right, Top, Bottom, and Center. The Center layout opens content within the document group.



In the following example, the Application Workspaces page was docked to the right to edit the dashboard.



Tabs

New Tabs

Click the **New Tab** button in the bottom tab bar to open a new (Empty) tab. When you open a new page through the left-hand Navigation pane, the new tab will replace the (Empty) tab in the main window.

If actions or buttons in a specific OneStream page can launch a new page, those new pages will be opened in the same window as the original OneStream page.

> **Example:** If the Application Workspaces page is open in a floating window and you select a Cube View and then click the Open Data Explorer toolbar button, the new Data Explorer page will be opened in the same floating window.

Empty Tabs

If you click the **New Tab** button to add a new (Empty) tab and then select a navigation link in the left-hand Navigation pane, it will open a new page and replace the empty tab.

> **Example:** If you click the **New Tab** button to create an (Empty) tab, then click the Books navigation link, the system will replace the (Empty) tab with the Books tab.

> **IMPORTANT:** You cannot dock the (Empty) tab in a docked or floating window. The (Empty) tab only exists in the main window.

When the last open tab in the main window is dragged away, there will be an (Empty) tab and the New Tab button at the bottom of the main window.

> **Example:** If the Books tab is dragged out of the main window and added to a Cube Views docked window, the (Empty) tab and New Tab button will display in the main window.

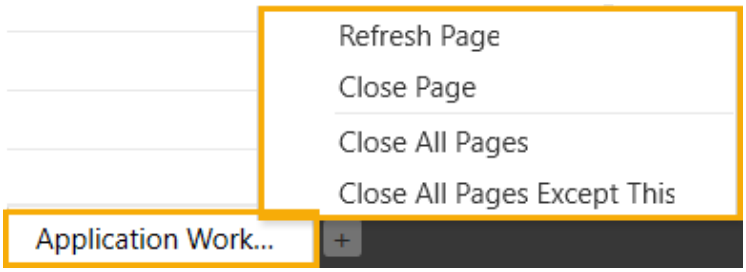
> **NOTE:** The **New Tab** button will always display in the main window.

Context Menu

When right-clicking on a tab or a docked page, you will have four options.

Context Menu Options

- Refresh Page
- Close Page
- Close All Pages
- Close All Pages Except This

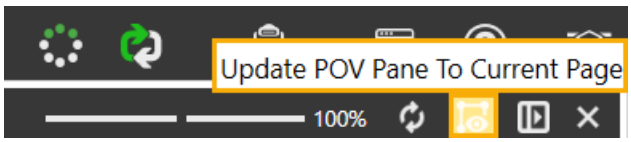


Pane

POV Pane

The **Update POV Pane To Current Page** toolbar button displays at the top-right of the main document group and any page in docked and floating windows. Items in the POV Pane may be grayed out due to multiple active pages or tabs. This toolbar button will not display in the main document group, if no other docking or floating windows are open.

To activate the POV on the docked, floating window or main document group content, select the **Update POV Pane To Current Page** toolbar button. When you hover over the toolbar button, it will highlight in yellow as an active icon.



When docked pages or floating windows are closed or re-added to the main window, the POV returns to its previous functionality and the toolbar button will not display. If you do not have any docked or floating windows, the existing functionality on the right POV pane applies.

> **Example:** When selecting a tab to open that specific page in the main window, the POV pane will update to that page's POV.

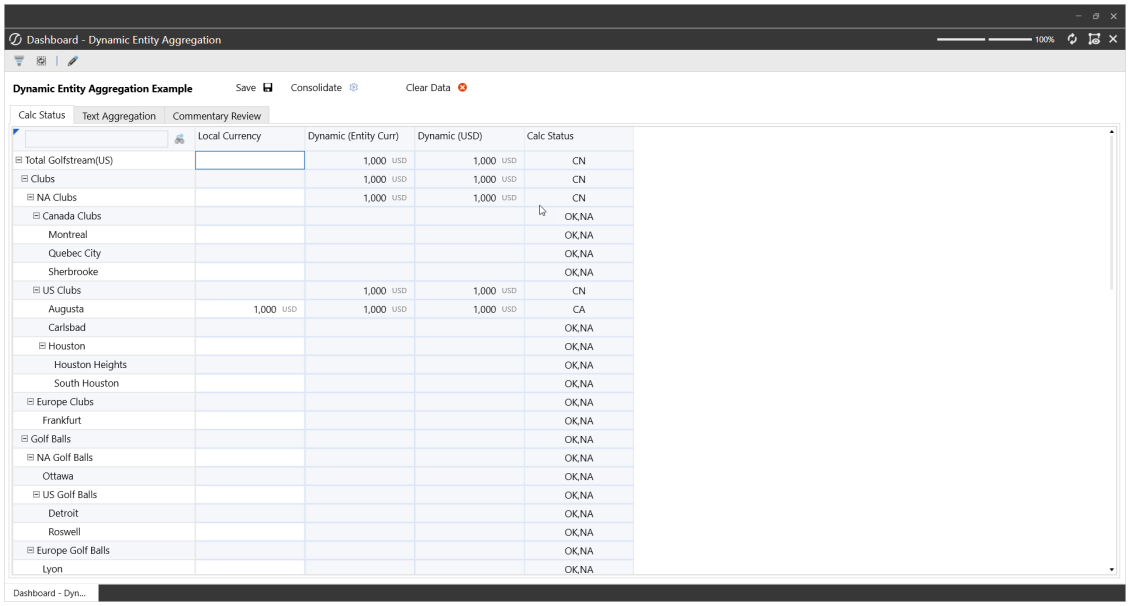
Left Navigation Pane

The Left Navigation Pane's OnePlace, Application, and System tabs will no longer be automatically selected depending on the page most recently opened in the main, docking, or floating window.

Floating Window

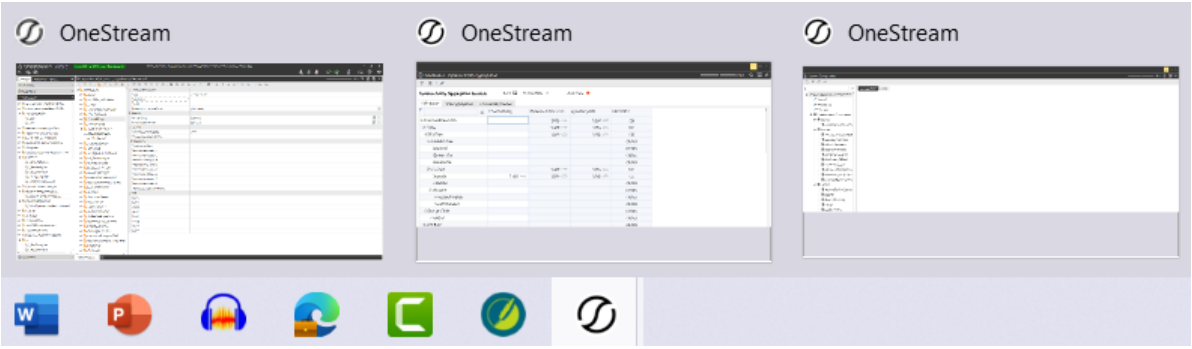
You can create floating windows by clicking, dragging, and then dropping a tab into a screen or monitor. You can also drag and drop other tabs and pages onto the floating window. If the floating window has multiple tabs or pages, clicking the top-right **Close Window** button will close the floating window and all pages contained within it.

> **NOTE:** The Toggle Page Size icon will not display in a floating window.



> **NOTE:** With the docking functionality, the floating windows maintain the same functionality.

Hover over the OneStream application in your taskbar to display different application floating windows in the foreground.



Refreshing Content

If a docked, floating, or the main window contains multiple tabs or pages, clicking the **Refresh** button at the top-right of an individual, open page only refreshes the content within that one page.

Undock Content

You can undock a tab or page in the following ways:

- You can click, drag, and drop the tab or page into a new location. For example, click and drag a tab or page from a floating window and back into the main window.

  > **NOTE:** If the last tab or page is dragged away or closed within a docked or floating window, that docked or floating window will also close.

- You can drop a tab or page back into the main window tab bar or over the **New Tab** button. This will dock the content into the main window.
- You can use the center docking component that displays when dragging tabs or pages into the specific window, which will open the tab or page in the selected document group rather than opening a new docking area.

Docking Known Issues

Consider the following when docking in OneStream.

Logon Page

- The Close Page button will always appear on the Logon page. For example, when you log off, select Change Application and then click the Close Page button in the top right corner of the Logon page. A blank screen will display. You must restart the application to log in again.

POV Pane

- The **Update POV Pane To Current Page** toolbar button appears in every docked or floating window even if the docked or floating window does not reference the POV pane settings. The **Update POV Pane To Current Page** toolbar button will not appear in the original, main window. You must create at least one additional docking or floating window for this toolbar button to appear.

Floating Window

- When you minimize and attempt to re-open a floating window on a separate monitor or screen, the floating window opens on the main application screen. You will need to drag the floating window back to the separate monitor or screen every time you re-open that minimized window.
- If you have a workflow page in a minimized floating window and then try to launch another workflow page from the main application page, the minimized floating window will display a black flash before loading.

Navigate Recent Pages

- When navigating recent pages, docking will consider tabs that you first created or if the page is refreshed or modified in the main document group for docked and floating windows. The back and forward have also been removed.
- In the Navigate Recent Pages dialog, when the same page is opened multiple times, such as a Cube Views page opened in the main window, a docked window, or a floating window, the Navigate Recent Pages dialog will only list the name "Cube View" three separate times. It does not specify which Cube View page (main, docking, or floating) each item references.
- When using the Navigate Recent Pages dialog box to open a tab in a maximized floating window, docking will restore the floating window size (not maximized).
- In the Navigate Recent Pages dialog box, the **OK** button is enabled even when an item is not selected. Click the **OK** button when no option is selected to close the dialog box.

List Box

- A list box may display when you open multiple tabs or pages and click the left most tab and then right-click a tab closer to the right end of the tab bar. The list box will display a drop-down list with all currently opened tabs instead of the context menu.

Docking Use Cases

The following sections provide sample use cases for docking to seamlessly create, multitask, and easily build in your OneStream. In many cases, you must toggle between various tabs and pages since only one window could be opened at a time when working in the application. Docking allows you to effectively view work side-by-side or create floating windows simultaneously. You can make changes on one screen and display the results on another screen when you save content and refresh.

Role	Benefits
End User: User that creates and edits Cube Views.	End Users can manage Cube Views more easily with Data Explorer and Designer displayed side-by-side.
End User: User that creates and edits dashboards.	End users can efficiently modify dashboards and their components by opening the Designer View and Workspaces pages side-by-side or in a floating window. This allows you to make quick changes without having to reopen different pages or tabs to find objects needed for a dashboard.
Administrator that needs to debug Business Rules.	Administrators can dock multiples pages on one screen. This will allow you to run end user actions, show the business rule, and view the error log by docking these three pages in their own individual docking areas on the same or separate screens.
Administrators that use Analysis and Planning	Administrators can dock multiple pages in a side-by-side view to view multiple analysis and input forms for planning purposes.

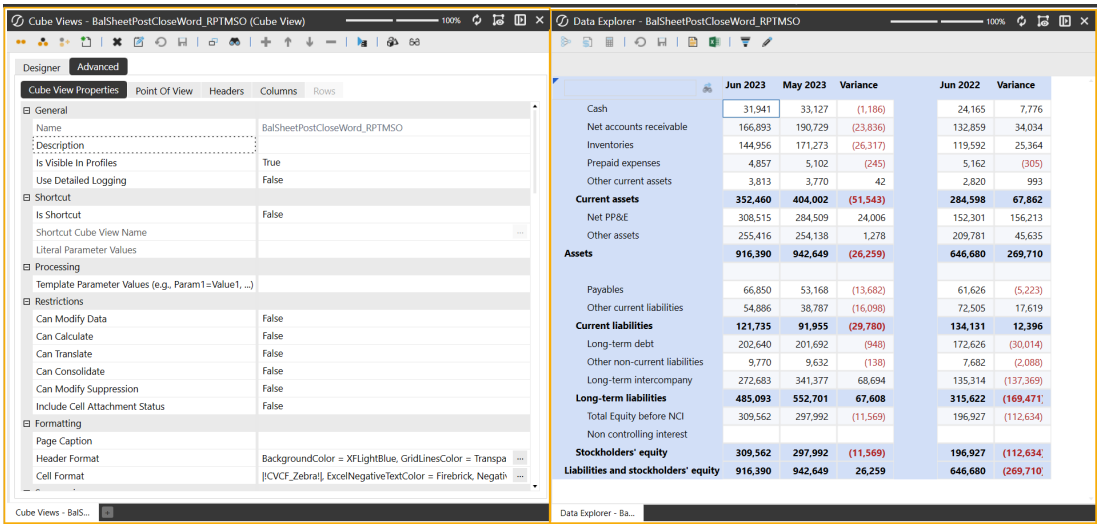
USE CASE

Side-by-Side Cube View Editing

Role	Benefits
End User: User that creates and edits Cube Views.	End Users can manage Cube Views more easily with Data Explorer and Designer displayed side-by-side.

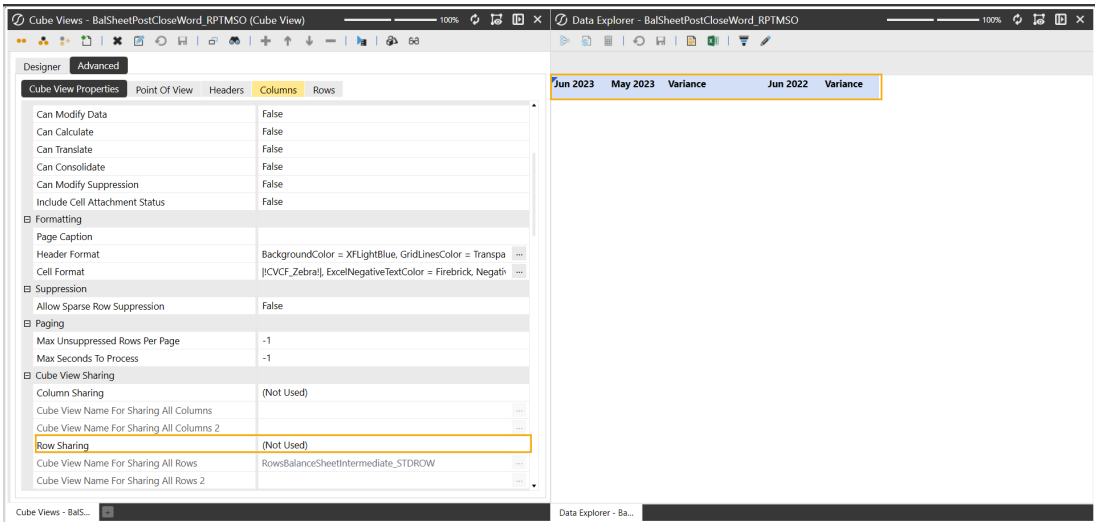
In this example, you can dock tabs or pages while editing the Cube View.

1. On the **Application** tab, under **Presentation**, click **Cube Views**.
2. Select the Cube View to edit and then click the **Open Data Explorer** toolbar button to open the Data Explorer page in another tab.
3. Click, drag, and drop the **Data Explorer** tab to any of following layouts: Left, Right, Top, or Bottom. In this example, the tab was moved to the right layout. The Cube View and the published version are now side-by-side.



4. On the left layout, begin to edit your Cube View. In this example, the Cube View's Row Sharing field was updated from All Rows to (Not Used).
5. Click **Save** to save all changes.
6. On the right layout, click the **Refresh** icon. The Cube View will refresh and display the

changes made.



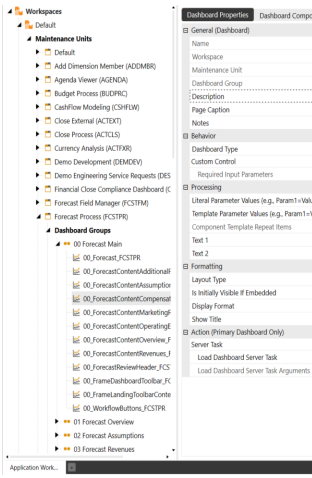
Modify Dashboards

In this example, you are creating a floating window to modify multiple dashboard components.

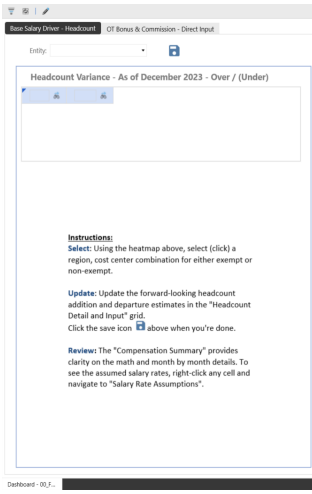
1. On the **OnePlace** tab, under **Dashboard**, select a dashboard.
2. Click the **Edit Dashboard** icon.
3. Once the Application Workspaces page loads and the Dashboard is selected, select the drop-down arrow next to the View Dashboard in Design Mode button, and select the **Set Selected Dashboard as Default**.
4. Click and drag the dashboard in Designer Mode to another screen. In this example, you created a floating window on the right monitor while having the Workspaces page on the left monitor.
5. To edit the MarginLeft value, on the right floating window, click the **Edit Dashboard** button to refresh and load the Dashboard on the left floating window.

6. Navigate to the Formatting Dashboard Properties section, and click the **Edit** button in the Display Format field. In Display format, update the MarginLeft field's value from 50 to 100.
7. Click **Save** to save all changes.
8. On the right floating window, click the **Refresh** icon. The Dashboard now displays the changes.
9. To edit the order of Dashboard Components, on the right floating window, click the **Edit Dashboard** button.
10. Select the Dashboard Components tab and move the order. In this example, the Embedded 04 Component was moved above the Embedded 04 Headcount Component.
11. Click **Save** to save all changes.
12. On the right floating window, click the **Refresh** icon. The Dashboard now displays the changes.

Left Monitor

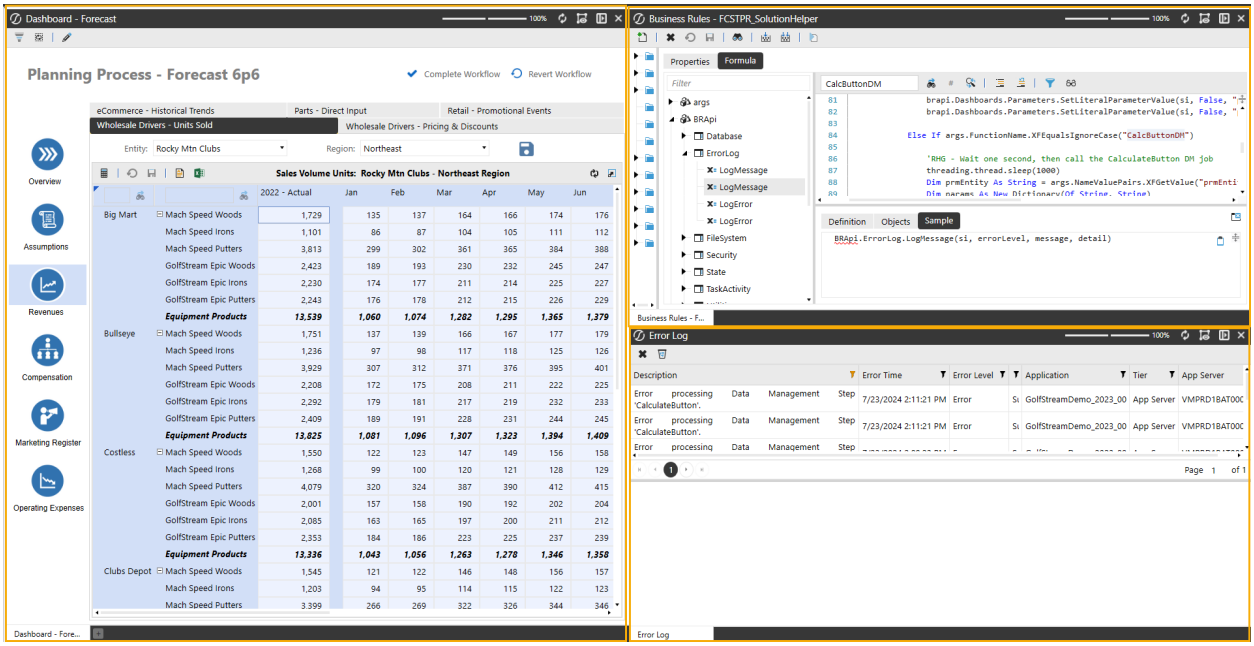


Right Monitor



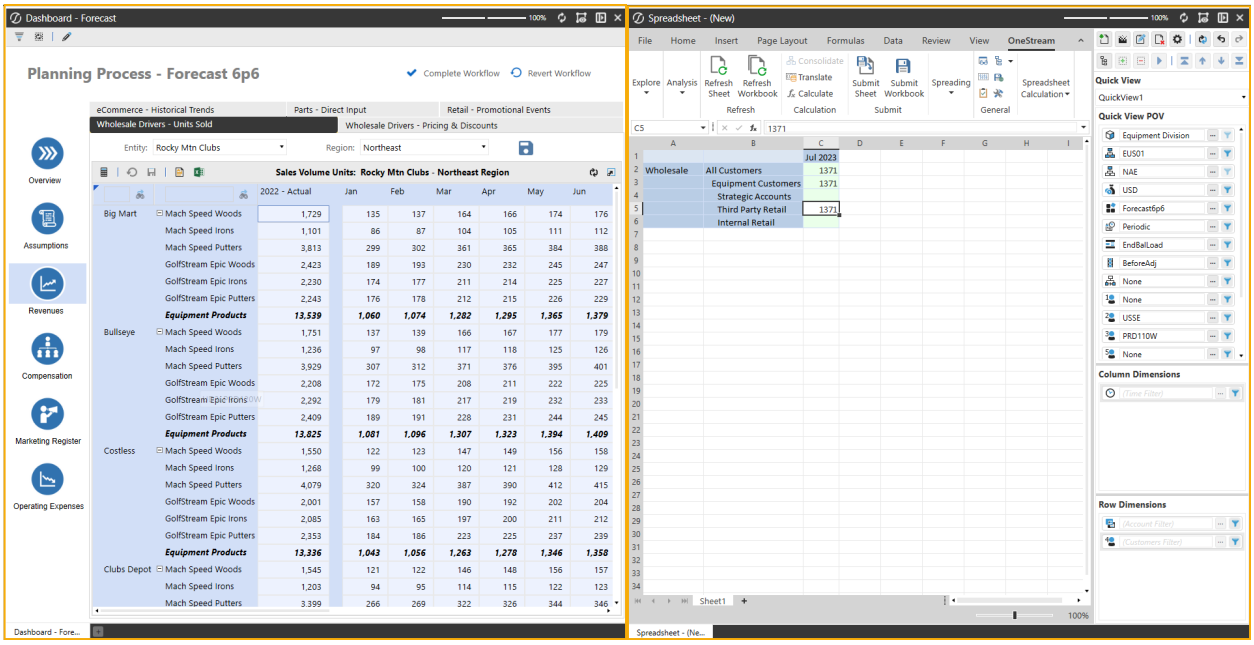
Debugging Rules

In this example, you can run end-user actions, show business rules, and view the error log simultaneously on one screen. You have docked multiple pages such as the "Forecast Planning Process" Dashboard, your Business Rules at the top-right layout, and the Error Log at the bottom-right layout. This allows you to view your dashboard, the syntax being used in a Business Rule, and the log of errors from users. When clicking the Save Calculate icon in your dashboard and then refreshing content on the right layout pages, you can view all the errors and begin to debug your business rules.

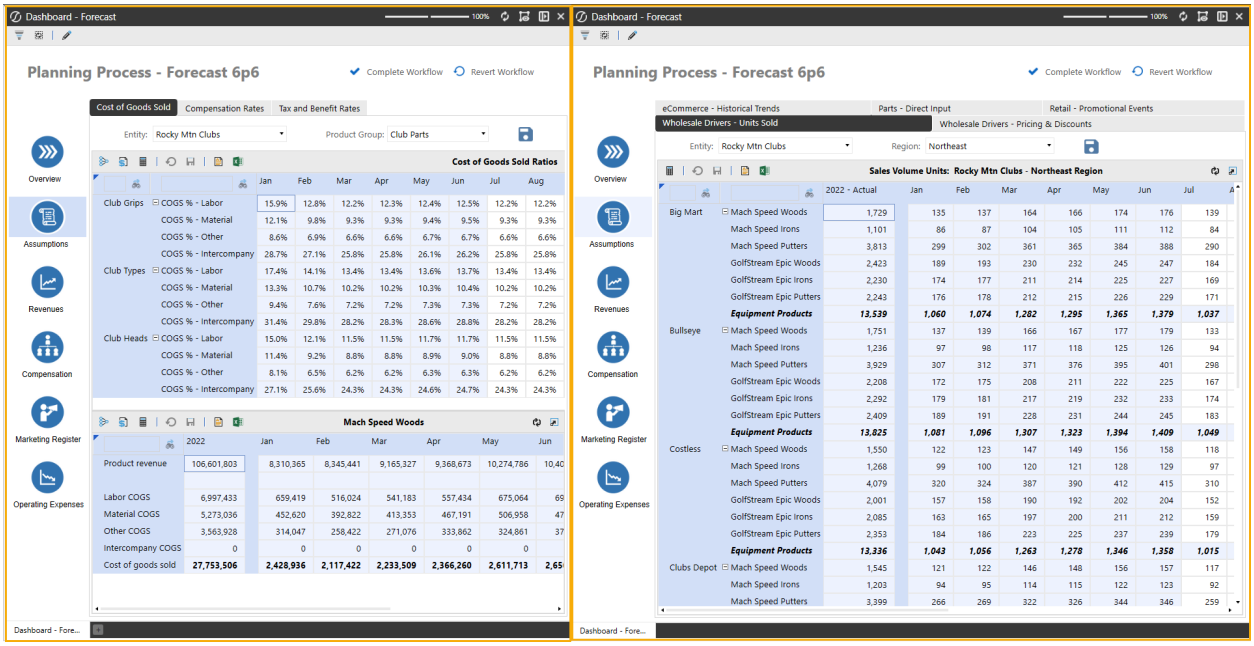


Simplified Analysis and Planning

In this example, you can dock content side-by-side to view multiple analysis and input forms for planning purposes. From an analysis perspective, if you need to analyze data in a quick view while preserving your original view, you can begin to dock both views side-by-side. On the right layout, you can begin ad hoc analysis while still maintaining the context of the dataset you are analyzing on the left layout.



Another example of using docking to simplify Analysis and Planning is to view your drivers and inputs for your forecast and assumptions simultaneously. In your Forecast Dashboard, open the Forecast and Assumptions page on separate tabs. Begin to dock the Assumptions page to the right layout with the Forecast page on the left layout. This allows you to display multiple drivers, such as the Cost of Goods sold driver and the Forecast input, side-by-side. When using docking for this instance, you can view multiple drivers you are interacting with while running your calculations. This allows you to view multiple end-user screens and inputs together.


Dashboard - Forecast
Planning Process - Forecast 6p6
Complete Workflow
Revert Workflow
Overview
Assumptions
Revenues
Compensation
Marketing Register
Operating Expenses
Cost of Goods Sold
Compensation Rates
Tax and Benefit Rates
Entity: Rocky Mtn Clubs
Product Group: Club Parts
Cost of Goods Sold Ratios
Mach Speed Woods
Dashboard - Forecast
Planning Process - Forecast 6p6
Complete Workflow
Revert Workflow
eCommerce - Historical Trends
Parts - Direct Input
Retail - Promotional Events
Wholesale Drivers - Units Sold
Wholesale Drivers - Pricing & Discounts
Entity: Rocky Mtn Clubs
Region: Northeast
Sales Volume Units: Rocky Mtn Clubs - Northeast Region

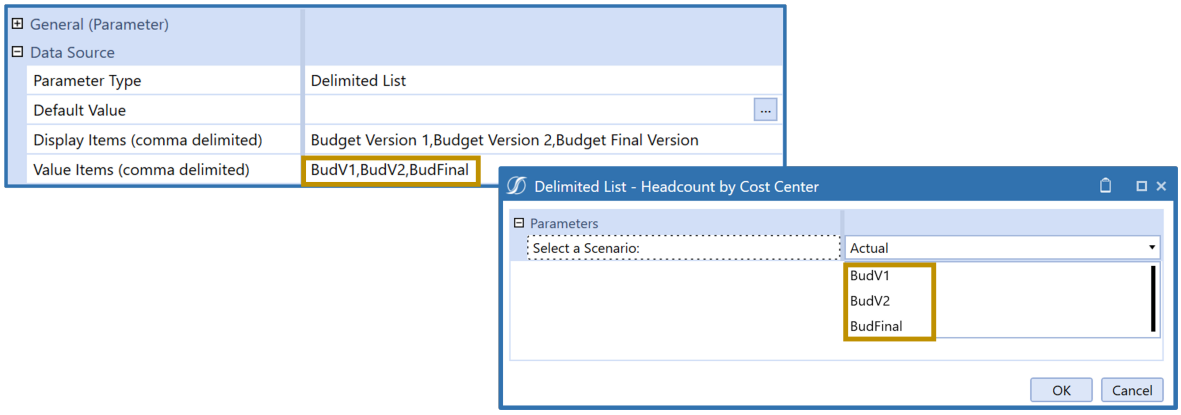
Filter Runtime Data and Reduce Maintenance with Parameters

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

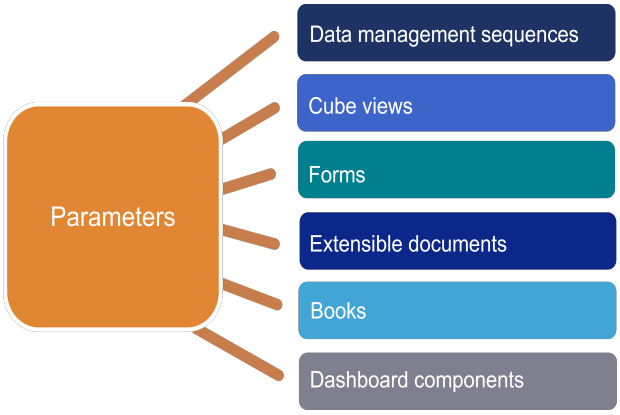
About Parameters

Depending on the type, parameters prompt you to specify data by entering text or selecting items from a drop-down list 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 new parameter, then specify the Literal Value parameter 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.

GolfStream™

Product Sales

Report Run By: Gemma Daly

	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 Input Value parameter settings.

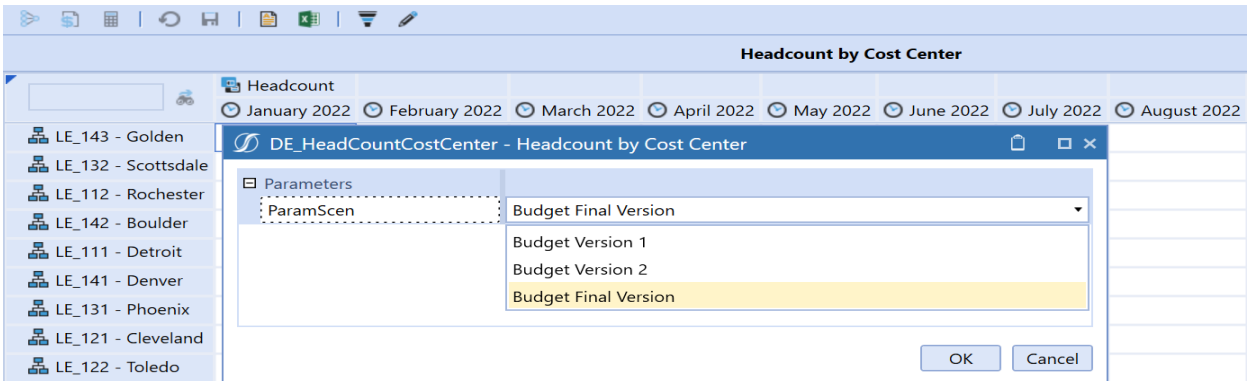
Delimited List Parameters

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 list 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, web browsers, 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 list containing 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 Delimited List parameter properties. See a sample.

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.

To create a Bound List parameter, first Create a Parameter, then specify the Bound List parameter 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 Parameter, then specify the Member List parameter settings.

Member Dialog Parameters

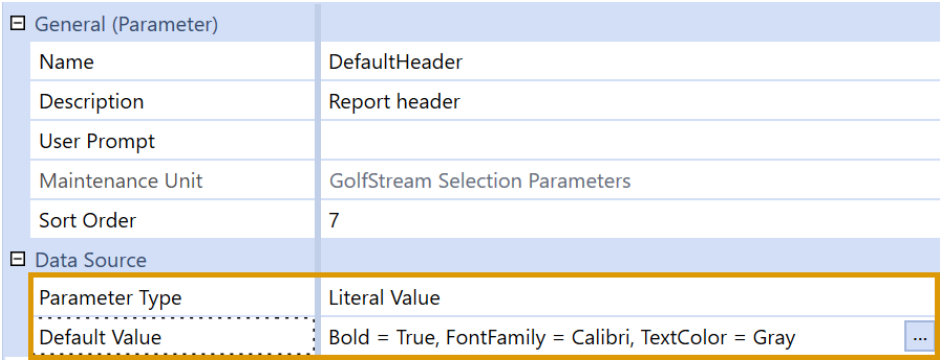
Similar to Member List parameter, the Member Dialog parameter lets you associate a member with the parameter through a dialog box with search capabilities. This is more appropriate for a dimension such as accounts or entities where you can use the hierarchy to select a base or parent member.

To create a Member Dialog parameter, first Create a Parameter, then specify the Member Dialog parameter 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.
- 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 Parameter Components in Dashboards.

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.

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

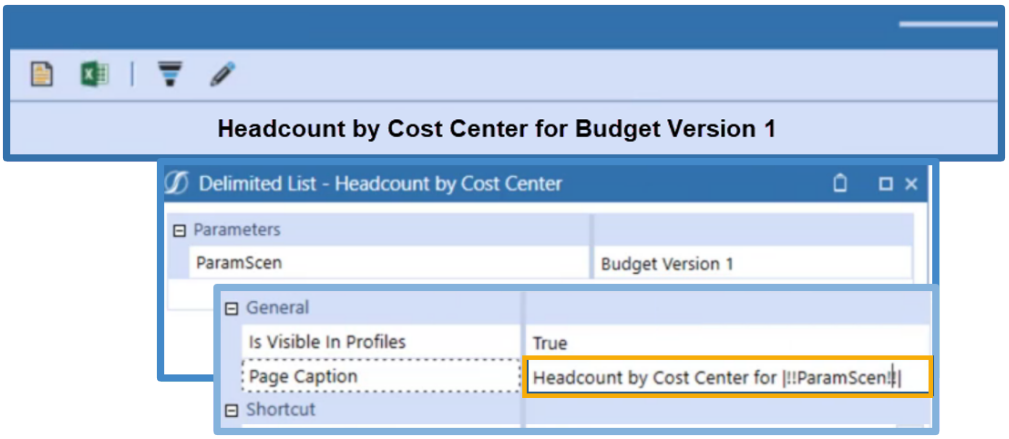
Enclose parameter names in pipes and exclamation points when you assign parameters to objects, regardless of parameter type. For example, |!ParamView!| prompts users to select year to date or periodic data when they run a cube view for reporting.

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

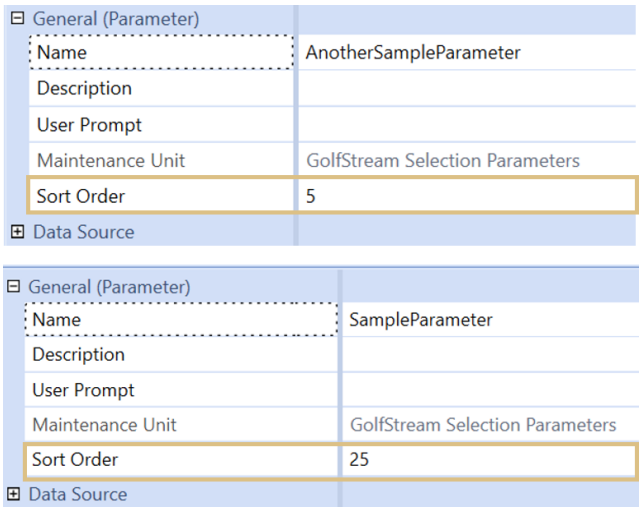
Design

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.

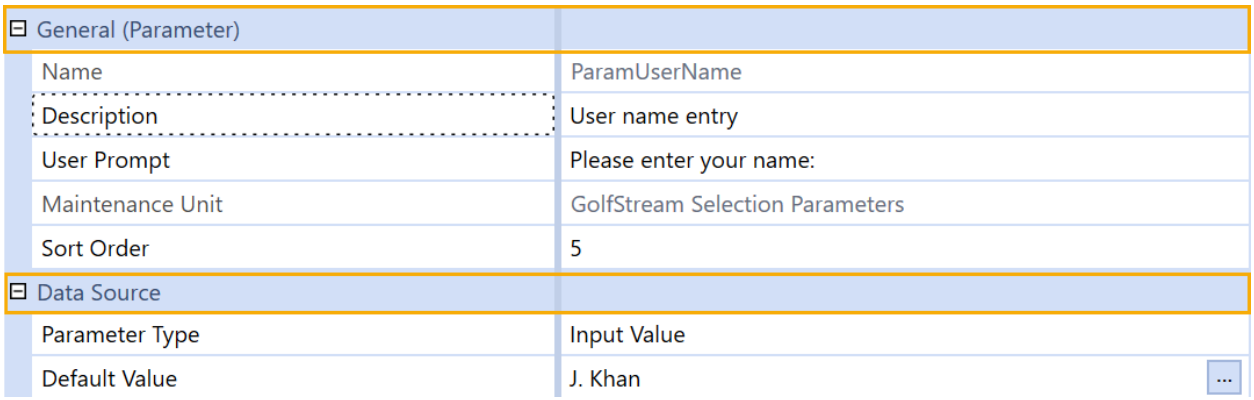
About Parameter Definitions

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 Nested Parameters.

1. Review the Requirements and Best Practices.
2. Click **Application** > **Presentation** > **Dashboards**

   > **TIP:** You can also define parameters by clicking **System** > **Administration** > **Dashboards**.

3. Click 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. Click **Create Parameter** in the dashboard toolbar.
5. In **Name** and **Description**, 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 *Parameter 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 list.
    - **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. Click **Save**.
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 click **Edit** to look it up.
2. Click **Save**.

> **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 click **Edit** to look it up.
2. Click **Save**.

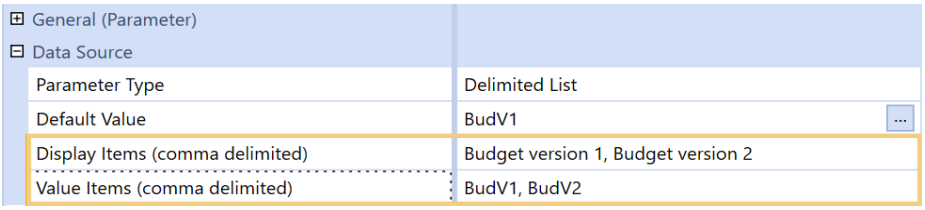
> **IMPORTANT:** When you assign the parameter, enclose the name in pipes and exclamation points. For example, |!DefaultHeader!|.

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 click **Edit** to look it up.
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 Syntax.

> **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. Click **Save** 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 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}
        ```

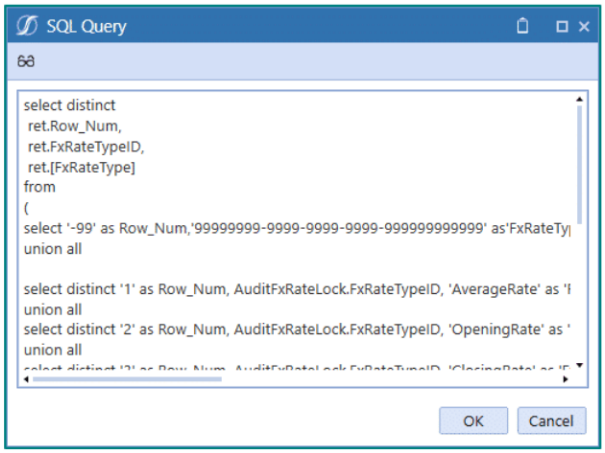
For a SQL Command Type: Use these properties to define the database location if using a SQL command type.

- **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 Servers* 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:

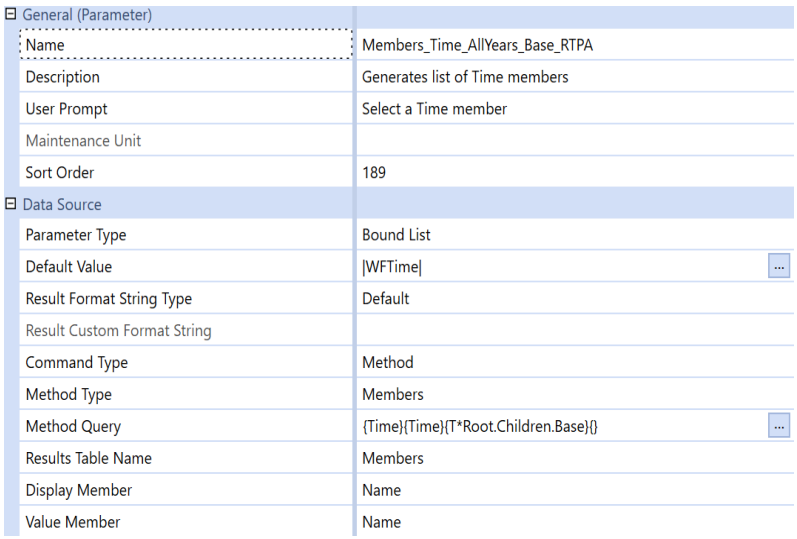


> **NOTE:** If you do not know the method query syntax, leave this field blank and click . This identifies the syntax location, runs the parameter, and provides a sample result set.

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. Click **Save** 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 click **Edit** to look it up.
2. Right-click in the **Default Value** dialog box, select **Paste** and then click **OK**.
3. Use **Display Member** to define how to display the returned members.
    - Type **Name** to display the member by name.
    - Type **Description** to display the member's description.
    - Type **Name and Description** to use both.
4. In **Cube**, either type the default value if known, or click **Edit** to use object lookup to select the cube containing the required dimensions.

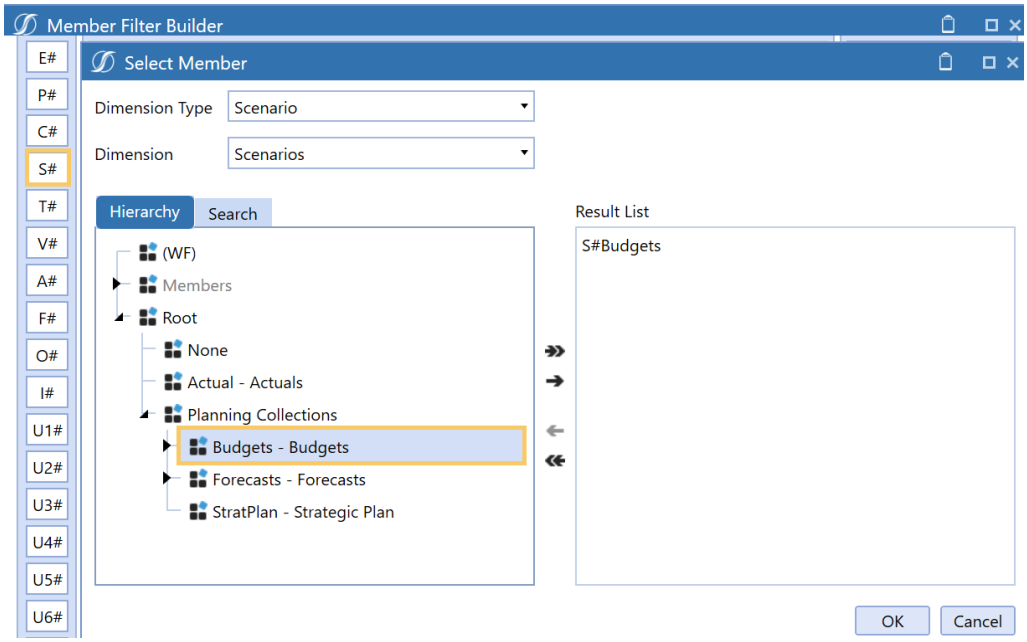
5. From **Dimension Type**, select the dimension category (such as Time or Account) that contains the appropriate dimension.
6. In **Dimension**, enter the name of the dimension to use or click **Edit** to select the dimension in the hierarchy.
7. In **Member Filter**, either type the member filter if known or click **Edit** to use the Member Filter Builder to specify the members to list.
8. Click **Save** to save your changes.

Specify Member Dialog Settings

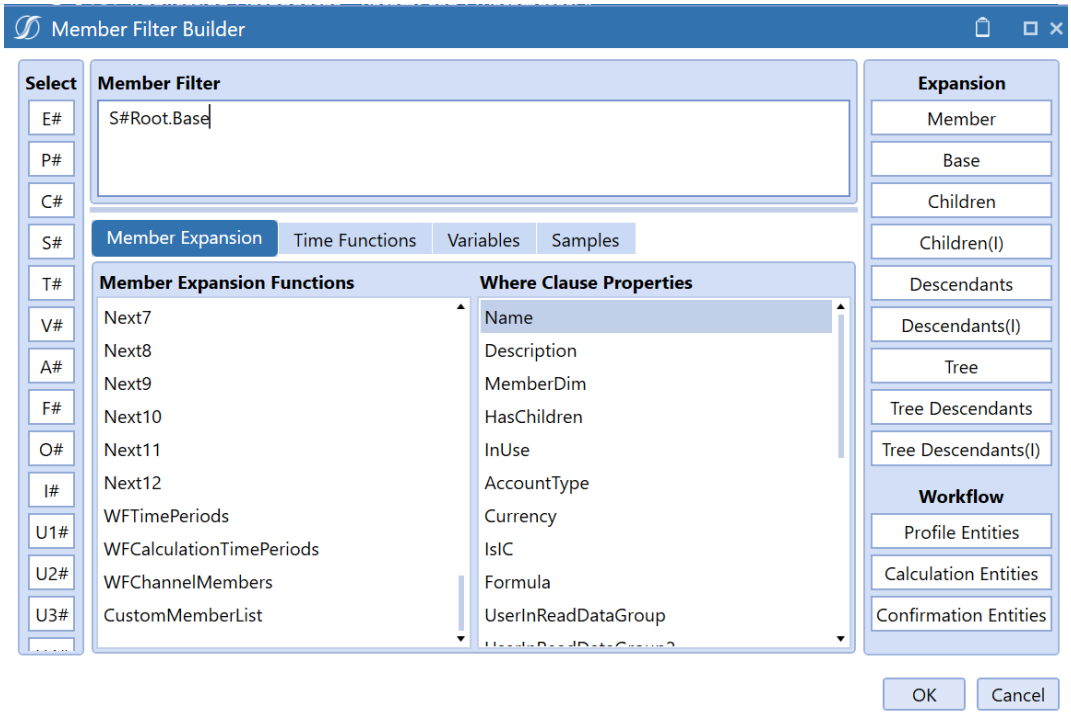
To finish defining a Member Dialog parameter:

1. (Optional) In **Default Value**, specify a dimension in the dimensional hierarchy to display when the parameter first runs. Enter the dimension name or click **Edit** to look up the dimension.
2. In **Cube**, click **Edit** to select the cube that uses the required dimension. To quickly find a cube, type 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 click **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 click **Edit** to use the Member Filter Builder to refine the members to display. For example:

- To display only Budget members in a Scenario dimension with multiple member groups such as Actual and Forecast, click **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. Click **OK** twice.
8. Click **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

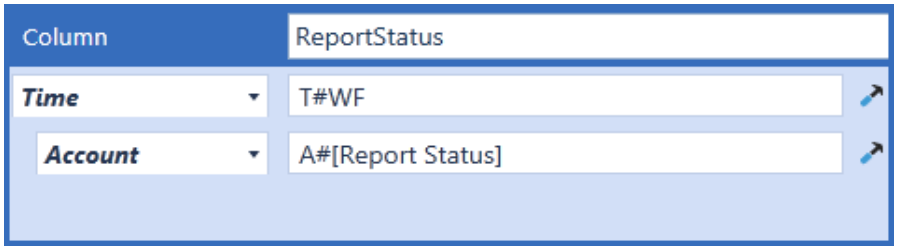
In this example, a Delimited List parameter is created and assigned to a sample GolfStream cube view so users can indicate if reports are not started, under review, or completed for each entity.

1. Click **Application** > **Presentation** > **Dashboard** and then select the dashboard maintenance unit where you want to store the parameter.
2. Click **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

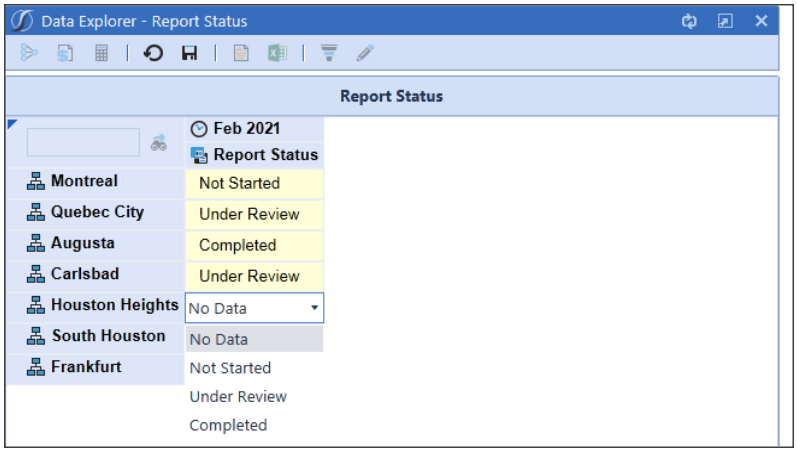
4. In **Display Items**, enter the status options that users can select when they run the cube view.
5. In **Value Items**, enter the corresponding status member names defined in the application. For example, *0NotStarted* is the Value Item for the Display Name *Not Started*.
6. Click **Save**.

7. Click **Application** > **Presentation** > **Cube Views** and select the cube view to use with the parameter.
8. Click **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. Click a 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 the cube 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 a parameter
- Rename a parameter
- Identify 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. Click **Application** and then the appropriate object.
2. Edit the object to assign the parameter. Click **Object Lookup** to find the parameter, and enclose the parameter name in pipes and exclamation points. For example, |!ParamDate!|. See Syntax.
3. Click **Save**.

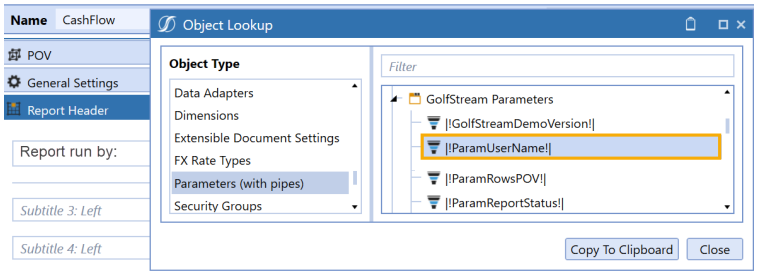
See these sample use cases:

- Assign to a Report Subheader
- Assign to a Cube View Column Header

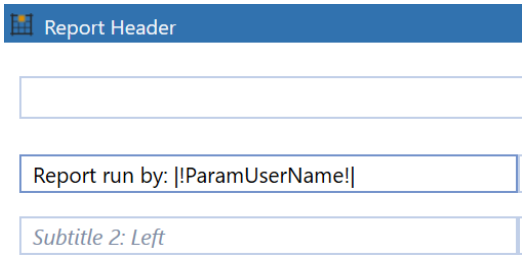
Assign to a Report Subheader

Perform these task to assign an Input Value parameter to prompt users to enter their name in a report header:

1. Click **Application** > **Presentation** > **Cube Views**.
2. Click **Cube View Group**, the appropriate cube view, **Designer**, and then **Report Header**.
3. In **Subtitle** enter **Report run by:**.
4. Click **Object Lookup**, and then **Parameters (with pipes)**.
5. Select the parameter to assign, and then click **Copy to Clipboard**.

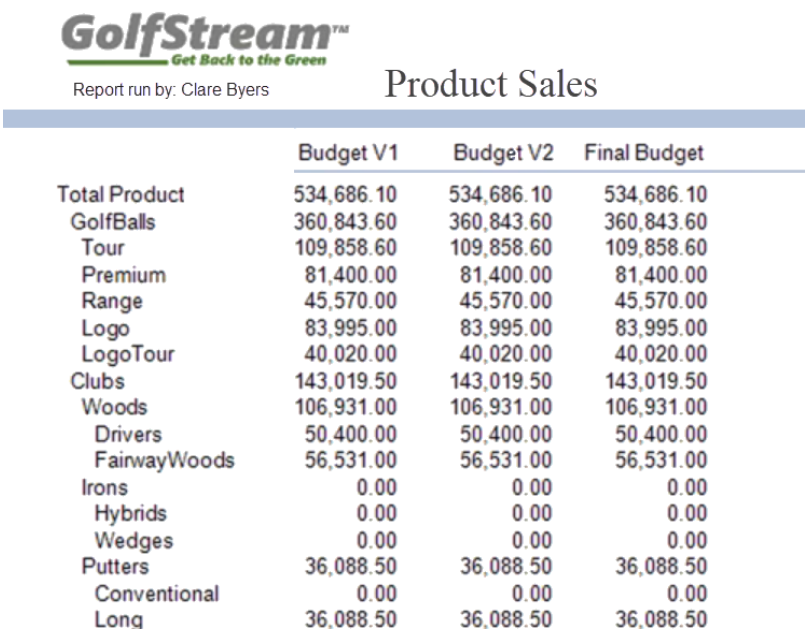


6. Paste the parameter after **Report run by:**, as shown:



7. Click **Save** and then **Open Data Explorer** to run the cube view.

8. Enter your name and click **Show Report**. The report generates, displaying your name as shown:



GolfStream™ Get Back to the Green

Report run by: Clare Byers

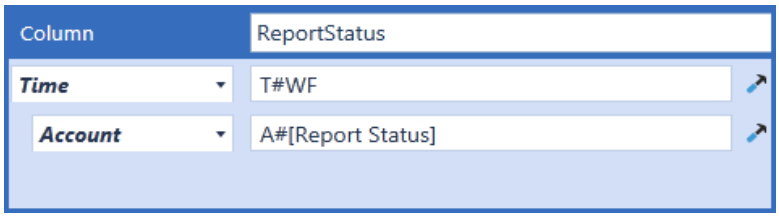
Product Sales

	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

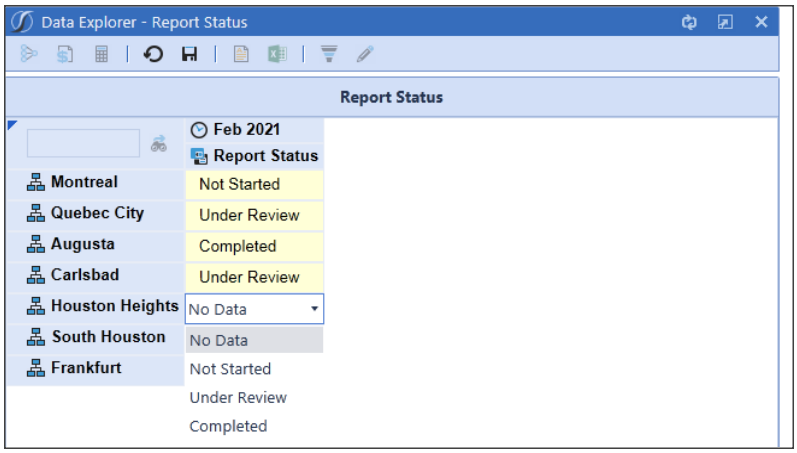
In this example, a Delimited List parameter is assigned to a sample GolfStream cube view so users can indicate if reports are not started, under review, or completed for each entity.

1. Click **Application** > **Presentation** > **Cube Views**.
2. Click the appropriate cube view.
3. Click **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. Click 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 affects of modifying it. Select the parameter and click **Show Objects That Reference The Selected Item**.

1. Click **Application** > **Presentation** > **Dashboards**.
2. Click the dashboard maintenance unit that uses the parameter, and then **Parameters**.
3. Click the parameter to modify settings described in Create a Parameter.
4. Click **Save**.

Copy a Parameter

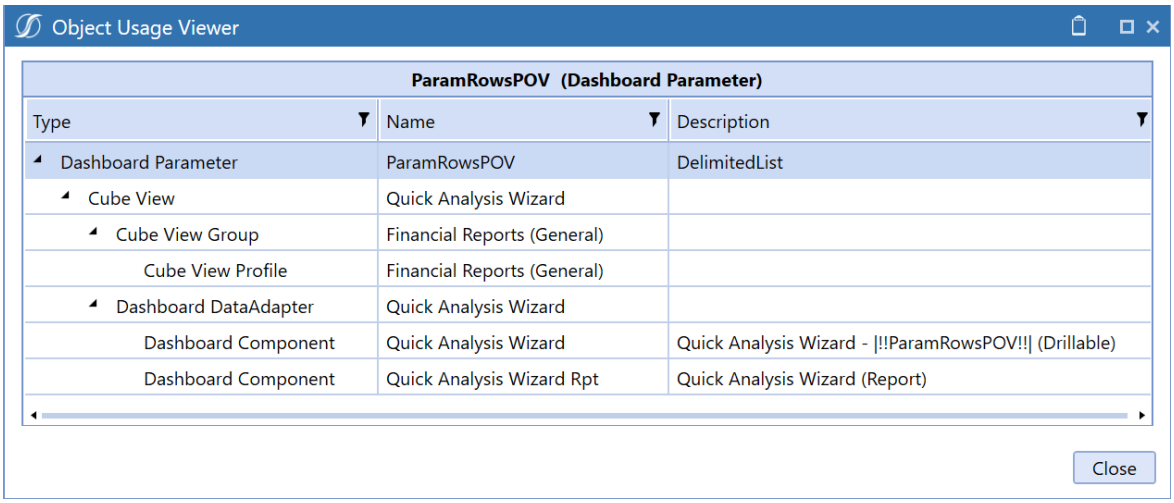
Copy a parameter to quickly create a new parameter that must function similarly.

1. Click **Application** > **Presentation** > **Dashboards**.
2. Click **Dashboard Maintenance Unit**, the dashboard maintenance unit in which the parameter is used, and then **Parameters**.
3. Click the parameter upon which to base a new parameter, then click **Copy**.
4. Click **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. Click **Application** > **Presentation** > **Dashboards**.
2. Click **Dashboard Maintenance Unit**, then the dashboard maintenance unit in which the parameter is used.
3. Click the parameter, then **Show Objects That Reference The Selected Item**. This identifies the objects from which you must remove the parameter. For example:



4. Click **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. Click **Cube View Groups**, the cube view, and then **Designer**.
    b. Click **Report Header**, delete the parameter from the subtitle, and then click **Save**.
5. To delete the parameter:

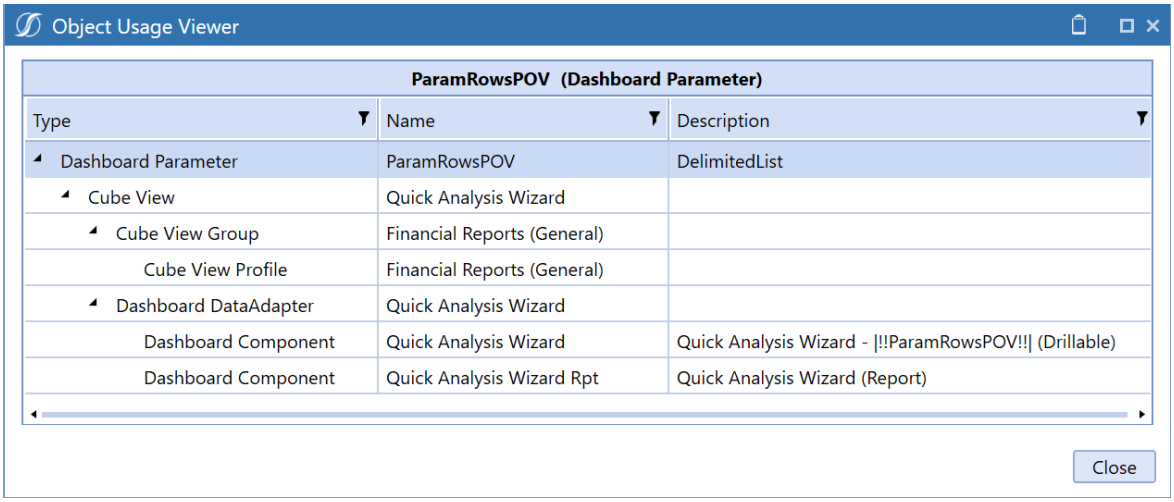
a. Repeat steps 2 - 4 to ensure you remove the parameter from all objects.

b. Click the parameter and then **Delete**.

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. Click **Application** > **Presentation** > **Dashboards**.
2. Click the dashboard maintenance unit that uses the parameter, and then **Parameters**.
3. Click 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:



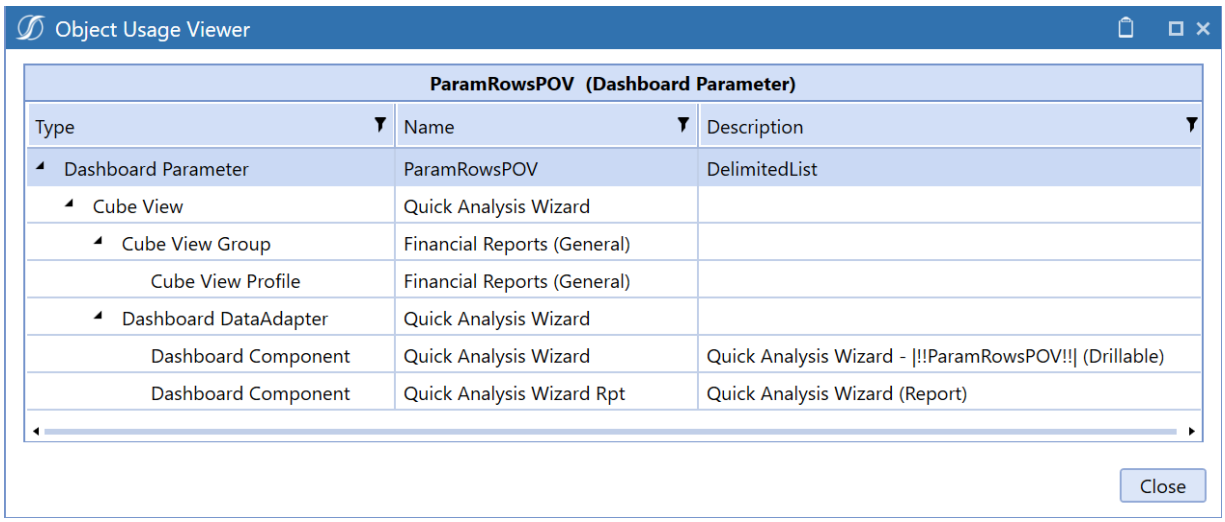
4. Click **Close**.

5. Click the parameter, then **Rename** and enter a different name.
6. Click **Save**.

Identify Parameter Assignments

Perform these steps to access a list of objects to which a parameter is assigned:

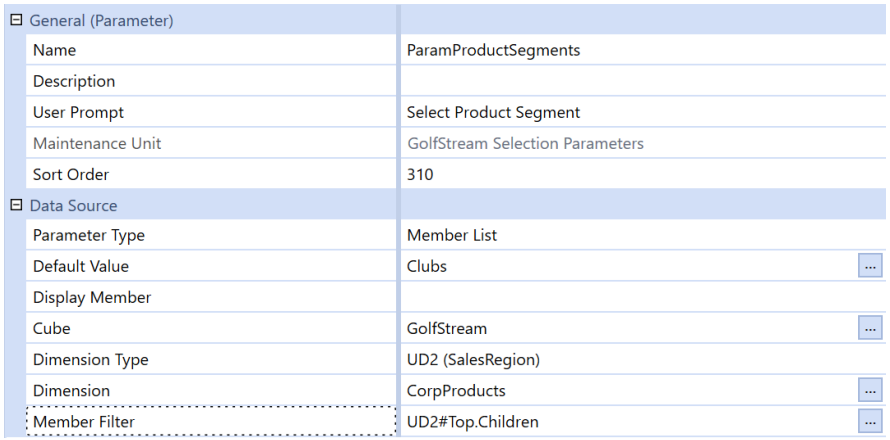
1. Click **Application** > **Presentation** > **Dashboards**.
2. Click the dashboard maintenance unit associated with the parameter, and then **Parameters**.
3. Click the parameter and then **Show Objects That Reference The Selected Item**. This lists all the objects that use the parameter.



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.

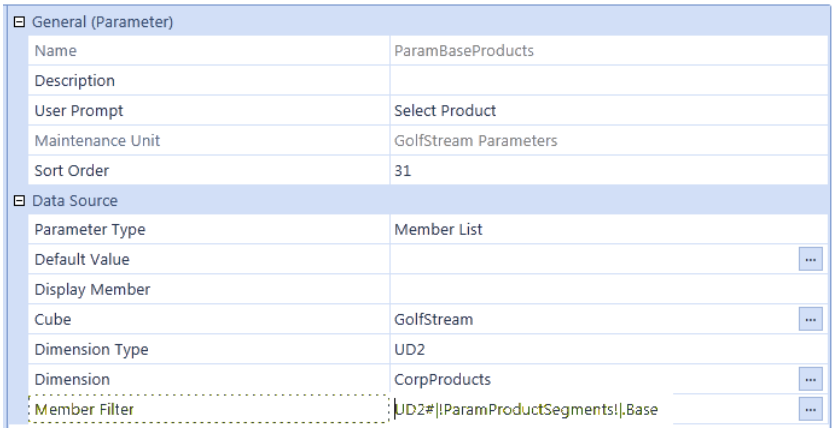
1. Click **Application** > **Presentation** > **Dashboards**.
2. Click the dashboard maintenance unit in which to store the parameters.
3. Click **Parameters**, and then **Create Parameter**.
4. Define the first parameter as shown below to prompt users to select a product segment.



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 ...

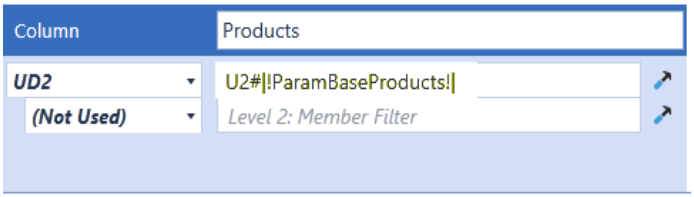
The Member Filter returns the child members of Top, in the UD2 dimension, in a drop-down list so users can select a product segment. For example, if Top has 2 children, TotalProducts and TotalServices, users can select only those members.

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.

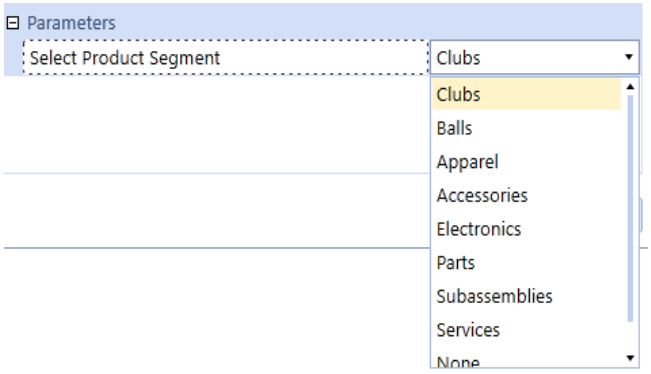
6. Save both parameters, and then click **Application** > **Presentation** > **Cube Views**.
7. Select the cube view where you want to use the parameters.
8. Select the column or row that requires a parameter and enter the parameter name using pipes and exclamation points. See Syntax. The sample cube view below displays products in columns, so a column is selected.



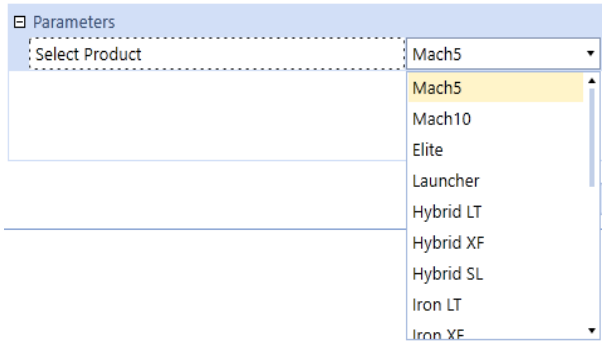
9. Select a dimension and click **Member Filter Builder** to ensure the ParamBaseProducts parameter references the ParamProductSegments parameter first, then returns a product list based on a selected segment.

10. Click **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.



Presenting Data With Extensible Documents

The Extensible Document Framework integrates OneStream content with Microsoft Office and Text Files. In turn, these documents become known as Extensible Documents. You can create Extensible Documents by selecting the type of document desired, which can be a Microsoft Word Document, Text Editor Document, Excel spreadsheet, OneStream Spreadsheet, PowerPoint presentation, or a text File. Extensible Documents allow you to display any information you want from OneStream, and because it is integrated with these different products, the data stays current and dynamic. For the purpose of this guide, XF Docs can also be referenced as Extensible Documents.

This section describes how to work with Extensible Documents.

Extensible Document Framework

The Extensible Document Framework integrates OneStream content with Microsoft Office and Text Files. In turn, these documents become known as Extensible Documents.

You can create extensible documents by first selecting the type of document desired, which can be a Microsoft Word document in Microsoft Word or Text Editor, Excel spreadsheet, OneStream Spreadsheet, PowerPoint presentation, or text file. After selecting the document, use OneStream's custom parameters, substitution variables, document variables, and retrieve functions to get specific information such as time periods, entity names, data cell values, text comments, and other information. You can also insert images into a Microsoft Word document, Text Editor document, PowerPoint presentation, or Excel spreadsheet for reports, charts, Cube Views, Excel spreadsheets, or PDF files.

After you configure an extensible document, save it as `[NameofDocument].xfdoc.ext`, where the `xfdoc` portion of the file name tells OneStream that this is an extensible document (for example, `CostSpreadsheet.xfdoc.xlsx`) and the .ext portion of the file name tells the type of file it is. Load the document into OneStream's file share to launch it from OneStream or include it in dashboards or report books.

The value of Extensible Documents are inherently linked to OneStream and eliminate the need for third party integration tools. Upon launching a document, the content is displayed based on the correct parameter values, images, or retrieve function values. Extensible documents allow you to display live data and objects from OneStream, which enables dynamic use and confidence in the information displayed.

About Creating Extensible Documents

Extensible Document Creation Process

To create an extensible document, begin with the Microsoft document or Text File needed to create the framework. Next, decide the type of data required and insert parameters, images, or retrieve functions needed to pull data from OneStream and dynamically update the extensible document. Once the document is complete, save it as follows (all examples below are using the file name `GolfStream Report`):

Saving a Word Document

`GolfStream Report.xfdoc.docx`

Saving an Excel Spreadsheet

`GolfStream Report.xfdoc.xlsx`

Saving a PowerPoint Presentation

`GolfStream Report.xfdoc.pptx`

Saving a Text File

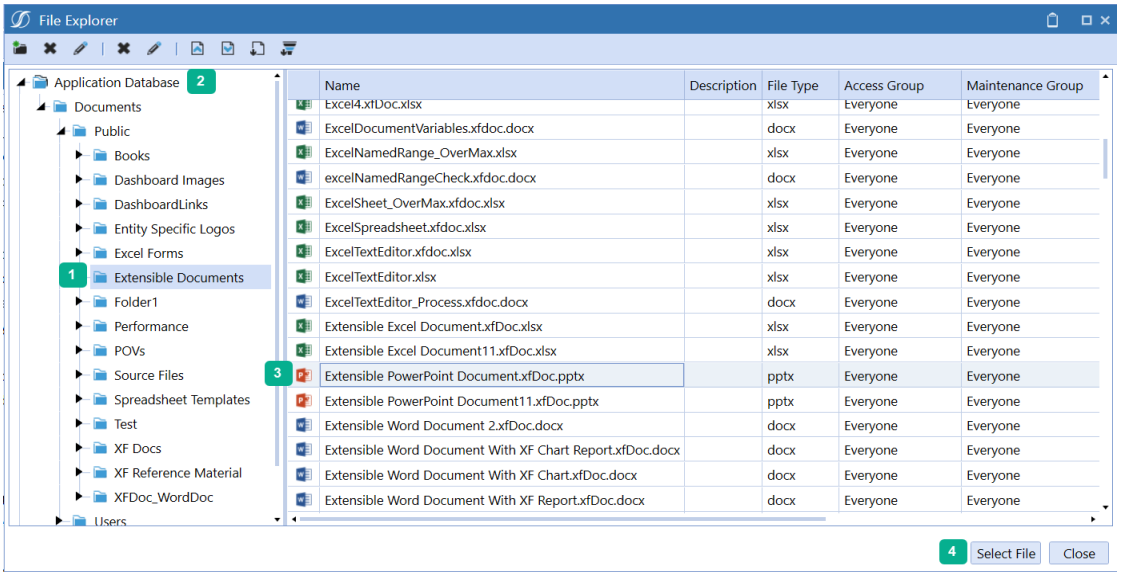
`GolfStream Report.xfdoc.txt`

> **NOTE:** Extensible documents are compatible with Microsoft Office version 2007 and later.

> **IMPORTANT:** When processing Document Variables within the .xfdoc file extension, OneStream will allow you to reload the file and not strip the ".xfdoc" out of the filename.

Once the document is saved in the format mentioned above, upload the document into OneStream's File Explorer.

Click **File Explorer** in OneStream to launch the **File Explorer** dialog box.



Select the folder in which to save the extensible document.

1. Select the folder in which to save the XF Docs.
2. Upload the document into OneStream.
3. Click on the desired file name.
4. Click **Select File** to launch the document and see the updated values.

Using Document Variables in Extensible Documents

Document Variables allow you to insert content into a Microsoft Word document or Text Editor as XF Docs. When using this method, you can view the processed OneStream content and edit the document simultaneously for Cube Views, Excel Sheets, Excel Named Ranges, Dashboard Reports, and more.

See Document Variable Setup and Creating an Extensible Document in Text Editor for more information.

Using Parameters and Substitution Variables in Extensible Documents

When creating extensible documents, use parameters as placeholders for OneStream member names, data, or user comments. Place a parameter in different areas of a document where specific information is needed. For example, if the |!MyEntity!| parameter was used in the document's framework, once it was launched from OneStream, the parameter would be recognized and replaced with the desired entity name.

Some parameters may prompt the user to select a specific entity, account, time period, or other data to view the document correctly. Parameters can be used in any Word document, PowerPoint presentation, Excel spreadsheet, or text file. Any parameter that exists in OneStream can be used in an extensible document. See Parameters for more details.

See the Creating an Extensible Document sections for examples on how to incorporate parameters into an extensible document.

Substitution variables can also be used in Word, Excel, PowerPoint, or a text file. These variables call out details such as an Application Name |AppName|, User Name |UserName|, or refer to a specific POV which creates versatility when reusing the same document. See Substitution Variables for details.

See Creating a Document in Microsoft PowerPoint for an example on how to incorporate substitution variables into an extensible document.

You can preserve formatting if you are inserting a parameter into an Extensible document. This is optional and allows you to include formatting and ensure formatting is preserved during updates. See Parameters in Extensible Documents for more information.

Using Images in Extensible Documents

You can display a variety of reports, charts, Cube Views, Excel spreadsheets, PDFs and rich text in an extensible document. Any one of these items can be used in a Word document, PowerPoint presentation, or Excel spreadsheet.

You can insert any image, right-click on it, and choose the View Alt Text right-click option. In Microsoft Word, this will open the View Alt Text box. In the Alt Text box, you can enter the required Document Variable arguments, referencing the path to the content to display in place of the current image.

After saving the Document Variable argument Alt Text in the image, you must Process and Open the document through the OneStream File Explorer or refresh the OneStream Text Editor or Spreadsheet page to process it. After processing the extensible document, the Document Variable content will replace the image.

For more details on an image's configuration, refer to the extensible document syntax in the **Object Lookup** dialog box in OneStream, or see Extensible Document Settings.

See Creating an Extensible Document in Microsoft Word and Extensible Documents with Rich Text Overview for more information.

Using Retrieve Functions in Extensible Documents

You can use retrieve functions such as `XFGetCell` in Excel. These functions run on the server when launched from OneStream. Log into the Excel Add-In, load a cube view, convert to XFGetCell, then click **Refresh Data** to display the updated Excel spreadsheet.

You can also create Excel charts based on the retrieve values. These charts display the correct data when you launch the spreadsheet from OneStream. Use the `XFGetCellVolatile` retrieve function to refresh the Excel chart and display updated data. Excel requires a volatile function for proper refreshing when using charts that reference calculated cells.

See Creating an Extensible Document in Microsoft Excel for an example on how to use retrieve functions with extensible documents.

There is also another type of retrieve function called `XFCell` which retrieves data from a single cell in OneStream. This is intended for text documents such as Word or PowerPoint. For example, `XFCell(A#20500:E#Clubs)` returns a value for the account and entity intersection.

Parameters and substitution variables may also be used in an `XFCell` formula. For example, `XFCell(A#20500:E#|!MyEntityParameter!|:T#|Global|)` returns a value for the specified account, the entity selected at run-time, and your application's global time period.

You can include additional settings in an `XFCell` function used within an extensible document to format resulting data (for example, `XFCell(A#20500:E#Clubs, Culture=User, NumberFormat=N3, DisplayNoDataAsZero=True, Scale=3, FlipSign=True, ShowPercentSign=False`).

> **NOTE:** Any dimensions not specified in the formula come from the current POV.

For more examples and details on XFCell's syntax, refer to the extensible document Settings in the **Object Lookup** dialog box in OneStream, or see Extensible Document Settings under Object Lookup in Presenting Data With Books, Cube Views and Dashboards.

See Creating an Extensible Document in Microsoft PowerPoint for an example on how to use XFCell with extensible documents.

Using Rich Text Content Controls

Rich text content controls are used in Microsoft Word only and allow users to add a Cube View, a dashboard report, a Word document (`.docx` or `.xfDoc.docx`), a rich text file (`.rtf`), or a text file (`.txt`) to any extensible Word document. When the extensible document is launched, the content control is replaced with formatted text that can be edited and reformatted as desired. See Creating an Extensible Document in Microsoft Word for more information.

Document Variable Setup

You can start in Microsoft Word or OneStream Text Editor to embed a document variable with narrative. When processing your document in OneStream File Explorer, select “**Process and Open in Text Editor Page**” or "**Process and Open**" to see the output in Text Editor or Microsoft Word. You can also make any edits or changes to the narrative (non-document variable content) in Text Editor including inserting new text or images.

There are various functions you can use to easily embed refreshable content such as:

Document Variables: Allows you to insert content in a Microsoft Word document or OneStream Text Editor for XF Docs.

Rich Text Functionality: Allows document variables to support rich text functionality and view rich text content in Microsoft Word or OneStream Text Editor.

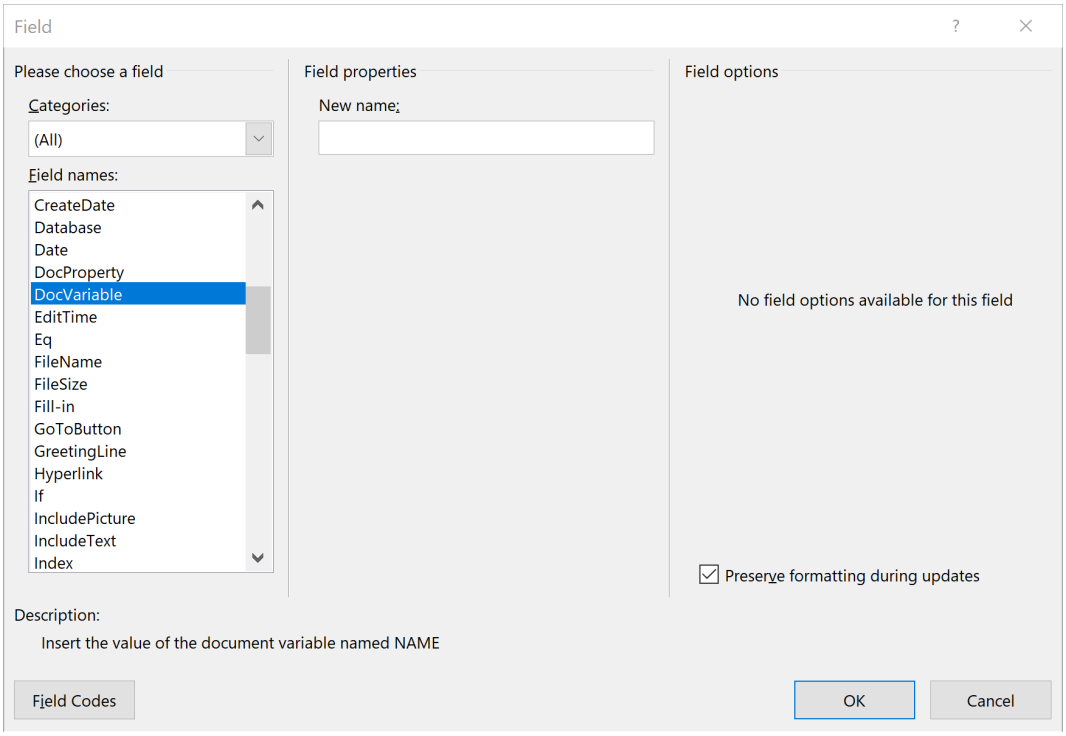
Show/Hide Field Codes: Allows you to view embedded content while in edit mode by toggling the content to see it processed or unprocessed. This allows you to update Document Variables quickly then refresh the Text Editor page to view updated content instead of navigating to OneStream File Explorer to Process and Open again.

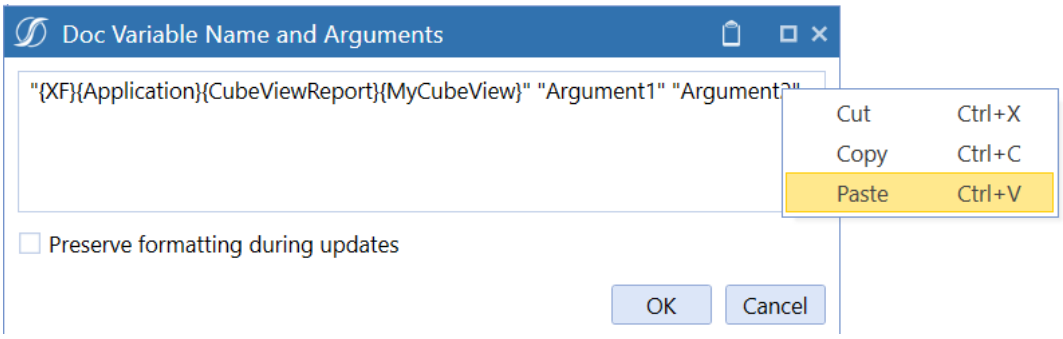
Refresh Content: Allows you to save and refresh your edits directly from the document to see your updates.

MergeFormat: Allows you to preserve document formatting when using Parameters or Substitution Variables.

Document Variable Field

This is the field that allows you to insert extensible content from OneStream objects in Microsoft Word or Text Editor for XF Docs.

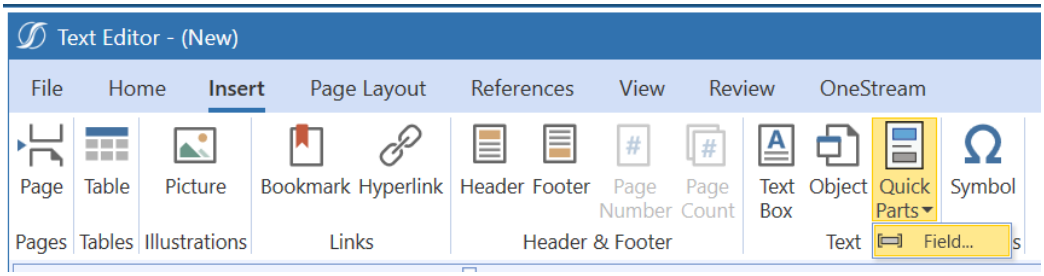


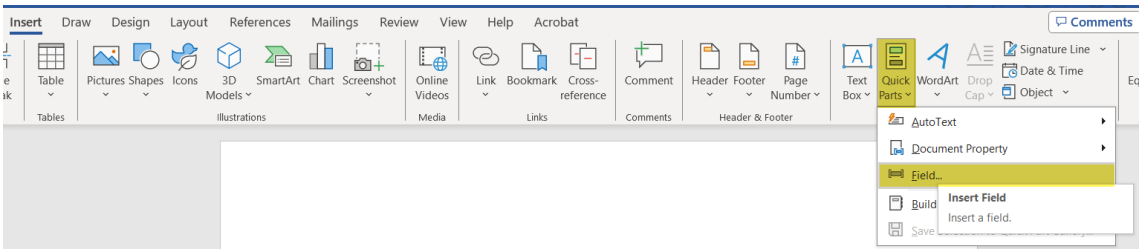


Quick Parts

Quick Parts is a Microsoft Word document feature and is available in Text Editor. The concept is similar to how we use Substitution Variables in OneStream.

1. You insert fields into a document that serve as a placeholder for items such as Cube Views Reports, Excel Sheets, Excel Named Ranges, and more.

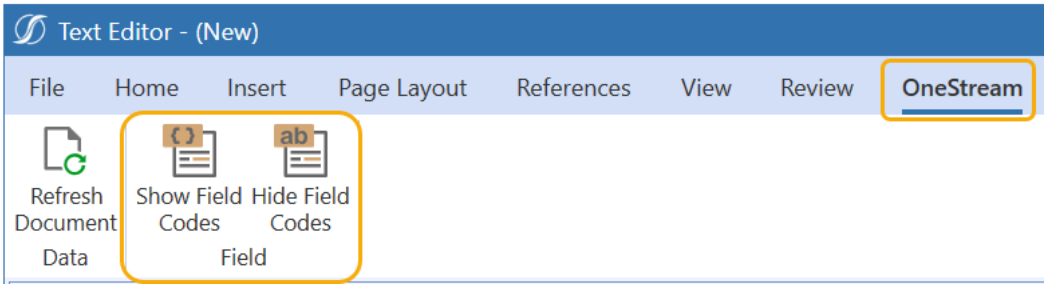




2. Upon saving, refreshing, and opening the doc, the fields dynamically update and display the desired content.
3. The field attributes and settings determine what, how, and the location of the field controls of where it displays.

Field Code

This function allows you to define the content item and formatting. Within Text Editor, you can use the **Show Field Codes** and **Hide Field Codes** buttons under the OneStream Ribbon to show or hide field codes.

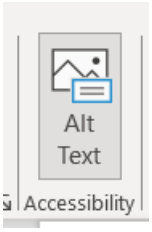


Within a Microsoft Word document, use Toggle (right-click option) in Word for the same result. You can also select the **Alt+F9** keys.

> **TIP:** If pressing the **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

Alt Text

You can edit using the Alt Text functionality in OneStream within Text Editor. This is located under **Application**> **Tools**>**Text Editor**>**Format**>**Alt Text**. In Microsoft Word, this is located under **Picture Format**>**Alt Text**. Alt Text is disabled for Text Boxes and Objects within Text Editor.



> **NOTE:** In OneStream Text Editor, the Format ribbon will not appear unless an image is selected.

Rich Text Functionality

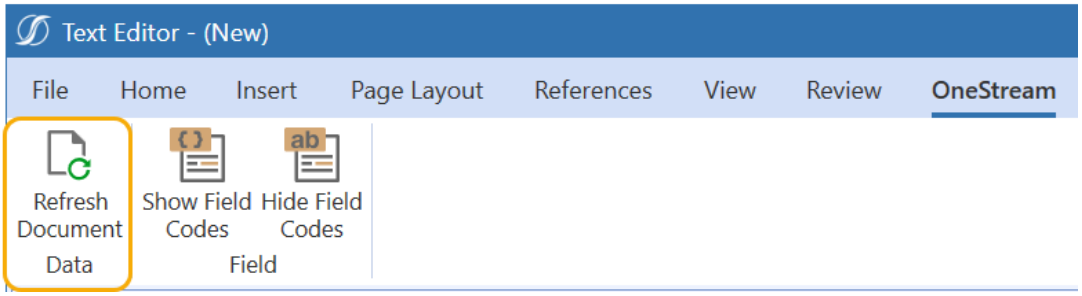
Document variables support the use of parameters in Microsoft Word or OneStream Text Editor. You can view parameters containing Literal Values equal to rich text format, and which can be referenced in the Document Variable. You can also paste rich text format or plain text to display in the specific Document Variable. Then, you can process and open the XF Docs in Microsoft Word and OneStream Text Editor.

> **TIP:** If the rich text parameter content does not immediately display in Microsoft Word, press the **Alt** + **F9** or **Alt** + **Fn** + **F9** keys simultaneously to display the processed content.

Refresh Document

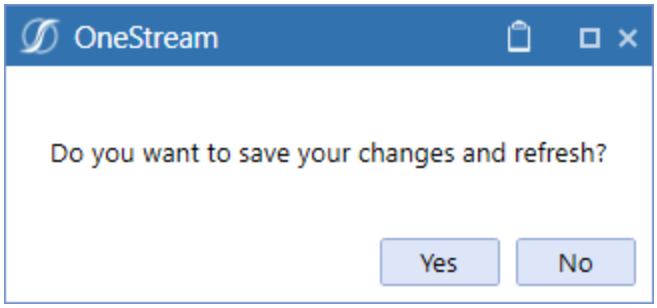
You can automatically refresh a document and OneStream will update all the new content and narrative immediately within Text Editor. This allows you to edit, save, and re-process all content quicker in Text Editor.

1. Under the OneStream ribbon within Text Editor, a **Refresh Document** icon displays.



2. When you click on the Refresh Document icon, the embedded content will be refreshed.

3. When you click on the Refresh Document icon, in a processed XF Doc, you will be prompted with a dialog. The dialog will display "Do you want to save your changes and refresh?"



> **NOTE:** If you select Yes, the document will save and re-process. If you select No, it will not save the document or re-process.

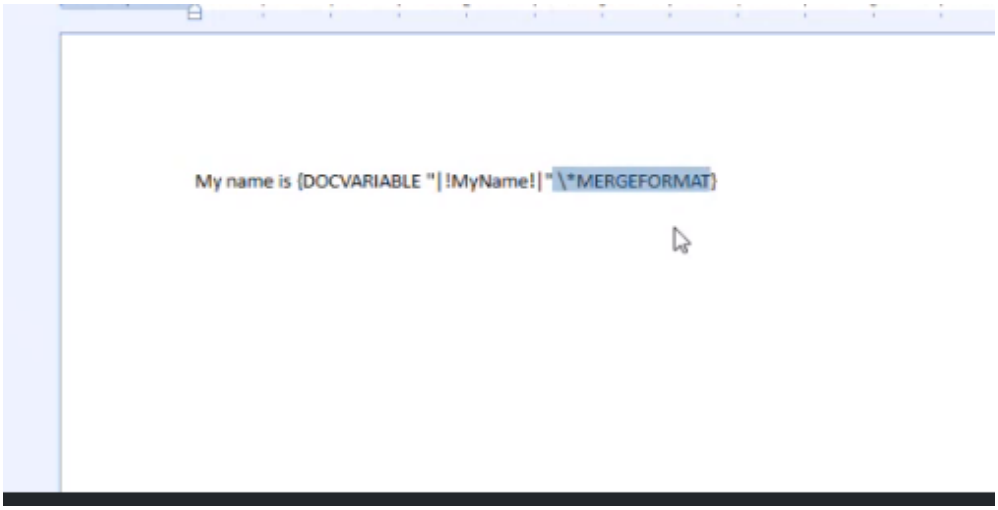
Preserve Formatting During Updates

You can preserve formatting if you are inserting a parameter in an XF Doc. This is optional and allows you to include and ensure that formatting will be preserved during updates through MergeFormat.

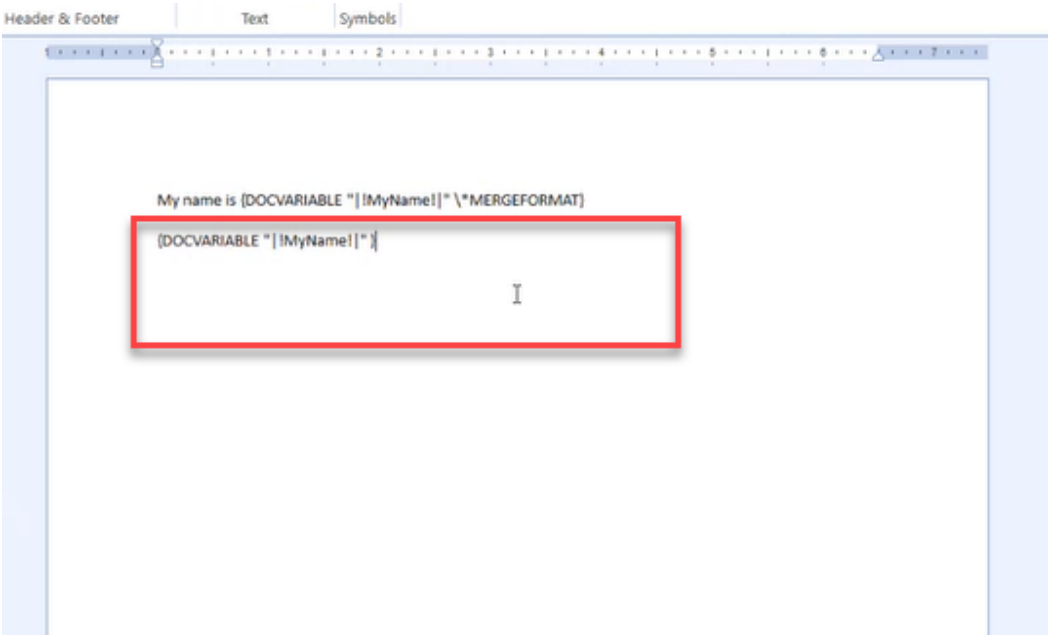
1. When inserting a document variable within a Microsoft Word or Text Editor, the document variable field dialog appears. There is an option to select **Preserve formatting during updates**.

> **NOTE:** Within Microsoft Word, it is checked by default.

2. When the **Preserve formatting during updates** checkbox is selected, it will add the "*MERGEFORMAT" syntax to the end of the variable.



When the **Preserve formatting during updates** checkbox is not selected, the "*MERGEFORMAT" syntax will not be added to the end of the variable.

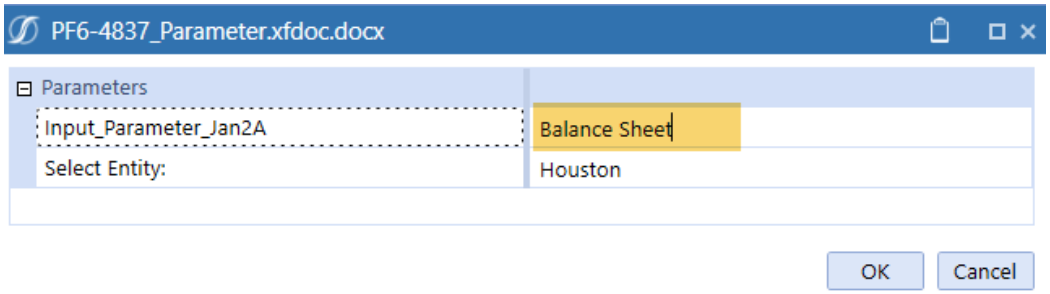


Parameters in Extensible Documents

Using Parameters

You can use parameters and substitution variables as placeholders to enhance extensible documents with selections for time period, entity, data cell values, and more.

In this example, parameters were inserted in places to obtain information from OneStream. The parameters named were created to display the input parameter for the cube view name and entity for this document. When the XF Doc is processed at run-time, it prompts you to select both values.



After selecting parameters, the document displays the desired data. Additionally, you can update content on the parameter and add more narrative to an XF Doc. You can inherit the format of a parameter from where it's being inserted within a paragraph.

For example, you make edits to your XF Docs by making the text red, bold, and font size 18. Within OneStream Text Editor or Microsoft Word, the values display with the associated "Cube View" requested in the Document Variable field and the text entered appears with the correct font, color and boldness. The newly added Income Statement cube view appears in the report and the second value entered into the XF Doc.

"MERGEFORMAT" text can be removed. This simply means, if you insert this in Microsoft Word, it will take the formatting of the previous paragraph or previous range in the paragraph.

Parameter and Substitution Variable Setup in Text Editor

In a new or existing XF Doc:

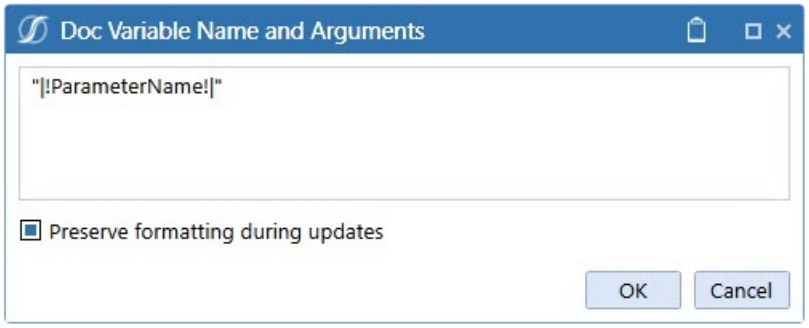
1. Within Text Editor, navigate to **Insert**> **Quick Parts**> **Field**.
2. Remove all existing syntax between the first set of quotes and everything after.
3. Copy and paste the Parameter or Substitution Variable name with surrounded by quotes; make sure to include standard object syntax.

   > **NOTE:** Ensure to include standard object syntax covers the exclamation points and the pipes around the parameter.

4. Enable **Preserve formatting during updates**.

   > **NOTE:** This is an optional setting that will preserve formatting.

5. Click **OK.**



Result:

Create an Extensible Document in Microsoft Word

You can easily embed refreshable content to allow users to edit and narrate a document for the purposes of publishing it into a report. This allows you to embed and refresh content using Document Variables in a Microsoft Word document through the following actions:

- Insert a Cube View in Microsoft Word
- Insert an Excel Sheet/Excel Named Range in Microsoft Word
- Insert a Microsoft Word Document in Microsoft Word
- Insert a Rich Text Document in Microsoft Word
- Insert a Text Document in Microsoft Word
- Insert a Report in Microsoft Word

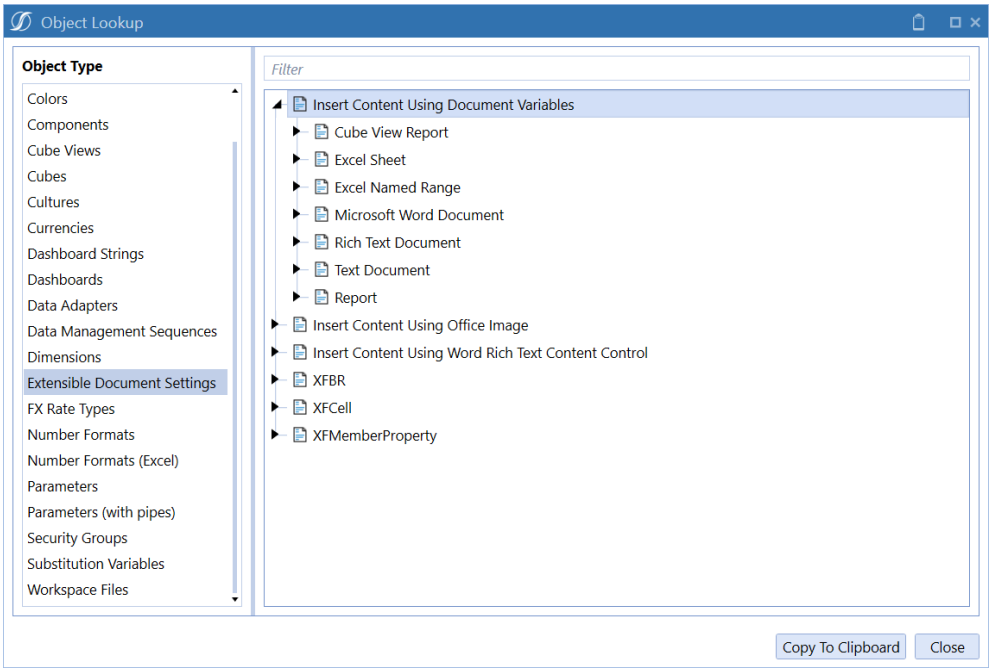
NOTE: When inserting content into a Word document, the embedded content takes on the page settings of the main document. To change the page settings for the inserted content, add a section break before the inserted content. After the section break, specify the desired page settings for the embedded content.

IMPORTANT: Copy all syntax needed to insert content using document variables directly from Object Lookup. See Extensible Document Settings.

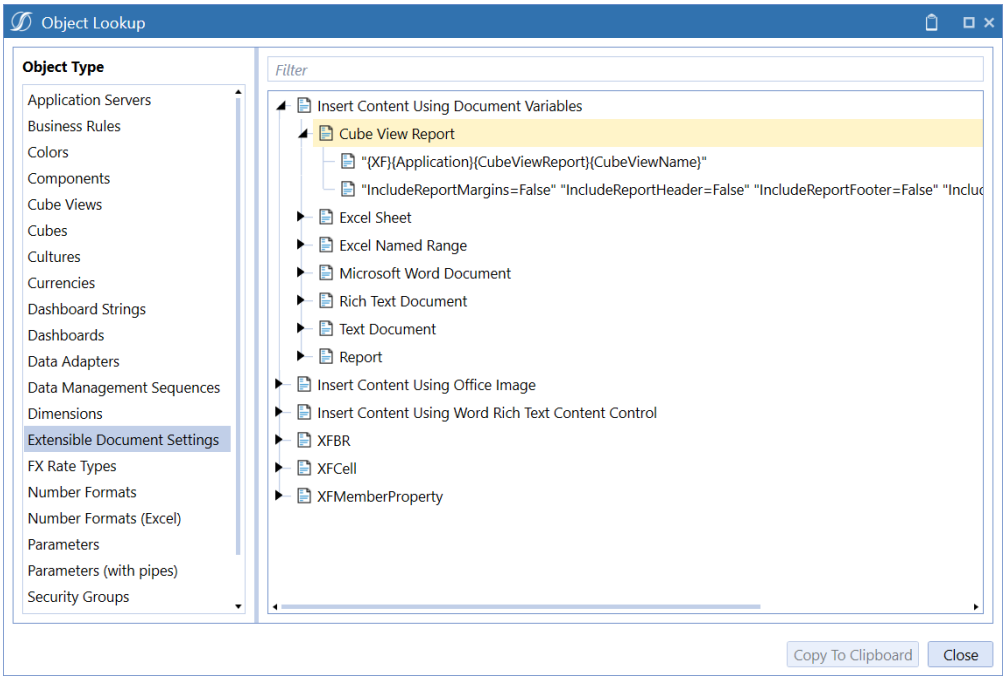
Insert a Cube View in Microsoft Word

> **NOTE:** The same format for inserting an image is used in Microsoft PowerPoint and Excel. Also, for Microsoft upgrade Office version 1804 or higher, Microsoft changed how the **Alt Text Title** property is stored for embedded images. This impacts the creation of extensible documents in OneStream. Existing extensible documents continue to run as expected. It only impacts newly created extensible documents for version 1804 or higher of Microsoft Office. See the additional following section on this topic.

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views
    - Workspaces
    - Data Management

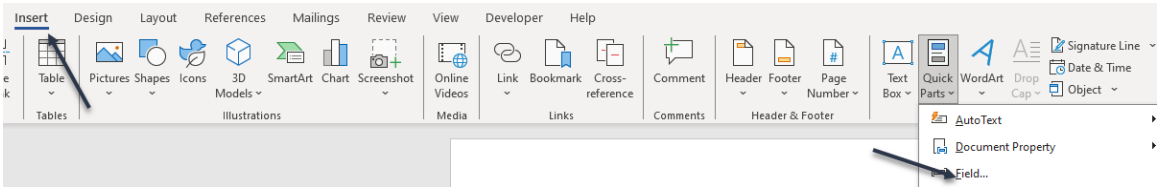


2. In the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Cube View Report**. Select the line of syntax, and click the **Copy to Clipboard** button to copy the first string.

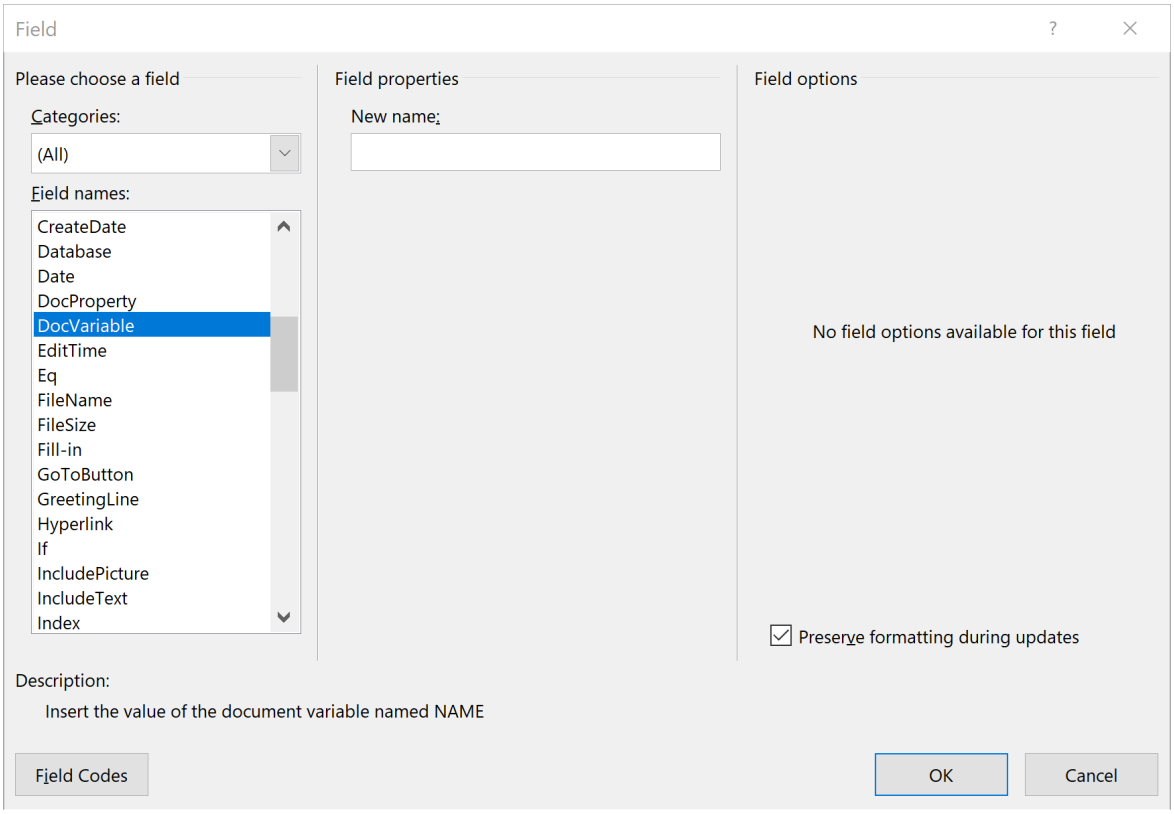


3. In Microsoft Word, begin with a new or existing document.

4. To insert your string, go to **Insert** > **Quick Parts** > **Field**.



5. A Field dialog box displays. Under **Field Names**, select **DocVariable**. In the **New name** field, paste the string. The string "`{XF}{Application}{CubeViewReport}{CubeViewName}`" displays in the field.



Optional: After the first string in the DocVariable field, you can paste the "IncludeReportMargins=False" arguments. You must separate the arguments by a space. See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under **Field Options**, ensure **Preserve formatting during updates** is selected. See Preserve Formatting during Updates for more information.

6. Update the values to reflect the folder structure and the cube view you will add to the report. Then in the Field dialog, click the **OK** button. The final string will display in the Microsoft Word document: "`{DOCVARIABLE}{XF}{Application}{CubeViewReport}{IncomeStatement}`".

> **IMPORTANT:** You must manually update and remove extra " (quotation mark) and \ (backslash) symbols, or the Document Variable will not process correctly.

7. Place your cursor on the page, and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page.

   > **IMPORTANT:** If expected content does not display after clicking the **OK** button in the Field dialog box, press the **Alt+F9** or **Alt+Fn+F9** keys.

   > **TIP:** If pressing the **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As**. Select a file name and add **.xfdoc** extension to the name. The file name is appended with ".xfdoc.docx".
9. Click **Save** to save the file and ensure it is saved in a familiar folder. For example, Documents.
10. Navigate back to OneStream, and open **File Explorer**. Under the folder of choice, select and upload the file in File Explorer.
11. Right-click on the file and select **"Process and Open"**. You may need to enter or select a parameter and then click **OK**.

    The Cube View Report is now embedded in the extensible Word document.

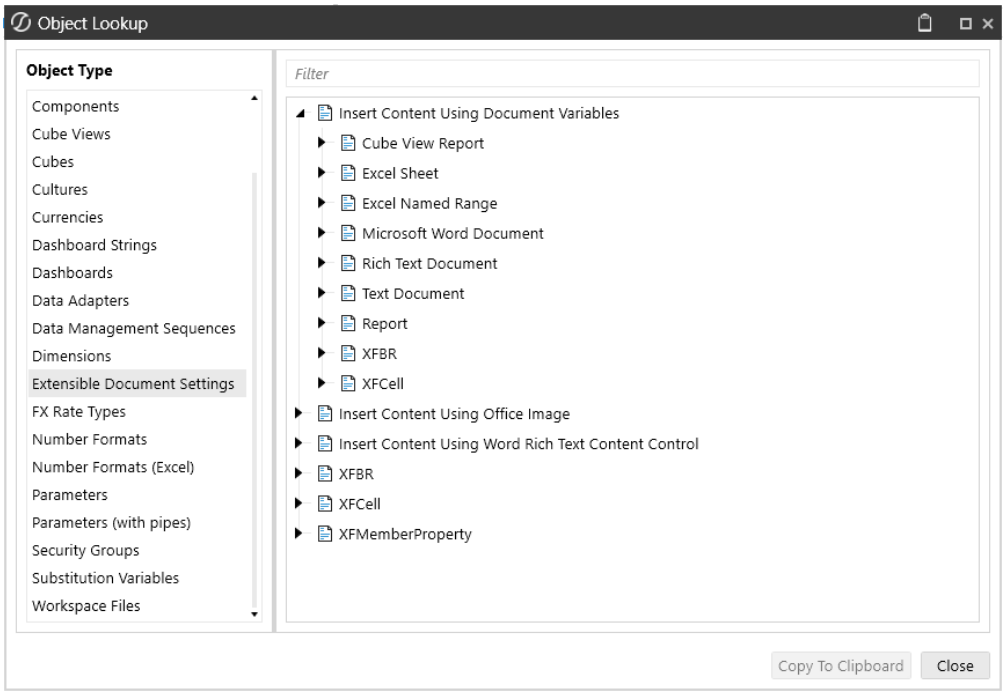
> **NOTE:** You can insert a Cube View using Microsoft Office Update version 1804 and higher.

> **TIP:** If the Document Variable content is shaded after processing and opening in Microsoft Word, update your settings in **File** > **Options** > **Advanced** > **Field shading: Never**.

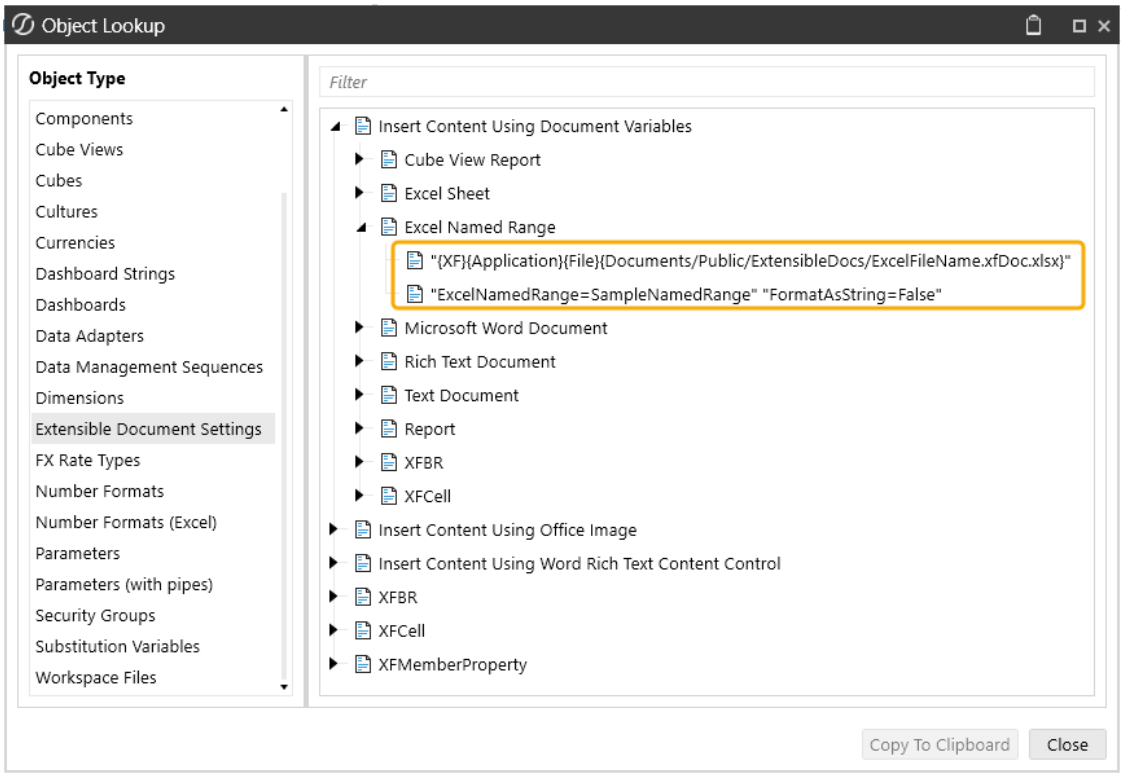
Insert Excel Sheets and Excel Named Ranges in Microsoft Word

> **IMPORTANT:** Excel Sheets and Excel Named Ranges must have 500 or fewer rows and 50 or fewer columns. If you include an argument for an Excel Sheet or Excel Named Range that exceeds the maximum limit, you will receive an error message when attempting to process.

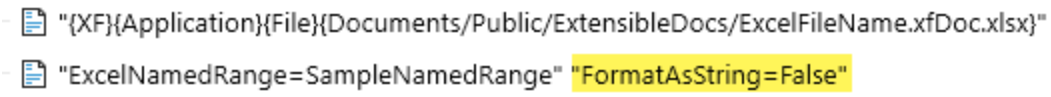
1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views
    - Workspaces
    - Data Management



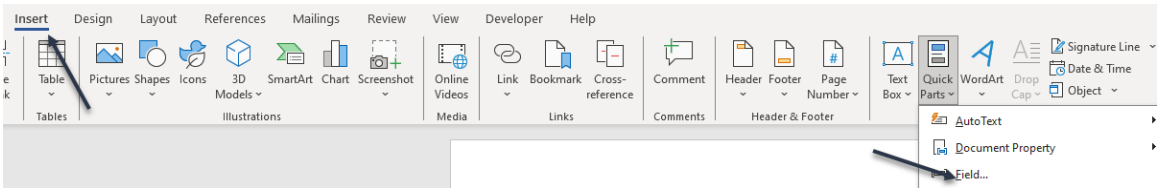
2. In the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Excel Named Range**. Select the line of syntax, and click the **Copy to Clipboard** button to copy the first string.



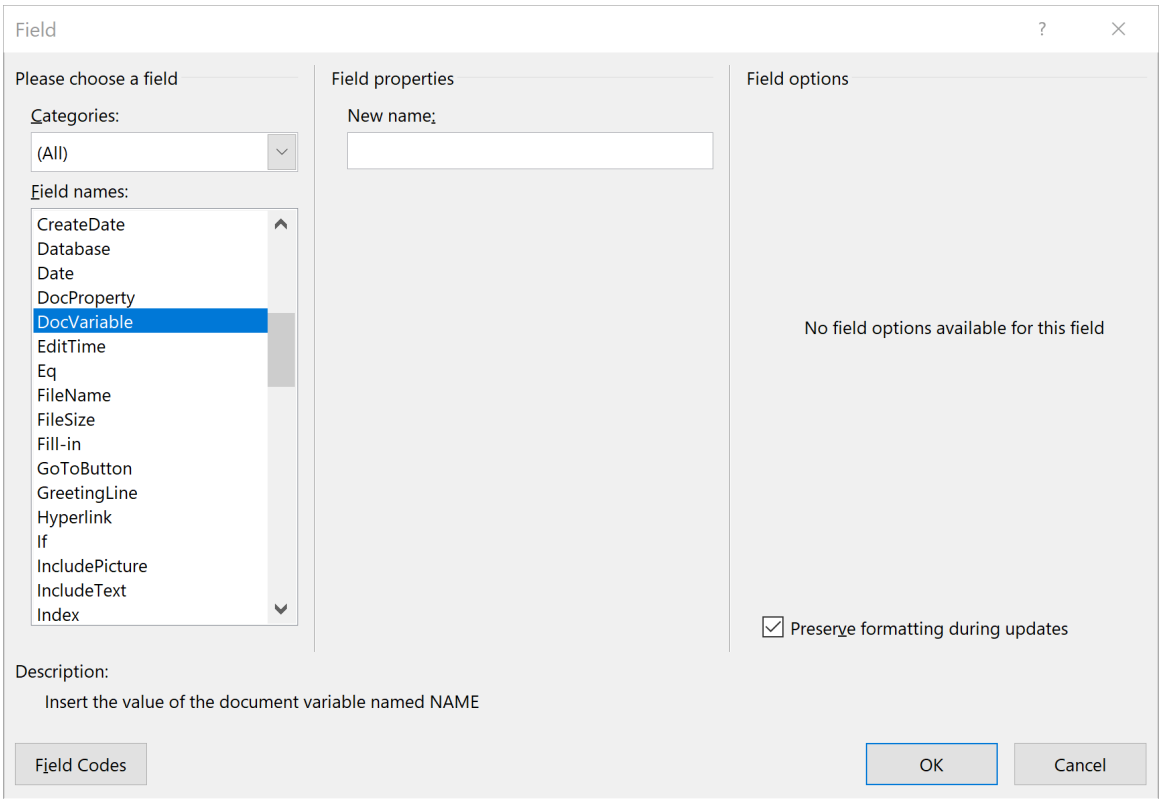
The default value for the FormatAsString parameter is False. When FormatAsString = True, the named range will display as a string instead of a table. If the named range has multiple cells or rows, the cell values will be appended together and each row will display on a new line. If FormatAsString is removed from the document variable, the named range will display as a table as if FormatAsString=False.



3. In Microsoft Word, begin with a new or existing document.
4. To insert your string, go to **Insert** > **Quick Parts** > **Field**.



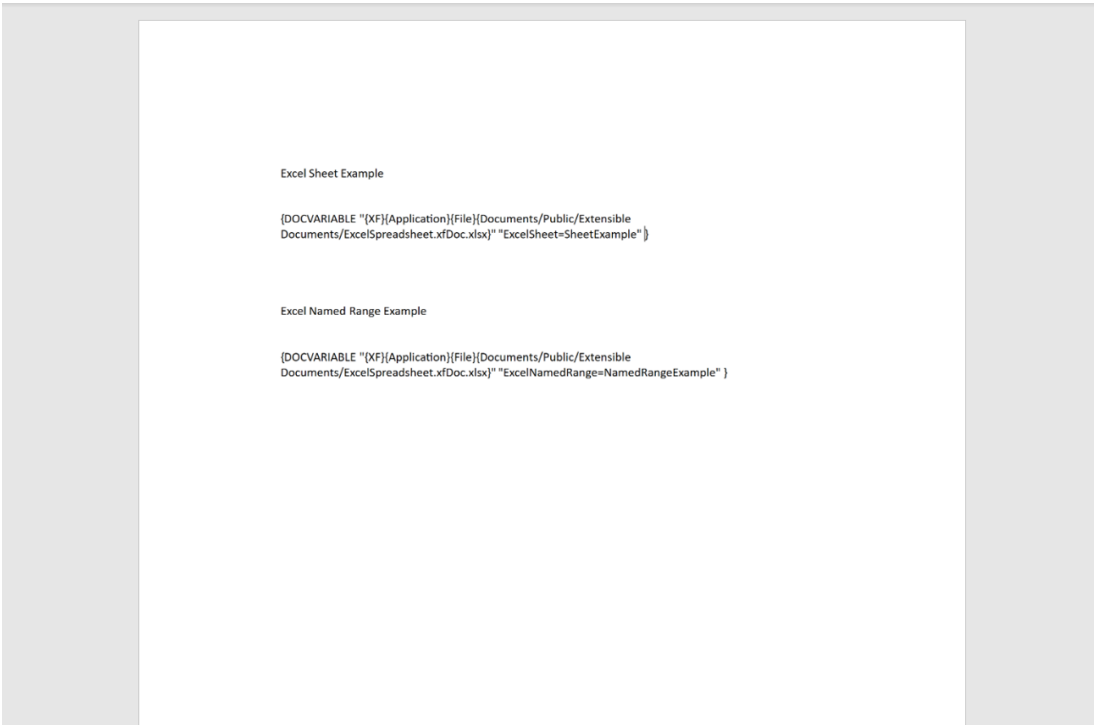
5. A Field dialog box displays. Under **Field Names**, select **DocVariable**. In the **New name** field, paste the string. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/ExcelFileName.xfDoc.xlsx}`" displays in the field.



Optional: After the first string in the DocVariable field, you can paste the "ExcelSheet=Sheet1" or "ExcelNamedRange=SampleNamedRange" arguments. You must separate the arguments by a space. See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under **Field Options**, ensure **Preserve formatting during updates** is selected. See Preserve formatting during updates for more information.

6. Update the values to reflect the folder structure and Excel path. Then in the Field dialog, click the **OK** button. The final string will display in the Microsoft Word document: `{DOCVARIABLE "{XF}{Application}{File}{Documents/Public/Extensible Documents/Excel4.xfDoc.xlsx}"ExcelSheet=Sheet2"}`



> **IMPORTANT:** You must manually update and remove extra " (quotation mark) and \ (backslash) symbols, or the Document Variable will not process correctly.

7. Place your cursor on the page, and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page.

   > **IMPORTANT:** If the expected content does not display after clicking the **OK** button in the Field dialog box, press the **Alt+F9** or **Alt+Fn+F9** keys.

   > **TIP:** If pressing the **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As**. Select a file name and add **.xfdoc** extension to the name. The file name is appended with ".xfdoc.docx".
9. Click **Save** to save the file and ensure it is saved in a familiar folder. For example, Documents).
10. Navigate back to OneStream, and open **File Explorer**. Under the folder of choice, select and upload the file in File Explorer.
11. Right-click on the file and select **"Process and Open"**. You may need to enter or select a parameter and then click **OK**.

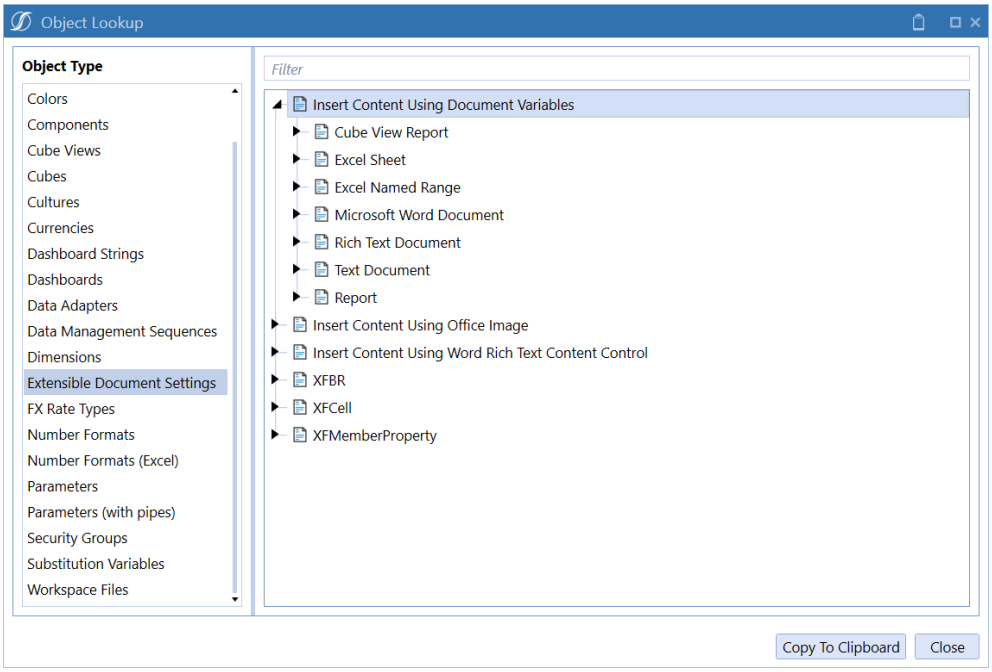
    The Microsoft Word document displays the data being requested by the Excel Sheet or Excel Named Range.

> **TIP:** If the Document Variable content is shaded after processing and opening in Microsoft Word, update your settings in **File** > **Options** > **Advanced** > **Field shading: Never**.

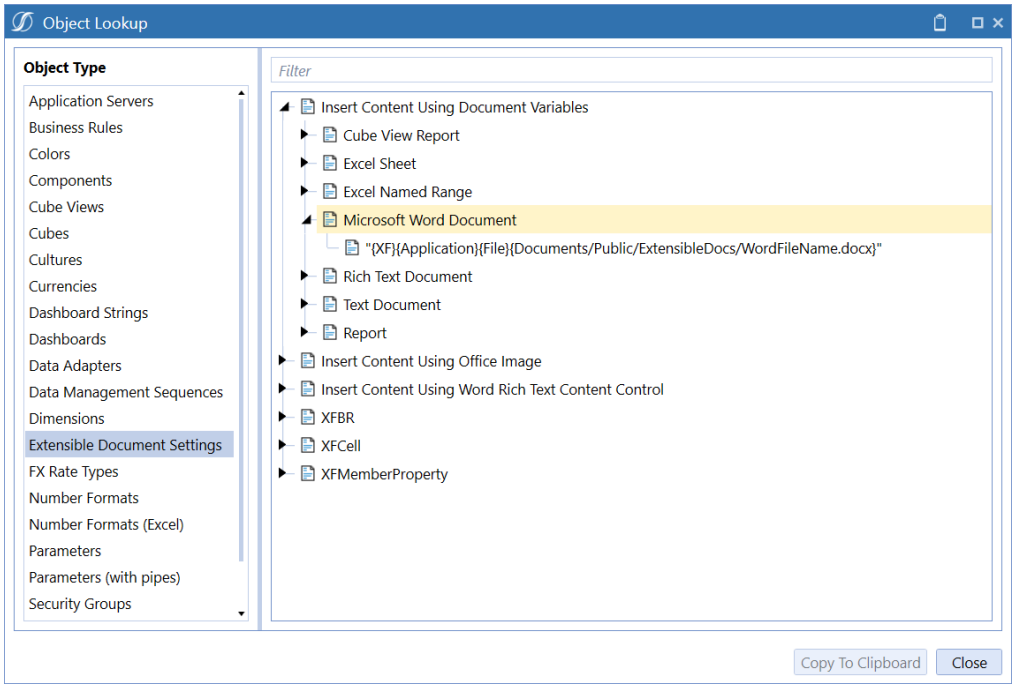
Insert a Microsoft Word Document in Microsoft Word

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:

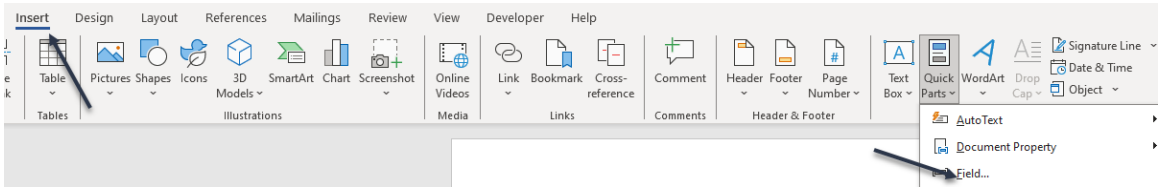
- Form Templates
- Books
- Cube Views
- Workspaces
- Data Management



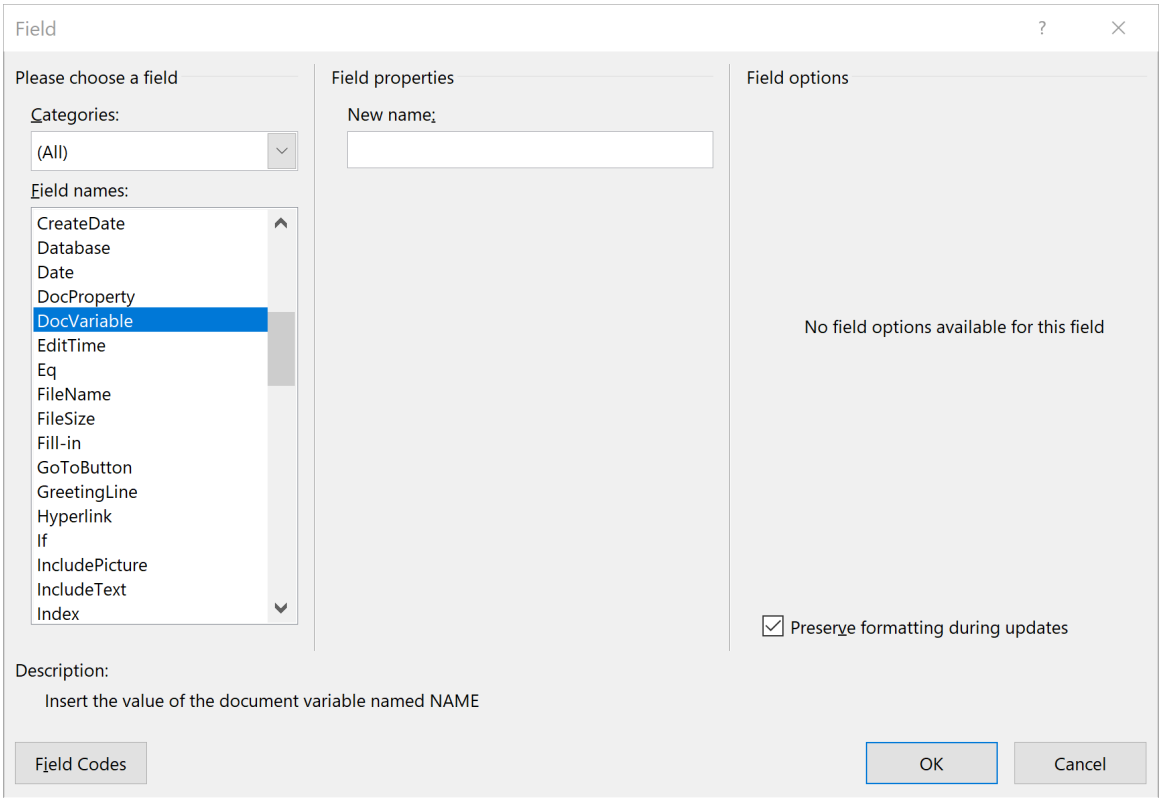
2. In the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Microsoft Word Document**. Select the line of syntax, and click the **Copy to Clipboard** button to copy the first string.



3. In Microsoft Word, begin with a new or existing document.
4. To insert your string, go to **Insert** > **Quick Parts** > **Field**.



5. A Field dialog box displays. Under **Field Names**, select **DocVariable**. In the **New name** field, paste the string. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/WordFileName.docx}`" displays in the field.



See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under **Field Options**, ensure **Preserve formatting during updates** is selected. See Preserve Formatting during Updates for more information.

6. Update the values to reflect the folder structure and the Microsoft Word Document you want to include in the report. Then in the Field dialog, click the **OK** button. The final string will display in the Microsoft Word document: `{ DOCVARIABLE \"{XF}{Application {File}{Documents/Public/ExtensibleDocs/WordFileName.docx}\" \* MERGEFORMAT }`.

> **IMPORTANT:** You must manually update and remove extra " (quotation mark) and \ (backslash) symbols, or the Document Variable will not process correctly.

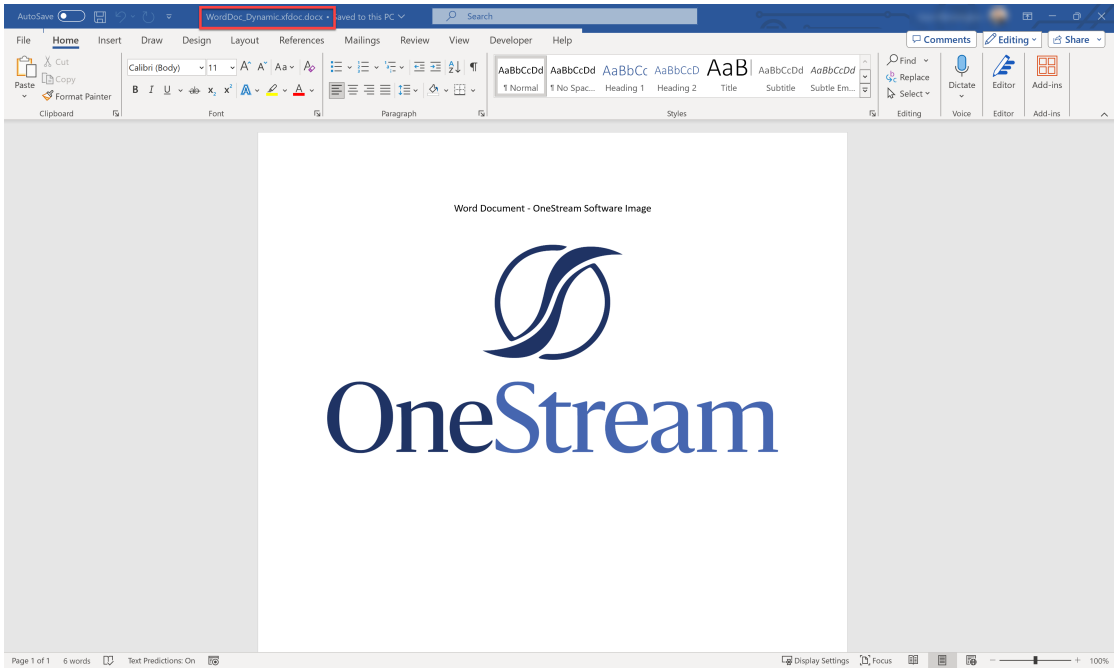
7. Place your cursor on the page, and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page.

   > **IMPORTANT:** If the expected content does not display after clicking the **OK** button in the Field dialog box, press the **Alt+F9** or **Alt+Fn+F9** keys.

   > **TIP:** If pressing the **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As**. Select a file name and add **.xfdoc** extension to the name. The file name is appended with ".xfdoc.docx".
9. Click **Save** to save the file and ensure it is saved in a familiar folder. For example, Documents.
10. Navigate back to OneStream, and open **File Explorer**. Under the folder of choice, select and upload the file in File Explorer.
11. Right-click on the file and select **"Process and Open"**. You may need to enter or select a parameter and then click **OK**.

    The Microsoft Word document displays the data being requested from the referenced Microsoft Word Document.

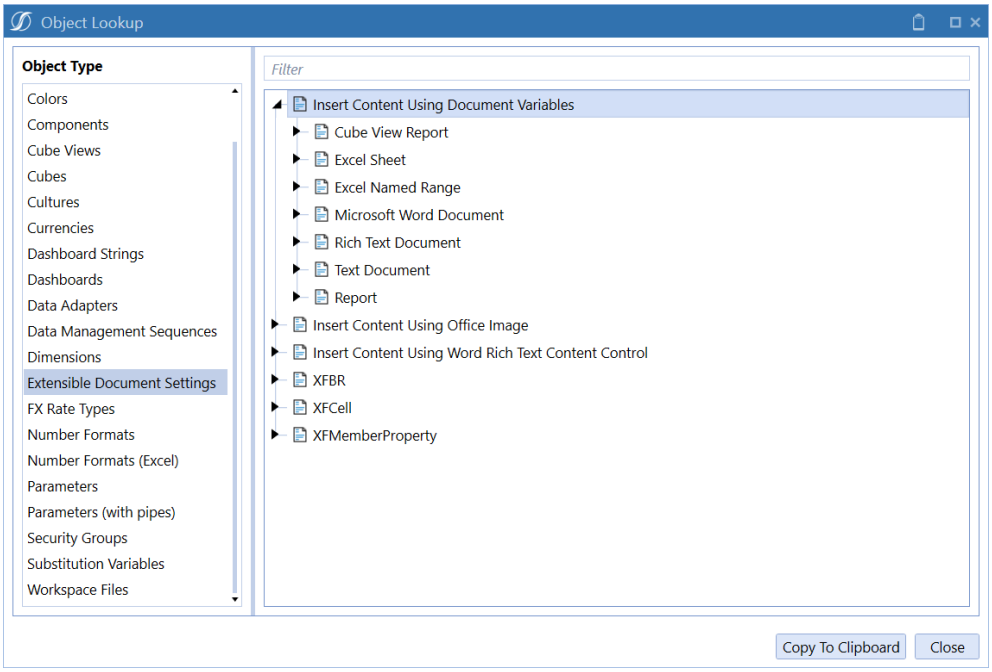


> **TIP:** If the Document Variable content is shaded after processing and opening in Microsoft Word, update your settings in **File** > **Options** > **Advanced** > **Field shading: Never**.

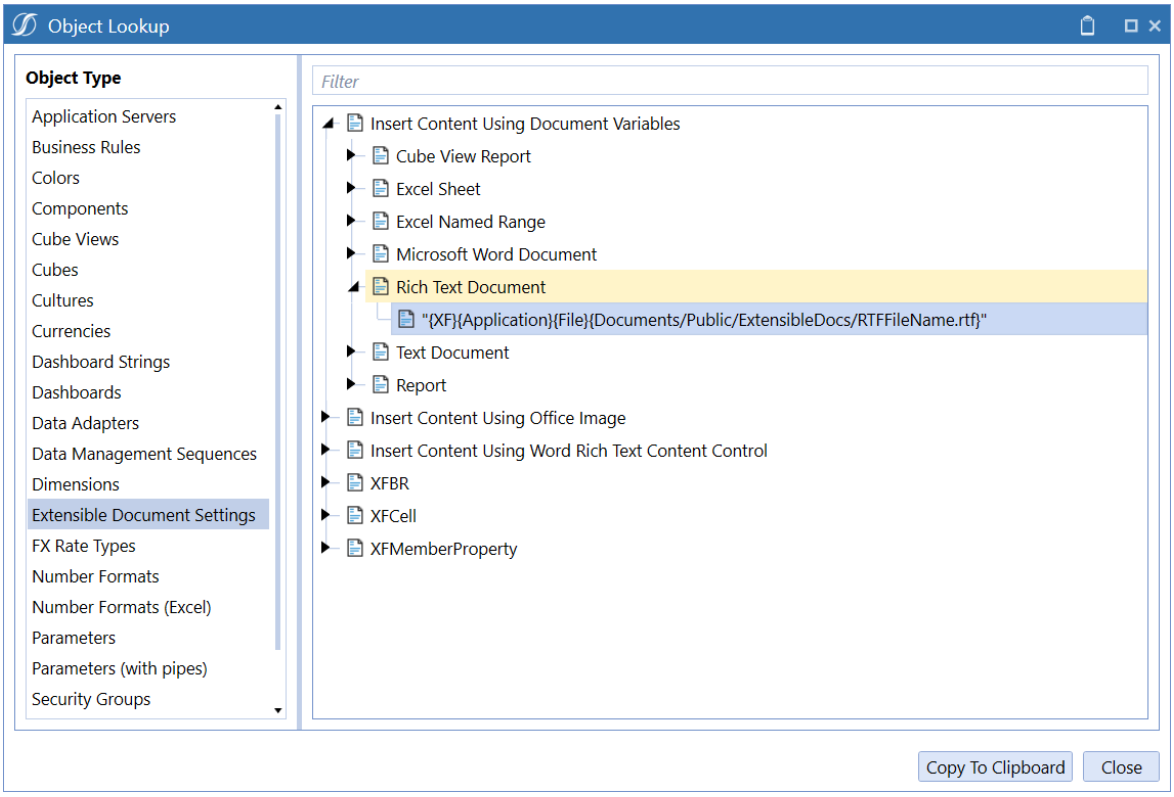
Insert a Rich Text Document in Microsoft Word

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views

- Workspaces
- Data Management

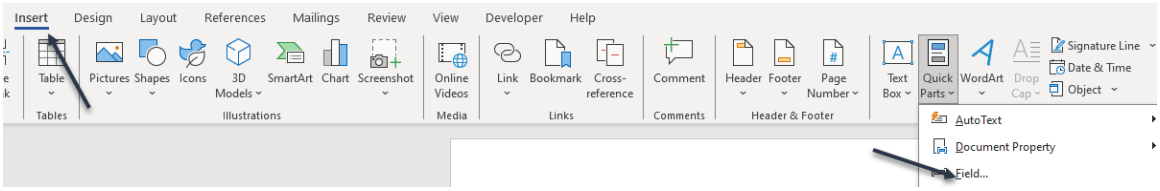


2. In the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Rich Text Document**. Select the line of syntax and click the **Copy to Clipboard** button to copy the string.

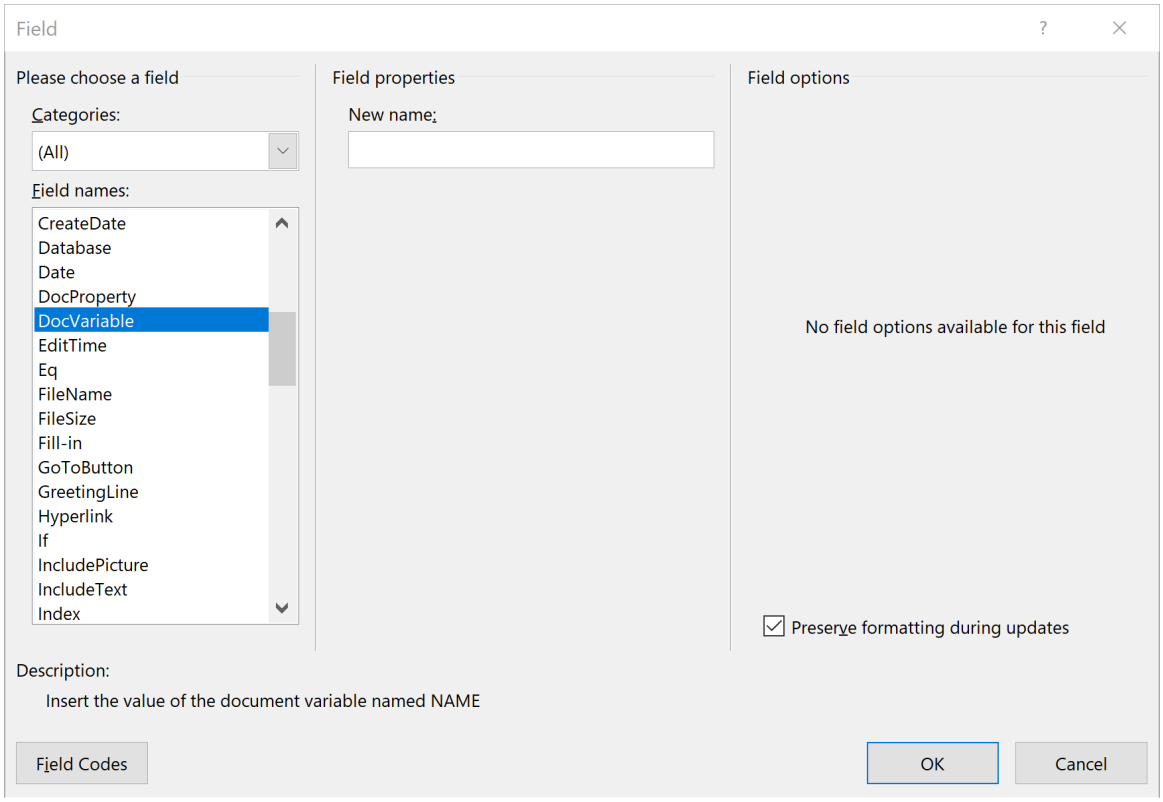


3. In Microsoft Word, begin with a new or existing document.

4. To insert your string, go to **Insert** > **Quick Parts** > **Field**.



5. A Field dialog box displays. Under **Field Names**, select **DocVariable**. In the **New name** field, paste the string. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/RTFFileName.rtf}`" displays in the field. See Extensible Document Settings for more examples on these different item types.



> **TIP:** Under **Field Options**, ensure **Preserve formatting during updates** is selected. See Preserve Formatting during Updates for more information.

6. Update the values to reflect the folder structure and the rich text document path. Then in the Field dialog, click the **OK** button. The final string will display in the Microsoft Word document: `{DOCVARIABLE"{XF}{Application}{File}{Documents/Public/Extensible Documents/RichTextFormatDoc.rtf}"}`

   > **IMPORTANT:** You must manually update and remove the \ (backslash) symbols, or the Document Variable will not process correctly.

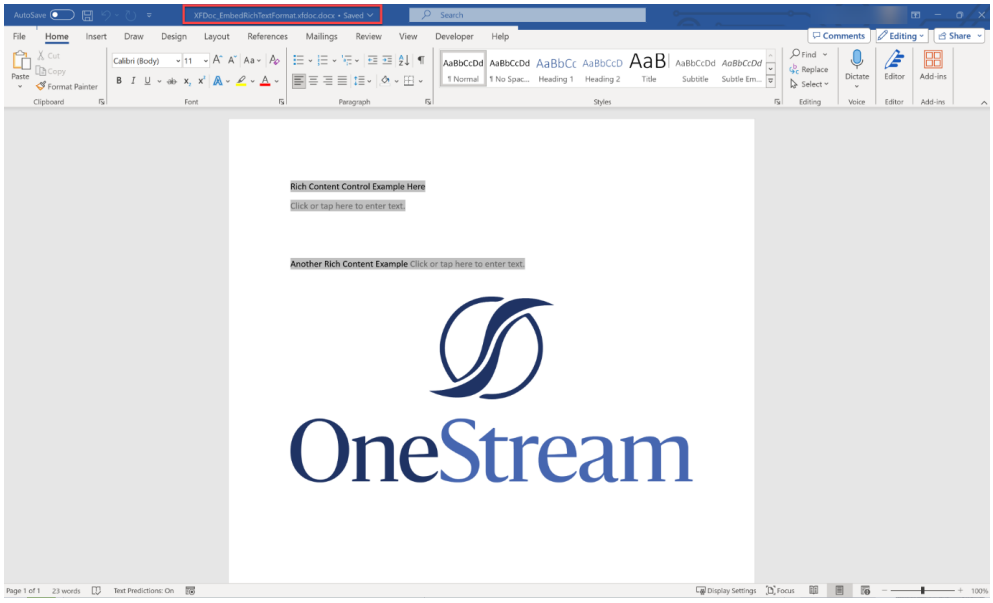
7. Place your cursor on the page, and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page.

> **IMPORTANT:** If the expected content does not display after clicking the **OK** button in the Field dialog box, press the **Alt+F9** or **Alt+Fn+F9** keys.

> **TIP:** If pressing the **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As**. Select a file name and add **.xfdoc** extension to the name. The file name is appended with ".xfdoc.docx".
9. Click **Save** to save the file and ensure it is saved in a familiar folder. For example, Documents.
10. Navigate back to OneStream, and open **File Explorer**. Under the folder of choice, select and upload the file in File Explorer.
11. Right-click on the file and select **"Process and Open"**. You may need to enter or select a parameter and then click **OK**.

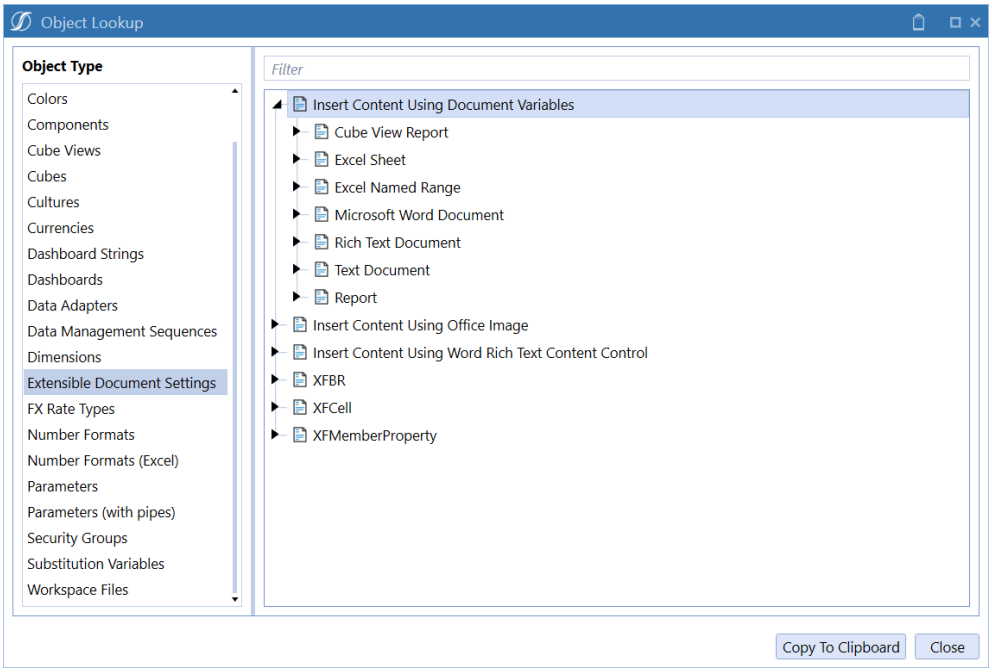
    The Microsoft Word document displays the data being requested for the Rich Text Document.



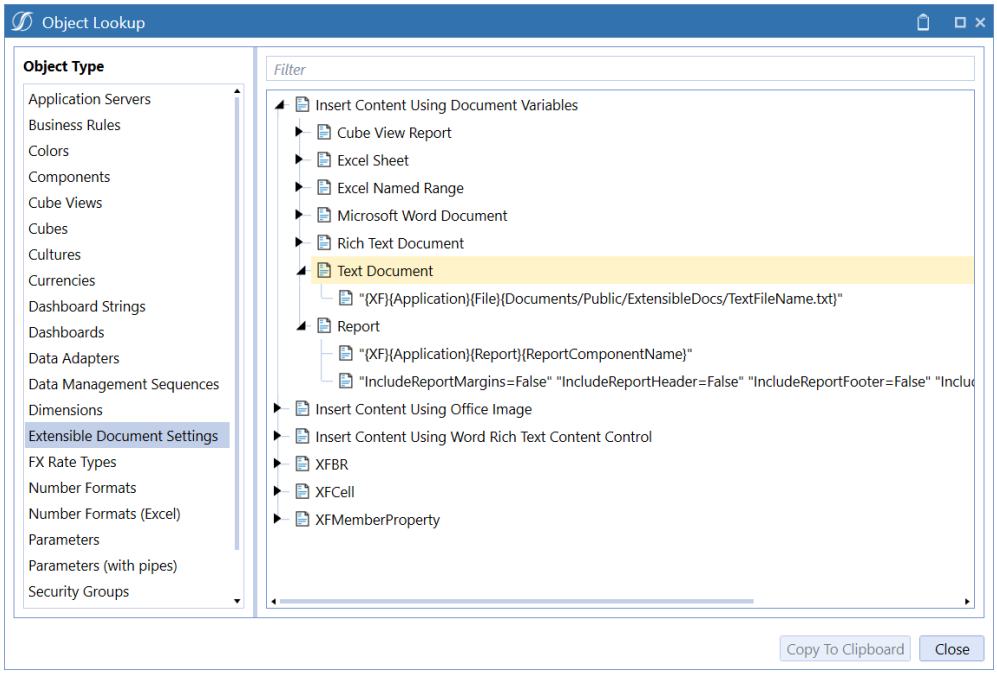
> **TIP:** If the Document Variable content is shaded after processing and opening in Microsoft Word, update your settings in **File** > **Options** > **Advanced** > **Field shading: Never**.

Insert a Text Document in Microsoft Word

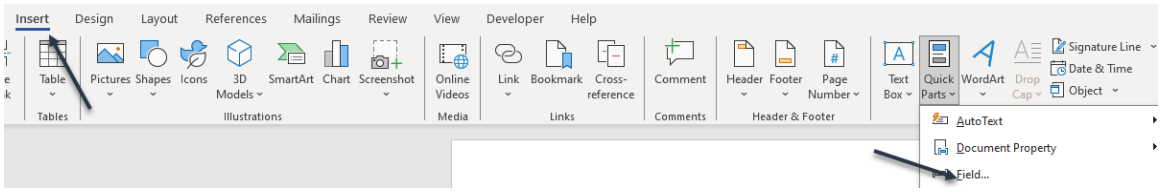
1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views
    - Workspaces
    - Data Management



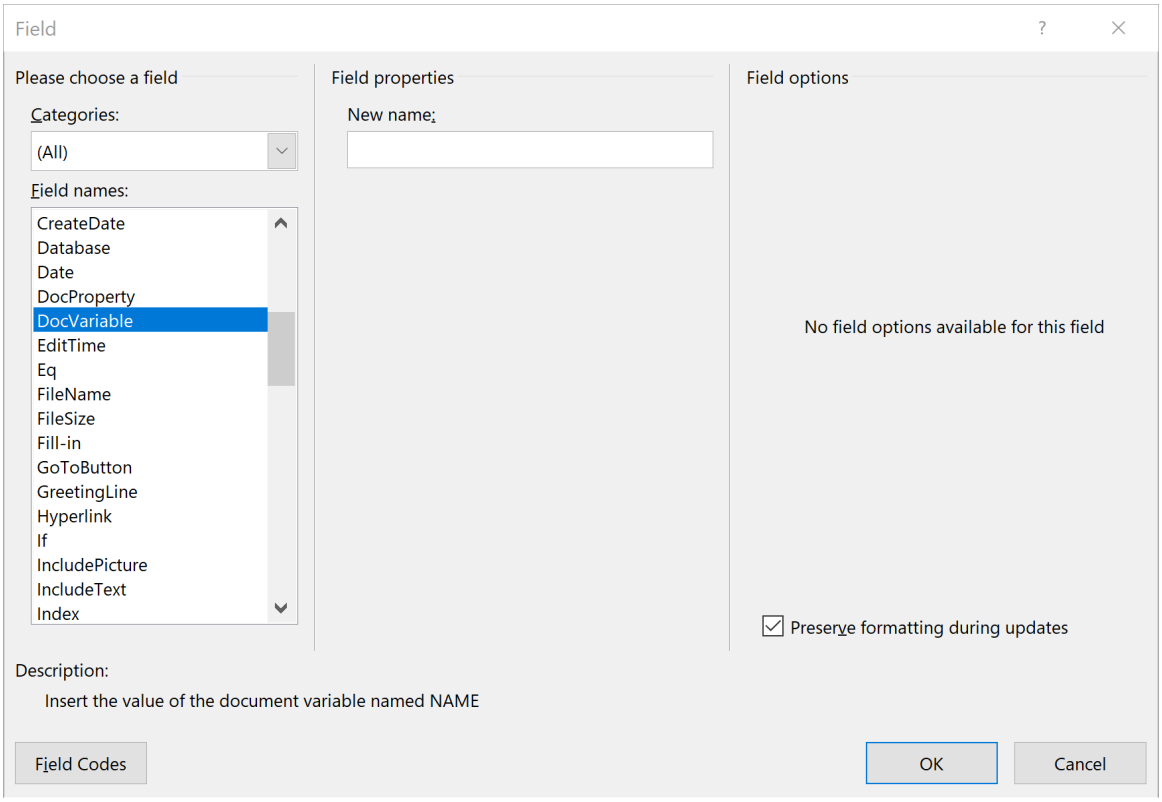
2. In the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Text Document**. Select the line of syntax and click the **Copy to Clipboard** button to copy the string.



3. In Microsoft Word, begin with a new or existing document.
4. To insert your string, go to **Insert** > **Quick Parts** > **Field**.



5. A Field dialog box displays. Under **Field Names**, select **DocVariable**. In the **New name** field, paste the string. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/TextFileName.txt}`" displays in the field.



See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under **Field Options**, ensure **Preserve formatting during updates** is selected. See Preserve Formatting during Updates for more information.

6. Update the values to reflect the folder structure and text file path. Then in the Field dialog, click the **OK** button. The final string will display in the Microsoft Word document:
`{DOCVARIABLE "{XF}{Application}{File}{Documents/Public/Extensible Documents/Testtoupdate.txt}"}`

> **IMPORTANT:** You must manually update and remove the \ (backslash) symbols, or the Document Variable will not process correctly.

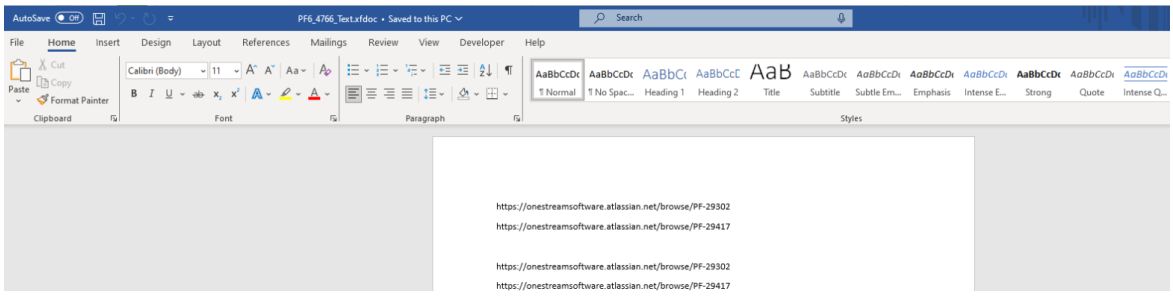
7. Place your cursor on the page, and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page.

   > **IMPORTANT:** If the expected content does not display after clicking the **OK** button in the Field dialog box, press the **Alt+F9** or **Alt+Fn+F9** keys.

   > **TIP:** If pressing the **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As**. Select a file name and add **.xfdoc** extension to the name. The file name is appended with ".xfdoc.docx".
9. Click **Save** to save the file and ensure it is saved in a familiar folder. For example, Documents.
10. Navigate back to OneStream, and open **File Explorer**. Under the folder of choice, select and upload the file in File Explorer.
11. Right-click on the file and select **"Process and Open"**. You may need to enter or select a parameter and click **OK**.

    The Microsoft Word document displays the data being requested from the Text File content.

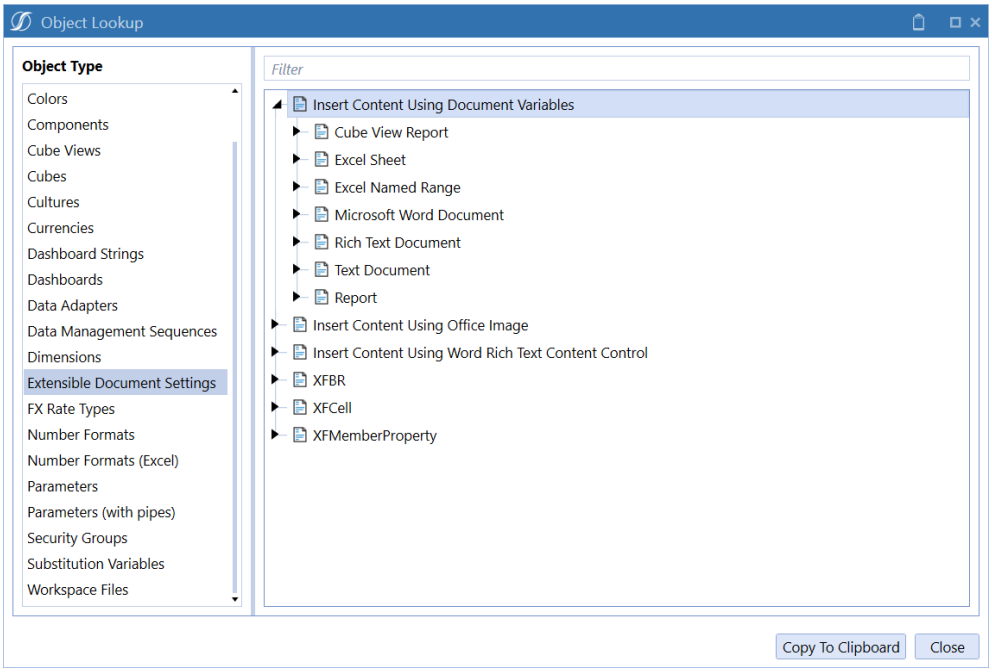


> **TIP:** If the Document Variable content is shaded after processing and opening in Microsoft Word, update your settings in **File** > **Options** > **Advanced** > **Field shading: Never**.

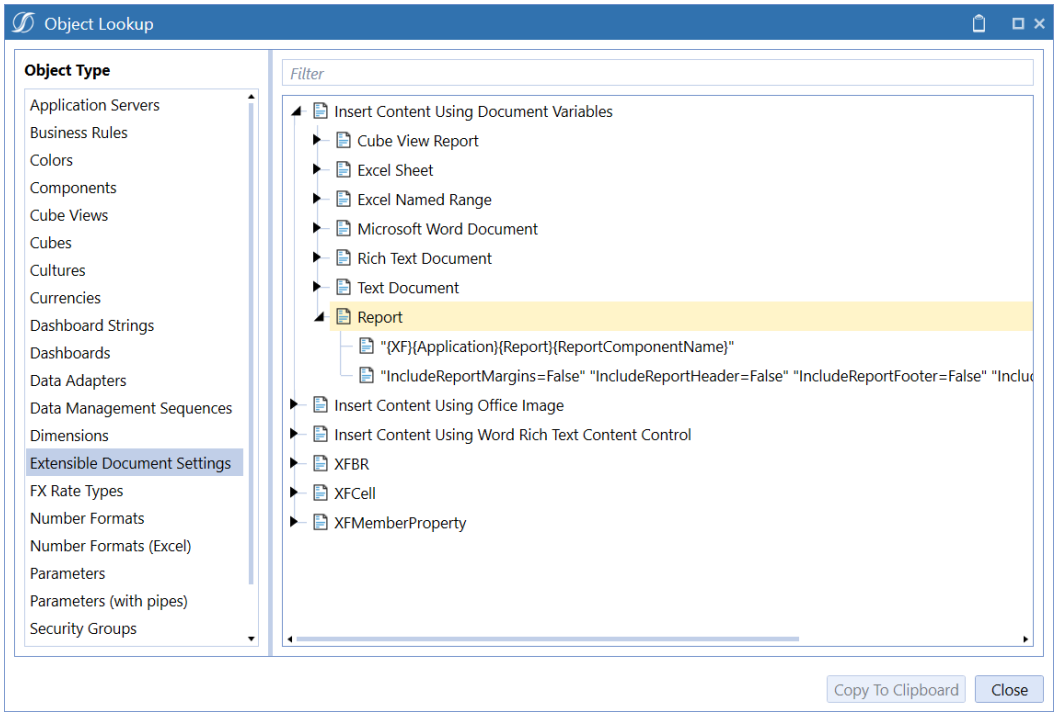
Insert a Report in Microsoft Word

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:

    - Form Templates
    - Books
    - Cube Views
    - Workspaces
    - Data Management

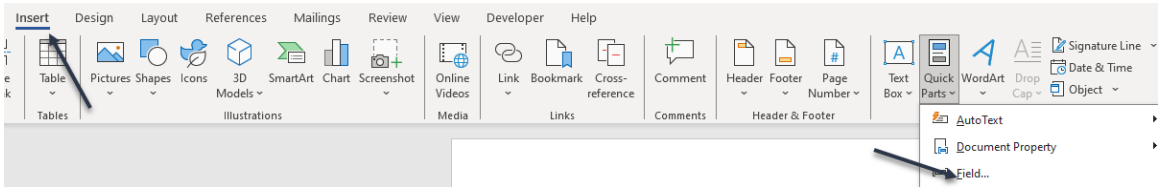


2. In the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Report**. Select the line of syntax, and click the **Copy to Clipboard** button to copy the first string.

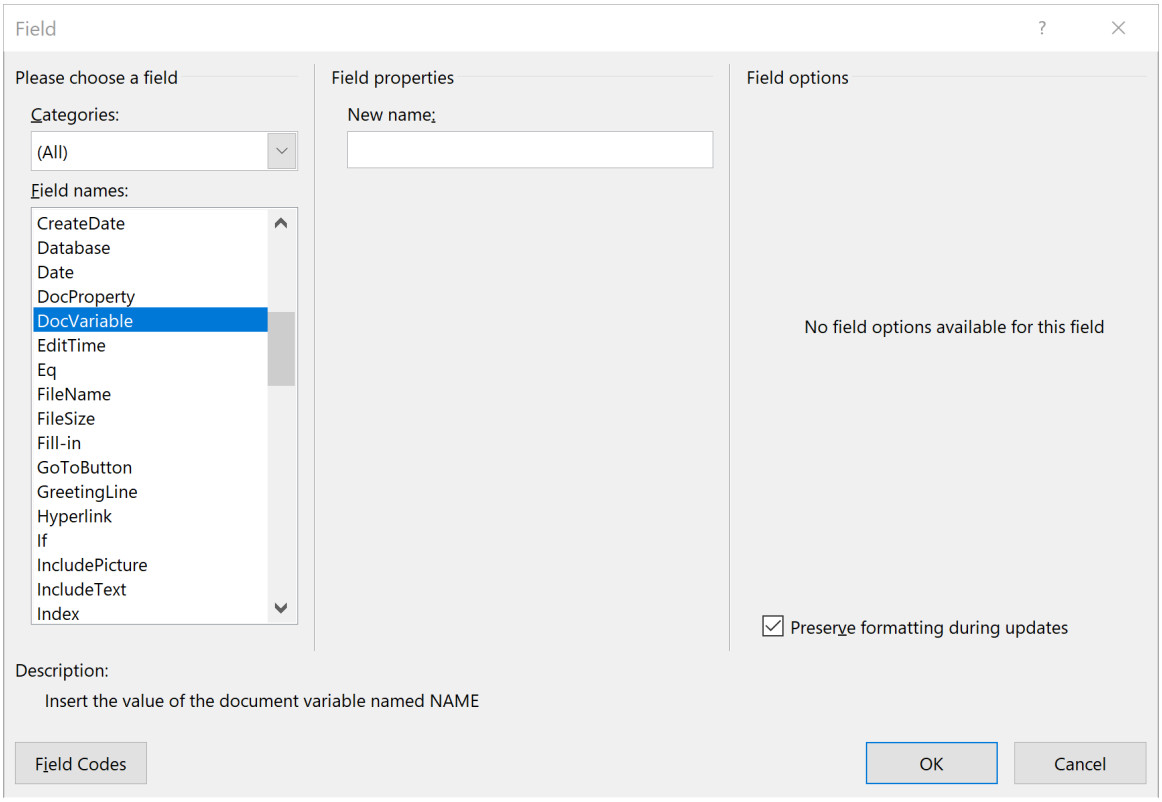


3. In Microsoft Word, begin with a blank or existing document.

4. To insert your string, go to **Insert** > **Quick Parts** > **Field**.



5. A Field dialog box displays. Under **Field Names**, select **DocVariable**. In the **New name** field, paste the string. The string "`{XF}{Application}{Report}{ReportComponentName}`" displays in the field.



Optional: After the first string in the DocVariable field, you can paste the "IncludeReportMargins=False" arguments. You must separate the arguments by a space. See Extensible Document Settingsfor more examples on these different item types.

> **TIP:** Under **Field Options**, ensure **Preserve formatting during updates** is selected. See Preserve Formatting during Updates for more information.

6. Update the values to reflect the report component name. Then in the Field dialog, click the **OK** button. The final string will display in the Microsoft Word document: `{DOCVARIABLE "{XF}{Application}{Report}{Report3}"}`

   > **IMPORTANT:** You must manually update and remove the \ (backslash) symbols, or the Document Variable will not process correctly.

7. Place your cursor on the page, and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page.

   > **IMPORTANT:** If the expected content does not display after clicking the **OK** button in the Field dialog box, press the **Alt+F9** or **Alt+Fn+F9** keys.

   > **TIP:** If pressing the **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As**. Select a file name and add **.xfdoc** extension to the name. The file name is appended with ".xfdoc.docx".
9. Click **Save** to save the file and ensure it is saved in a familiar folder. For example, Documents.
10. Navigate back to OneStream, and open **File Explorer**. Under the folder of choice, select and upload the file in File Explorer.
11. Right-click on the file and select **"Process and Open"**. You may need to enter or select a parameter and click **OK**.

    The Microsoft Word document displays the embedded report.

> **TIP:** If the Document Variable content is shaded after processing and opening in Microsoft Word, update your settings in **File** > **Options** > **Advanced** > **Field shading: Never**.

Creating an Extensible Document in Text Editor

You can easily embed refreshable content to allow users to edit and narrate a document for the purposes of publishing it into a report. This allows you to embed and refresh content using Document Variables within Text Editor through the following actions:

- Insert a Cube View in Text Editor
- Insert an Excel Sheet/Excel Named Range in Text Editor
- Insert a Microsoft Word Document in Text Editor
- Insert a Rich Text Document in Text Editor
- Insert a Text Document in Text Editor
- Insert a Report in Text Editor

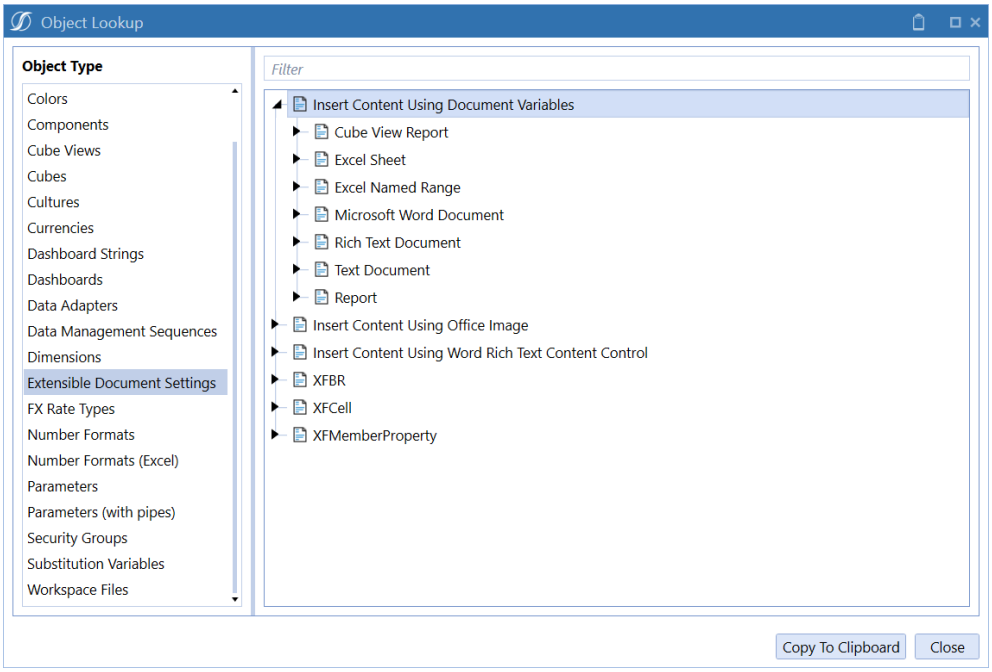
> **NOTE:** Text Editor contains the OneStream ribbon. See Text Editor Ribbon for more information.

> **IMPORTANT:** Copy all syntax needed to insert content using document variables directly from Object Lookup dialog box. See Extensible Document Settings .

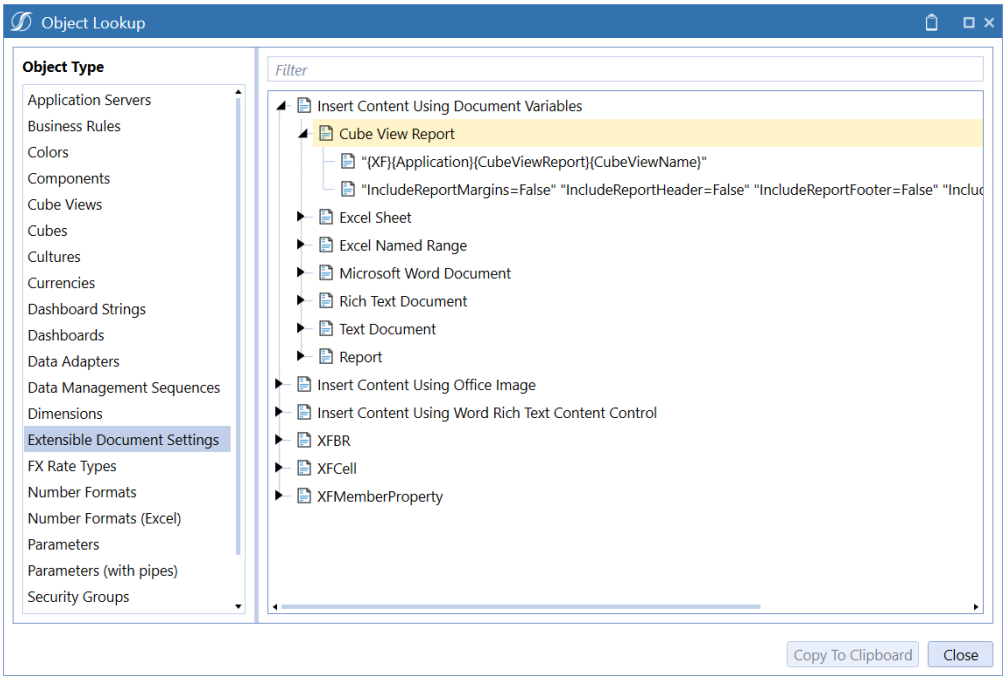
Insert a Cube View in Text Editor

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views

- Workspaces
- Data Management

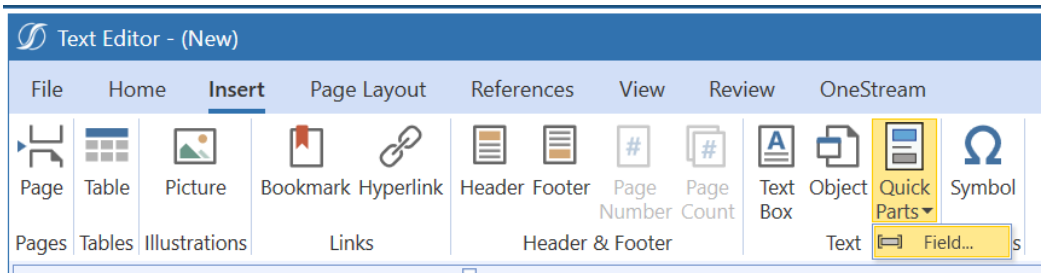


2. Once in the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Cube View Report**. Select the line of syntax and click **Copy to Clipboard** to copy the first string.

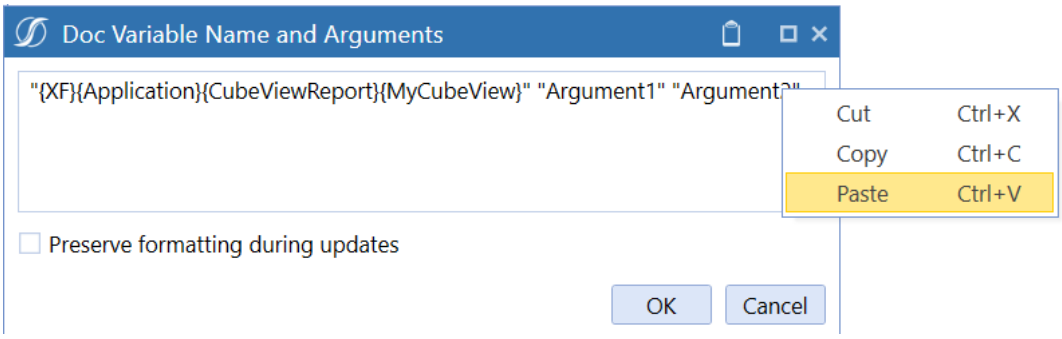


3. In the Windows Application, under Tools, click **Text Editor**. Begin with a new or existing file in Text Editor.

4. To inset your string, go to **Insert>Quick Parts> Field**.



5. A **Doc Variable Name and Arguments** dialog box appears. Highlight the placeholder text and paste the copied syntax. The string "`{XF}{Application}{CubeViewReport}{CubeViewName}`" displays in the field.

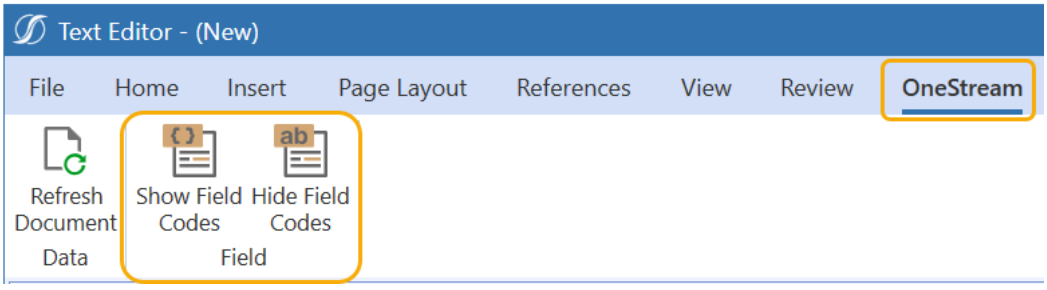


Optional: Cube View report arguments, such as "IncludeReportMargins=False", can be pasted in "Argument1" "Argument2" sections. See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under Field Options, preserve formatting during updates is unchecked. See Preserve Formatting during Updates for more information.

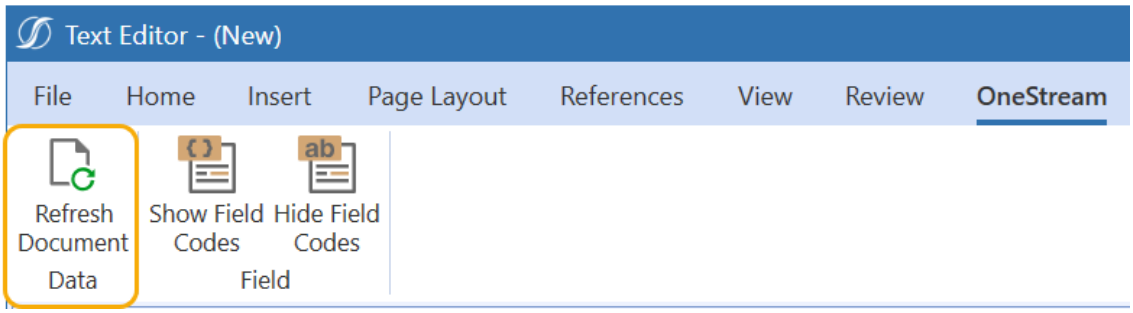
6. Update the values to reflect the cube view you will add to the report. Then in the Field dialog, click the **OK** button. The final string will display in Text Editor: `{DOCVARIABLE "{XF}{Application}{CubeViewReport}{IncomeStatement}.`

7. Place your cursor on the page and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page. You can also use the **Show Field Codes** and **Hide Field Codes** buttons under the OneStream Ribbon to show or hide field codes such as Document Variables.



> **TIP:** If pressing **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As** and select **Save As File in OneStream File System**. Select a specific folder in the File Explorer, then enter a File Name and add the **.xfdoc** extension. The file name is appended with ".xfdoc.docx".
9. In the OneStream ribbon, click the **Refresh Document** toolbar button.



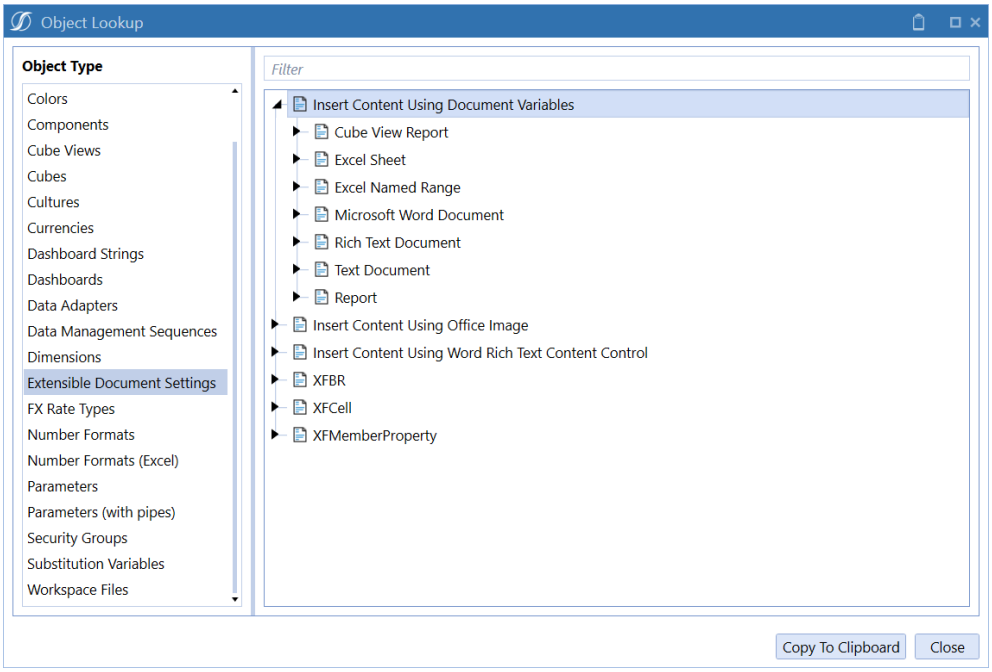
The Text Editor page refreshes and the embedded Cube View report is displayed.

Excel Sheets and Excel Named Ranges in Text Editor

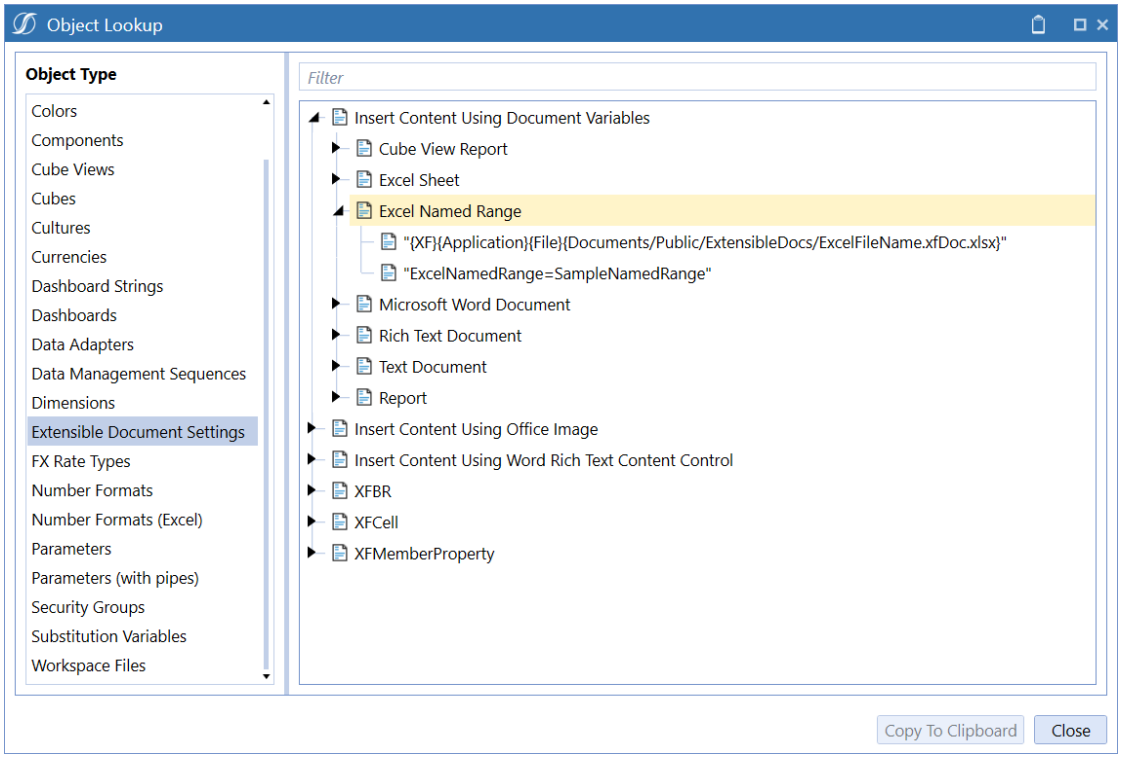
> **IMPORTANT:** Excel Sheets and Excel Named Ranges can only have 500 or less rows and 50 or less columns. If you include an Argument for an Excel Sheet or Excel Named Range that exceeds the maximum limit, you will receive an error message when attempting to process.

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:

- Form Templates
- Books
- Cube Views
- Workspaces
- Data Management

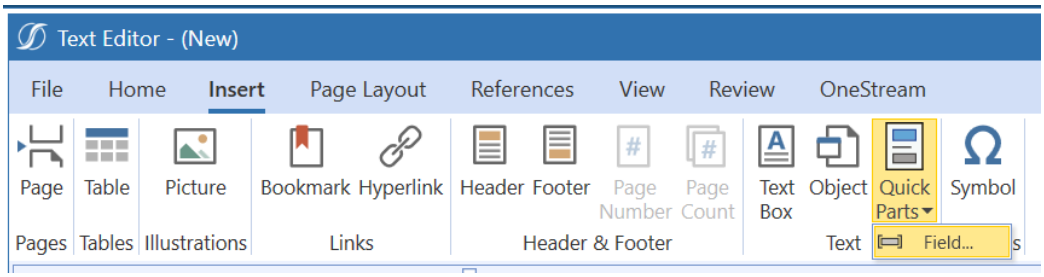


2. Once in the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Excel Name Ranged**. Select the line of syntax and click **Copy to Clipboard** to copy the first string.

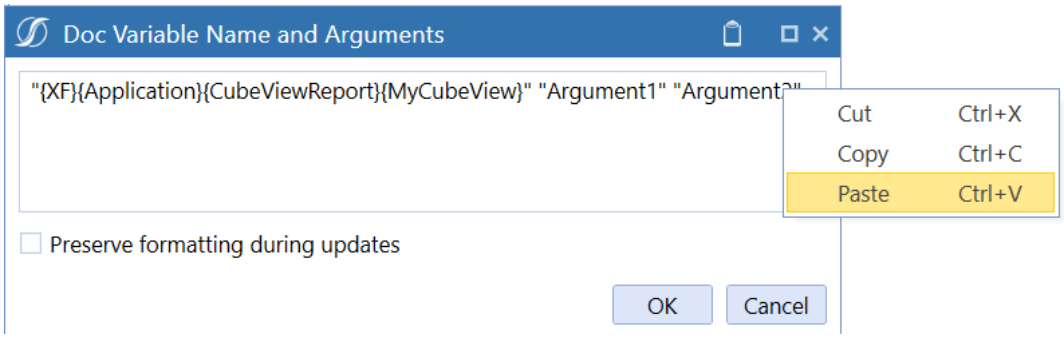


3. In the Windows Application, under Tools, click **Text Editor**. Begin with a new or existing file in Text Editor.

4. To inset your string, go to **Insert>Quick Parts> Field**.



5. A **Doc Variable Name and Arguments** dialog box appears. Highlight the placeholder text and paste the copied syntax. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/ExcelFileName.xfDoc.xlsx}`" displays in the field.

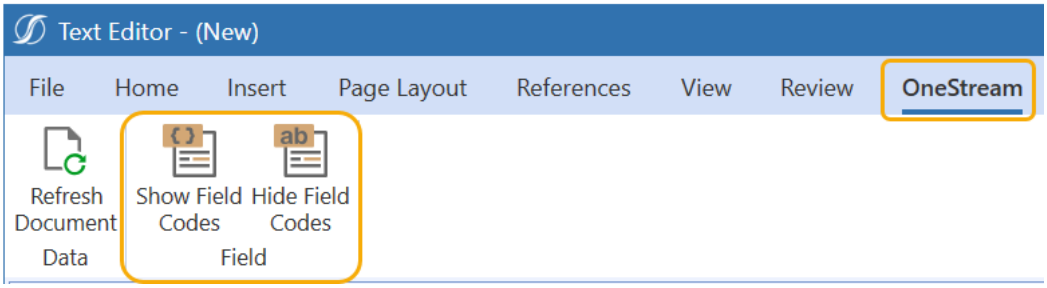


   Optional: The "ExcelSheet=Sheet1" or "ExcelNamedRange=SampleNamedRange" arguments can be pasted in "Argument1" "Argument2" sections. See Extensible Document Settings for more examples on these different item types.

   > **TIP:** Under Field Options, preserve formatting during updates is unchecked. See Preserve Formatting during Updates for more information.

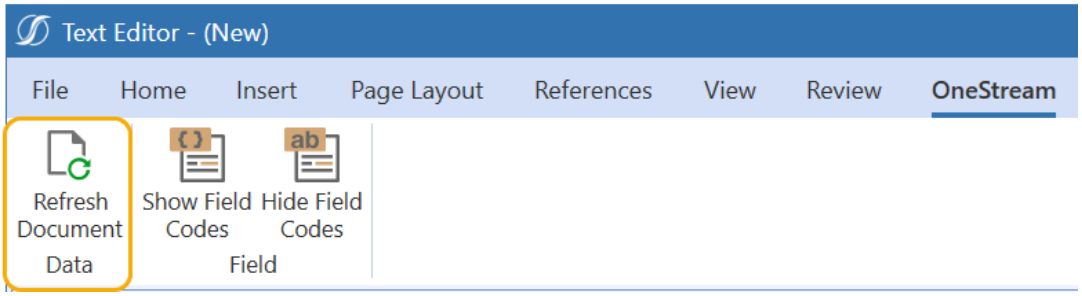
6. Update the values to reflect the folder structure and excel file path. Then in the Field dialog, click the **OK** button. The final string will display in Text Editor: `{DOCVARIABLE "{XF}{Application}{File}{Documents/Public/Extensible Documents/ExcelTextEditor.xfDoc.xlsx}" "ExcelSheet=Sheet2"`

7. Place your cursor on the page and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page. You can also use the **Show Field Codes** and **Hide Field Codes** buttons under the OneStream Ribbon to show or hide field codes such as Document Variables.

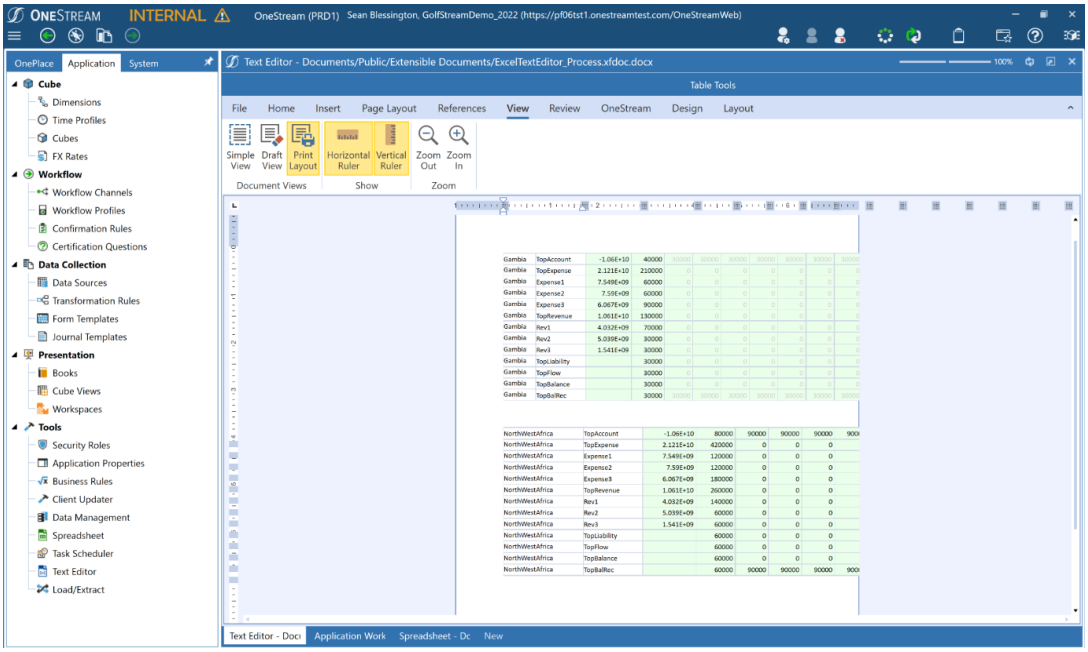


> **TIP:** If pressing **ALT+F9** keys does not work, press **ALT+Fn+F9** simultaneously instead.

8. Select **File and Save As** and select **Save As File in OneStream File System**. Select a specific folder in the File Explorer, then enter a File Name and add **.xfdoc** extension. The file name is appended with ".xfdoc.docx".
9. In the OneStream ribbon, click the **Refresh Document** toolbar button.



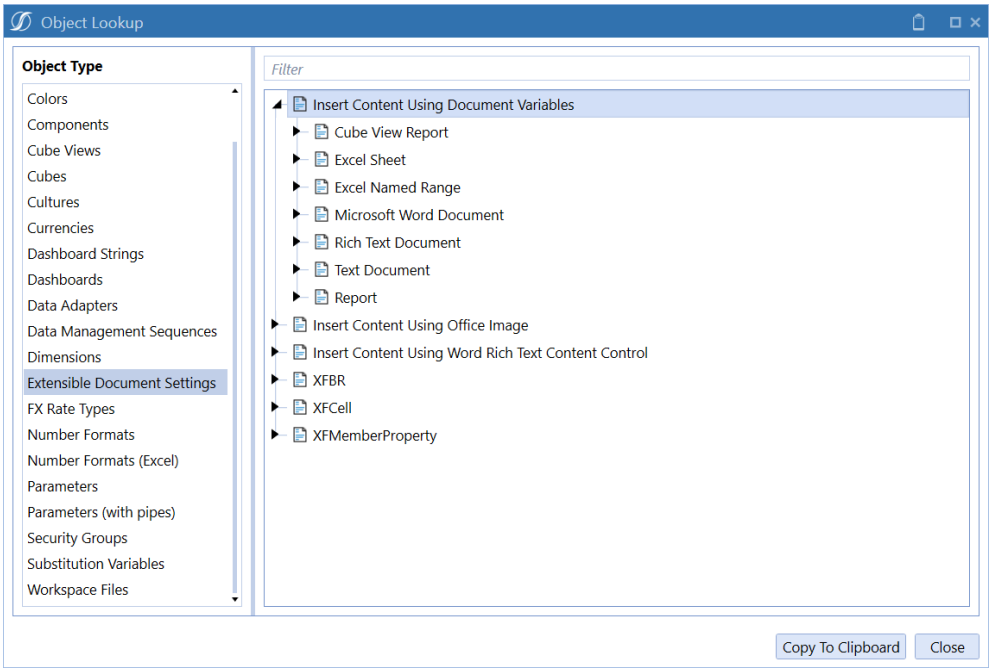
The Text Editor page refreshes and you see the Excel Sheet or Excel Named Range content displayed in place of the Document Variables.



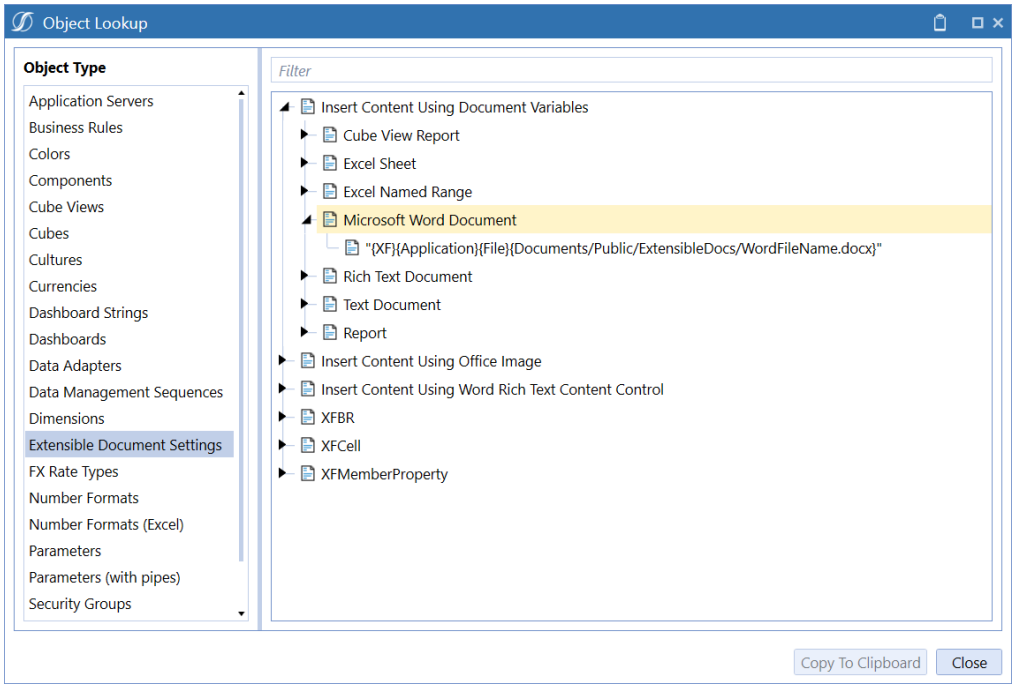
Insert a Microsoft Word Document in Text Editor

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:

    - Form Templates
    - Books
    - Cube Views
    - Workspaces
    - Data Management

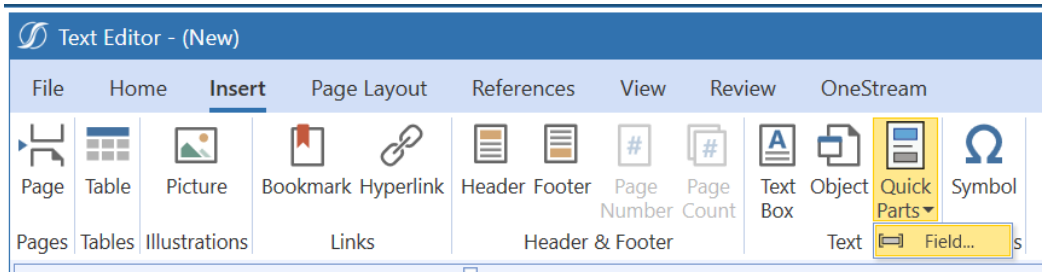


2. Once in the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and expand **Microsoft Word Document**. Select the line of syntax and click **Copy to Clipboard** to copy the string.

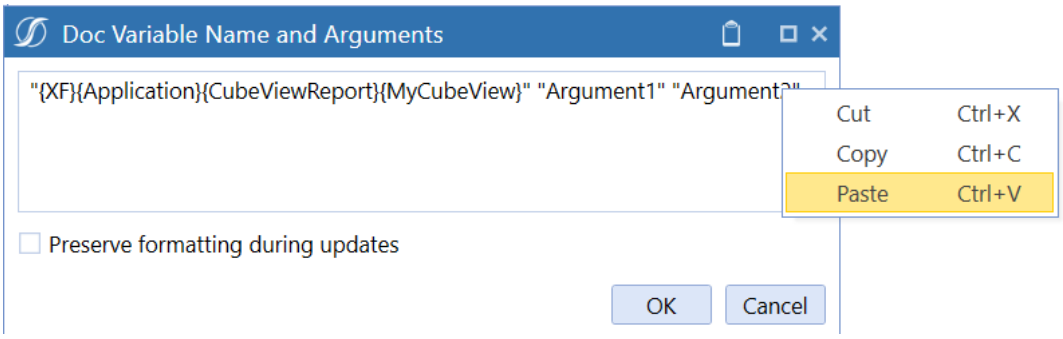


3. In the Windows Application, under Tools, click **Text Editor**. Begin with a new or existing file in Text Editor.

4. To inset your string, go to **Insert**>**Quick Parts**> **Field**.



5. A **Doc Variable Name and Arguments** dialog box appears. Highlight the placeholder text and paste the copied syntax. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/WordFileName.docx}`" displays in the field.

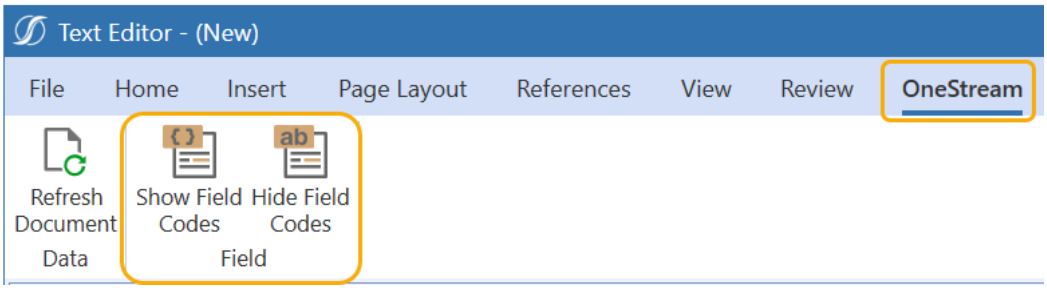


See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under Field Options, preserve formatting during updates is unchecked. See Preserve Formatting during Updates for more information.

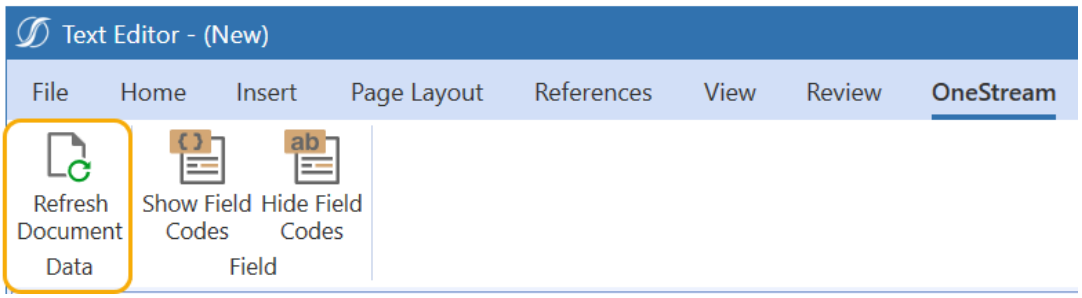
6. Update the values to reflect the folder structure and the word document path. Then in the Field dialog, click the **OK** button. The final string will display in Text Editor: `{DOCVARIABLE "{XF}{Application}{File}{Documents/Public/Extensible Documents/OneStreamWordDocument.docx}"}`.

7. Place your cursor on the page and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page. You can also use the **Show Field Codes** and **Hide Field Codes** buttons under the OneStream Ribbon to show or hide field codes such as Document Variables.

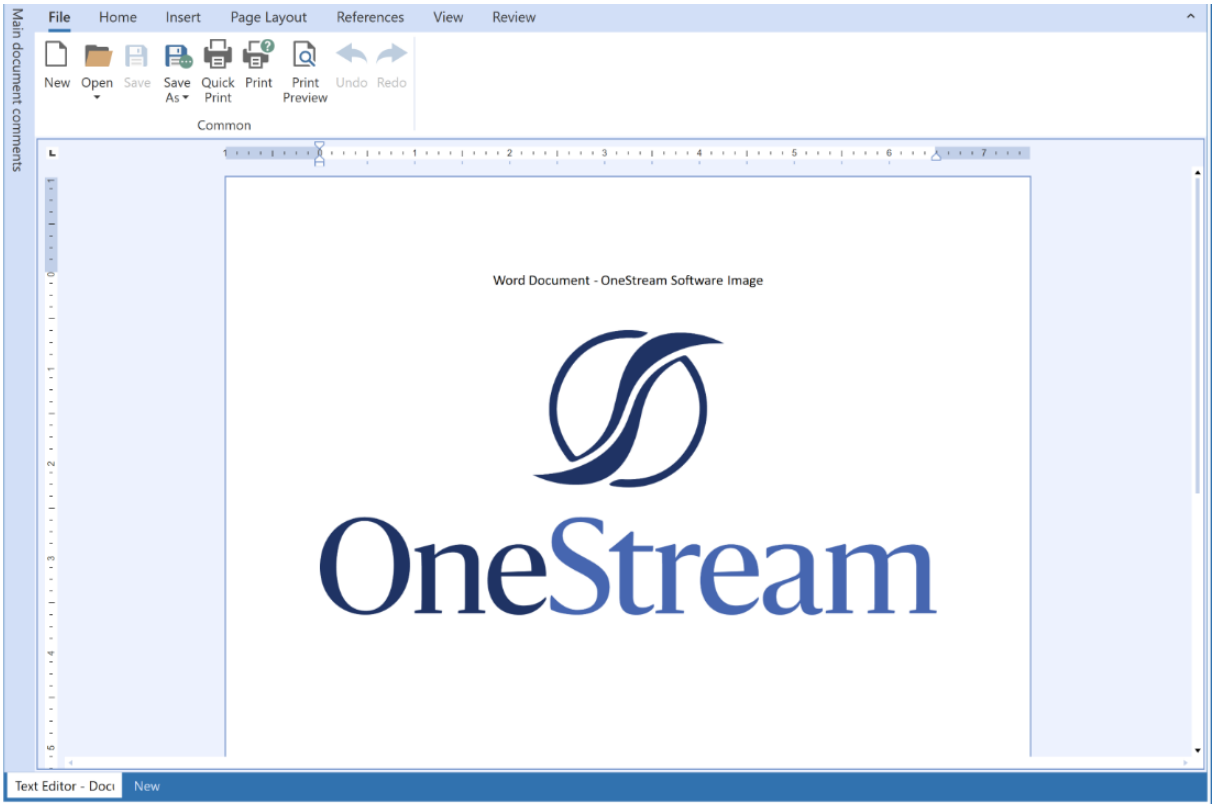


> **TIP:** If pressing **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As** and select **Save As File in OneStream File System**. Select a specific folder in the File Explorer, then enter a Name and add **.xfdoc** extension. The file name is appended with ".xfdoc.docx".
9. In the OneStream ribbon, click the **Refresh Document** toolbar button.



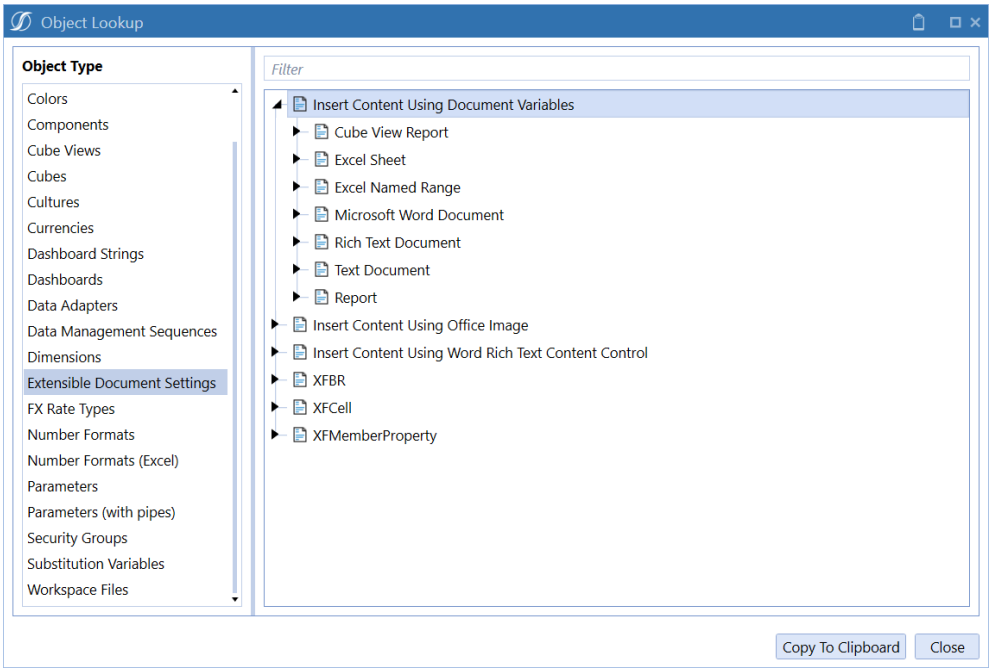
The Text Editor page refreshes and you see the embedded Microsoft Word document.



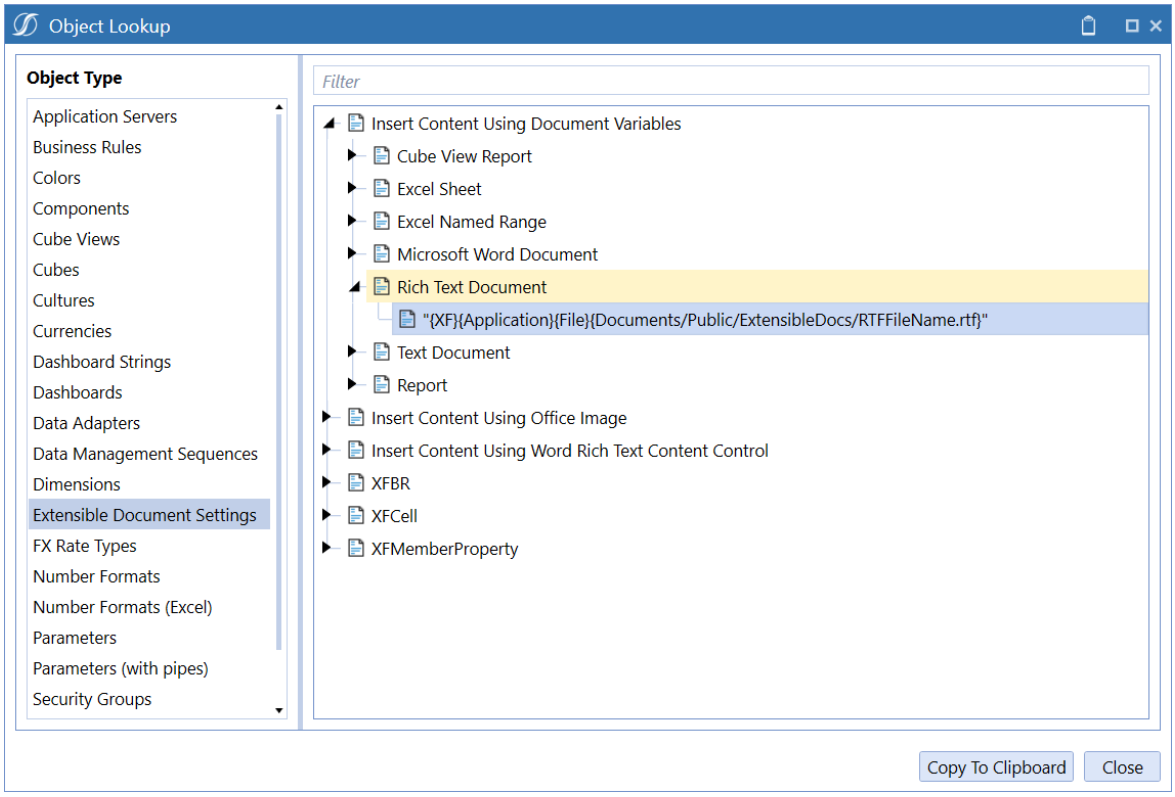
Insert a Rich Text Document in Text Editor

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert any type of content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views

- Workspaces
- Data Management

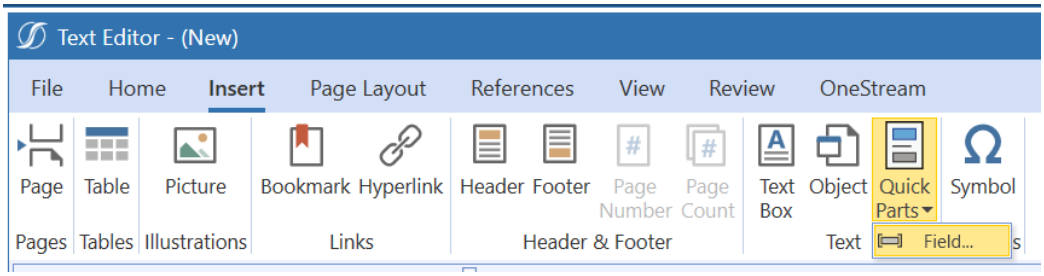


2. Once in the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Document Variables** and then expand **Rich Text Document**. Select the line of syntax and click **Copy to Clipboard** to copy the string.

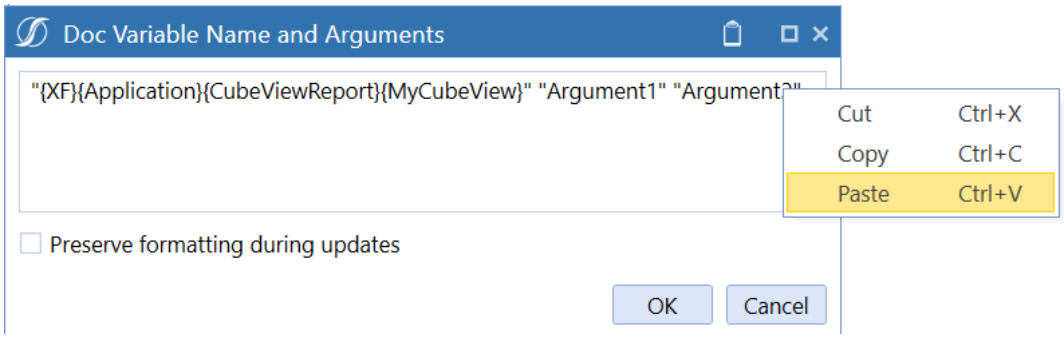


3. In the Windows Application, under Tools, click **Text Editor**. Begin with a new or existing file in Text Editor.

4. To inset your string, go to **Insert>Quick Parts> Field**.



5. A **Doc Variable Name and Arguments** dialog box appears. Highlight the placeholder text and paste the copied syntax. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/RTFFileName.rtf}`" displays in the field.



See Extensible Document Settings for more examples on these different item types.

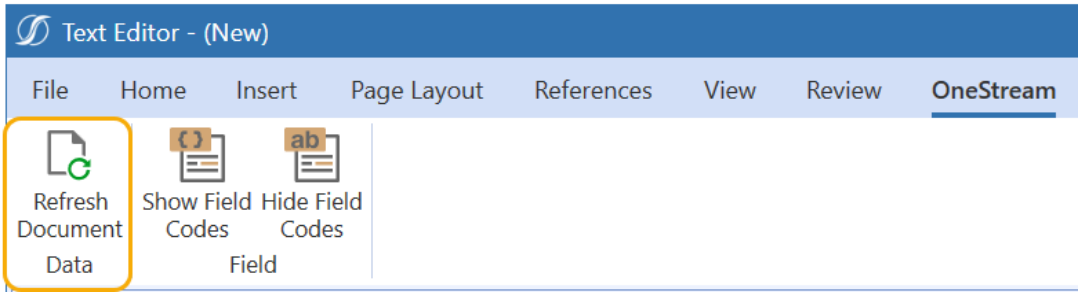
> **TIP:** Under Field Options, preserve formatting during updates is unchecked. See Preserve Formatting during Updates for more information.

6. Update the values to reflect the rich text format file path. Then in the Field dialog, click the **OK** button. The final string will display in Text Editor: `{DOCVARIABLE "{XF}{Application}{File}{Documents/Public/Extensible Documents/RichTextFormatDoc.rtf}"}`

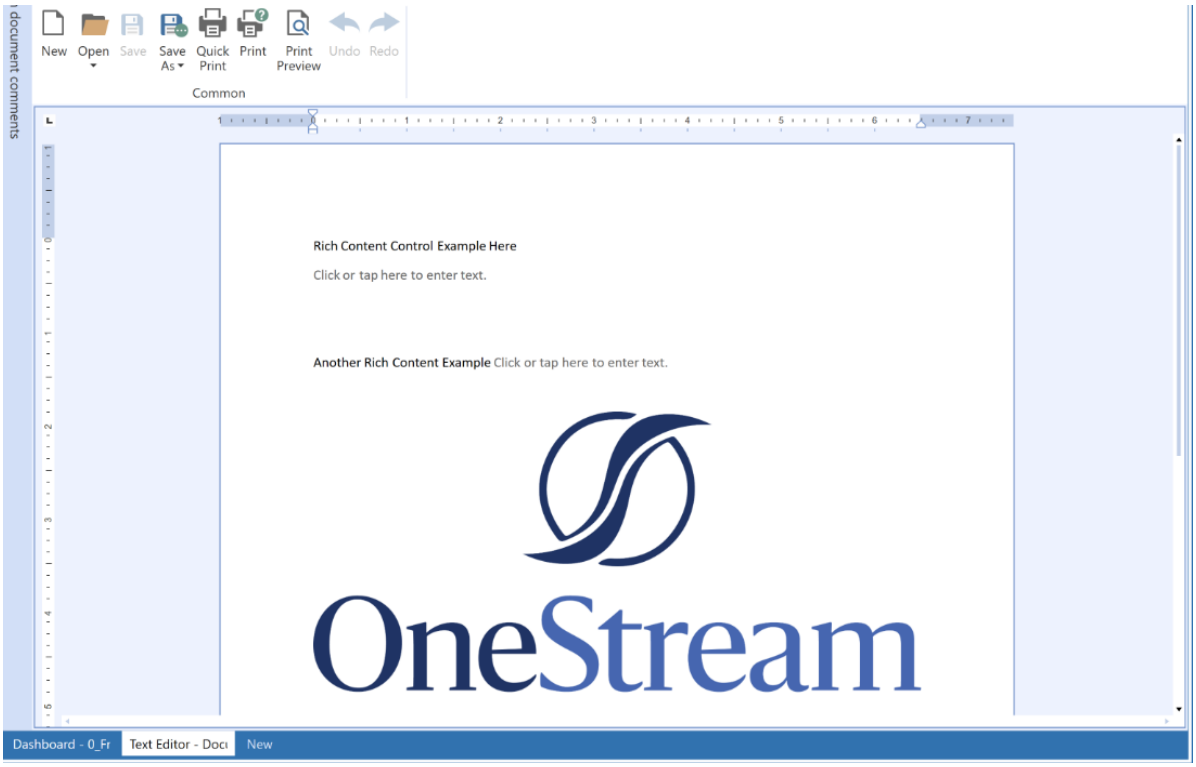
7. Place your cursor on the page and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page. You can also use the **Show Field Codes** and **Hide Field Codes** buttons under the OneStream Ribbon to show or hide field codes such as Document Variables.

> **TIP:** If pressing **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As** and select **Save As File in OneStream File System**. Select a specific folder in the File Explorer, then enter a File Name and add **.xfdoc** extension. The file name is appended with ".xfdoc.docx".
9. In the OneStream ribbon, click the **Refresh Document** toolbar button.



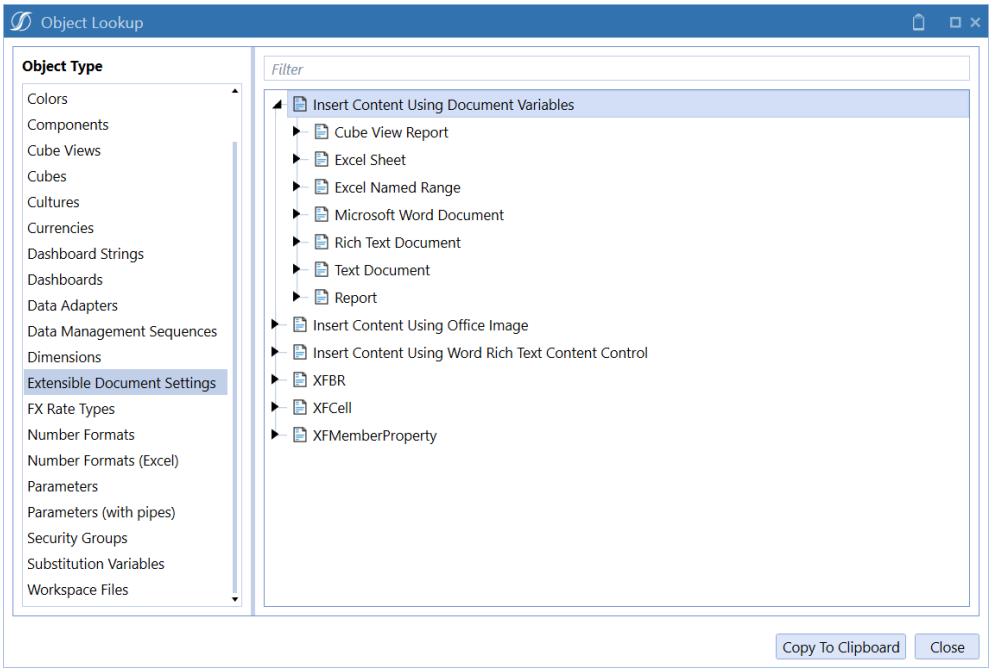
The Text Editor page refreshes and you see the embedded Rich Text Document.



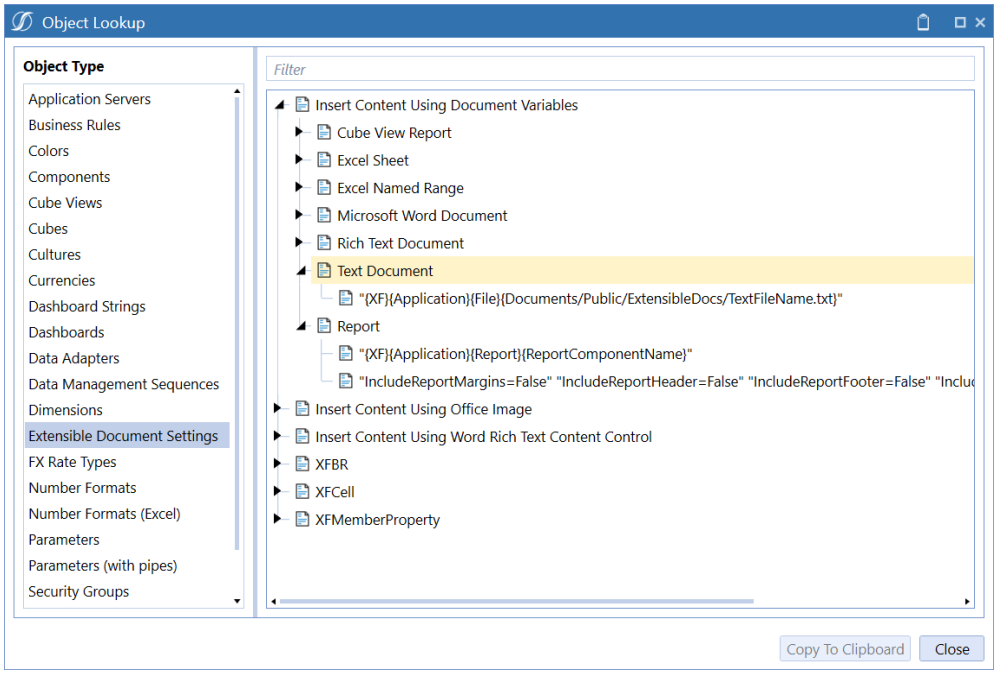
Insert a Text Document in Text Editor

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:

    - Form Templates
    - Books
    - Cube Views
    - Workspaces
    - Data Management

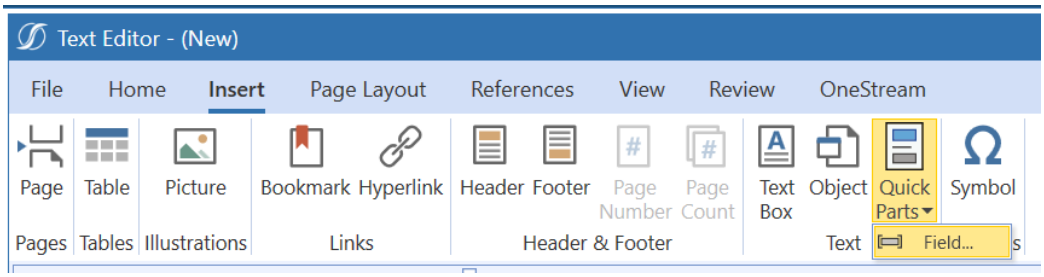


2. Once in the dialog box, select **Extensible Document Settings**. Expand **Insert Content using Document Variables** and then expand **Text Document**. Select the line of syntax and click **Copy to Clipboard** to copy the string.

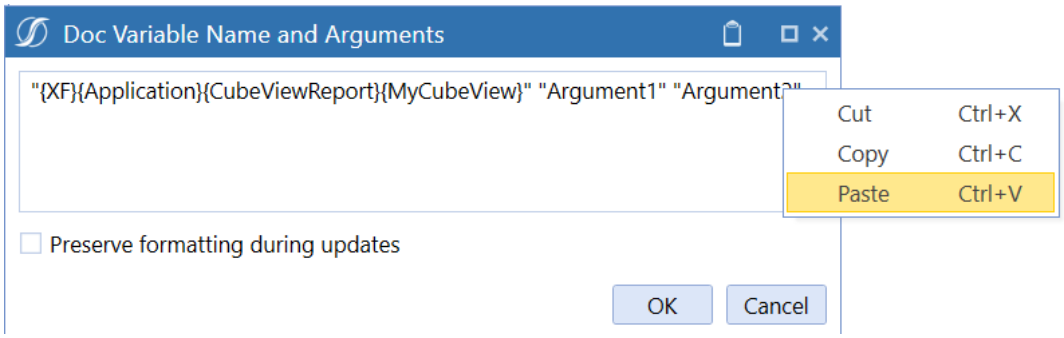


3. In the Windows Application, under Tools, click **Text Editor**. Begin with a new or existing file in Text Editor.

4. To inset your string, go to **Insert>Quick Parts> Field**.



5. A **Doc Variable Name and Arguments** dialog box appears. Highlight the placeholder text and paste the copied syntax. The string "`{XF}{Application}{File}{Documents/Public/ExtensibleDocs/TextFileName.txt}`" displays in the field.

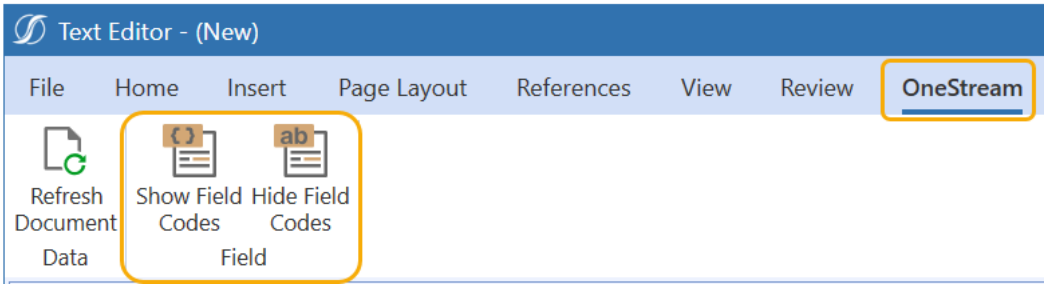


See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under Field Options, preserve formatting during updates is unchecked. See Preserve Formatting during Updates for more information.

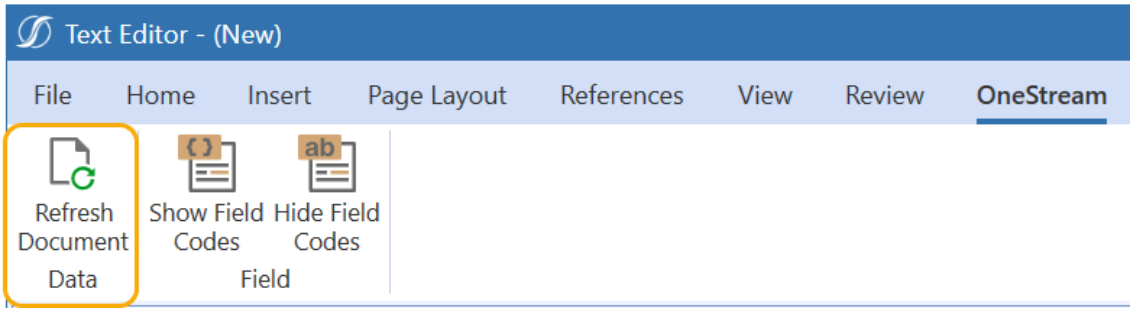
6. Update the values to reflect the folder structure and the text document path. Then in the Field dialog, click the **OK** button. The final string will display in Text Editor: `{DOCVARIABLE "{XF}{Application}{File}{Documents/Public/Extensible Documents/Testtoupdate.txt}"}`

7. Place your cursor on the page and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page. You can also use the **Show Field Codes** and **Hide Field Codes** buttons under the OneStream Ribbon to show or hide field codes such as Document Variables.

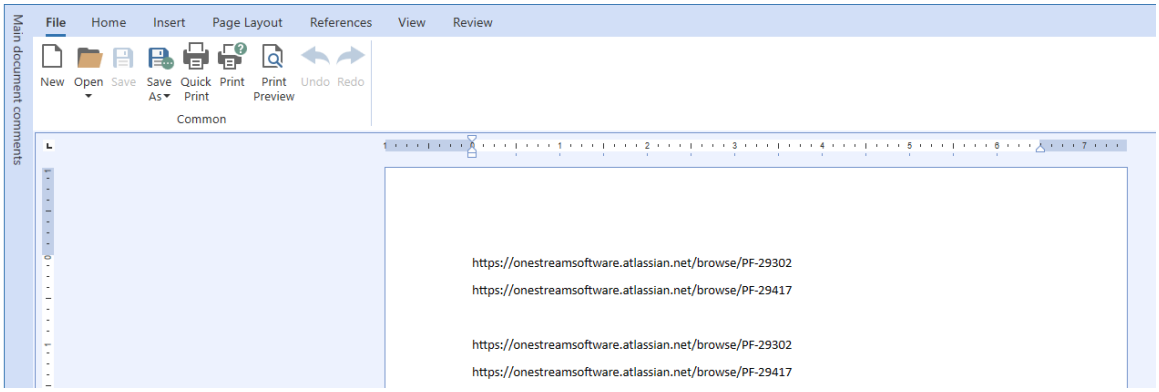


> **TIP:** If pressing **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As** and select **Save As File in OneStream File System**. Select a specific folder in the File Explorer, then enter a Name and add **.xfdoc** extension. The file name is appended with ".xfdoc.docx".
9. In the OneStream ribbon, click the **Refresh Document** toolbar button.

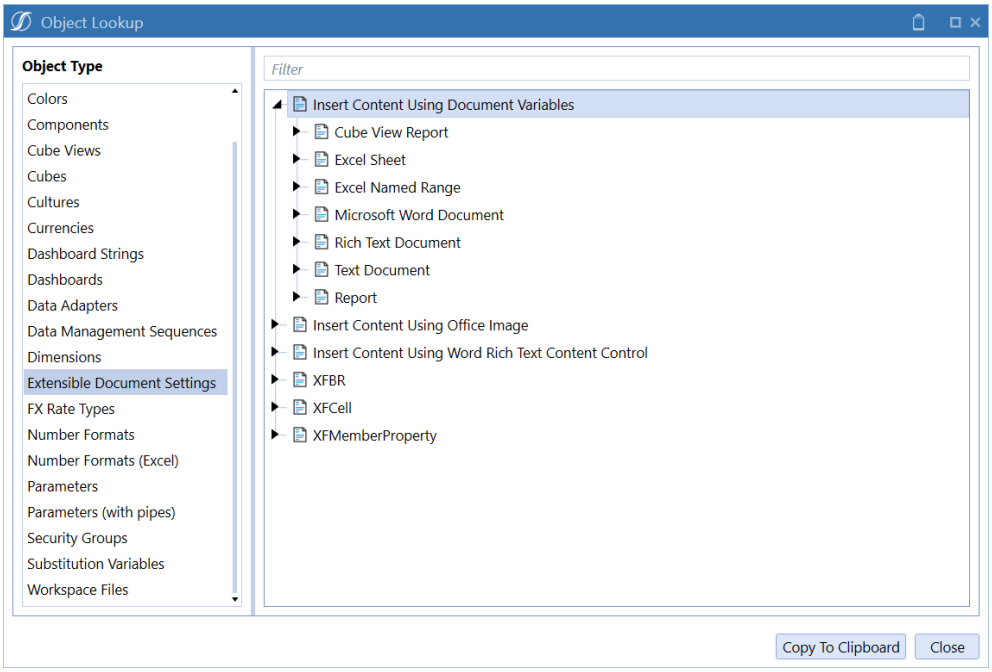


The Text Editor page refreshes and you see the embedded text document.

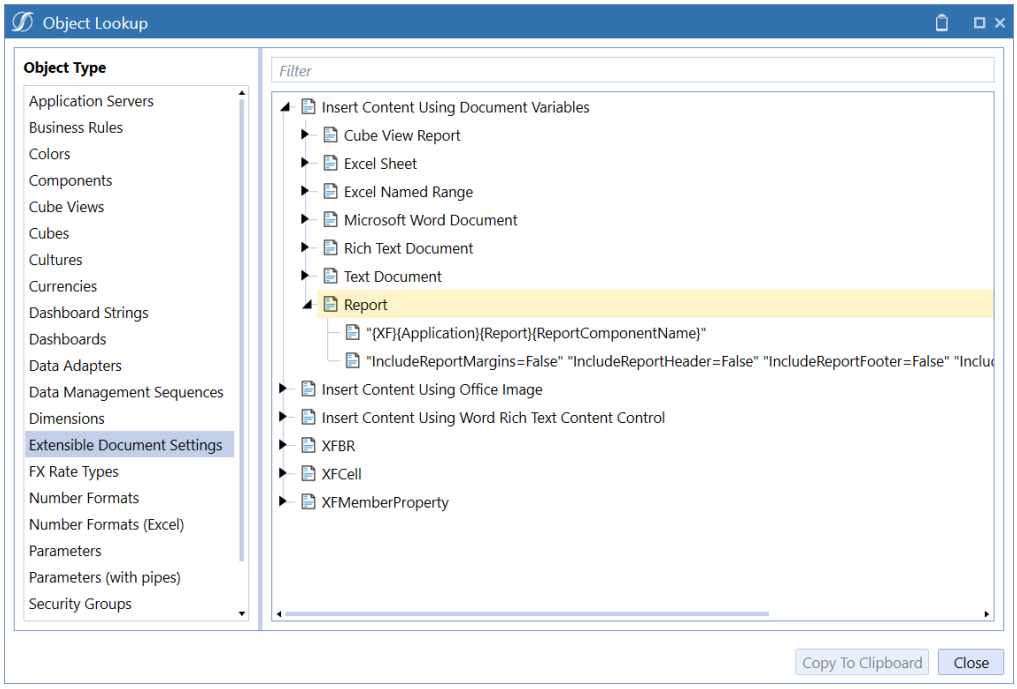


Insert a Report in Text Editor

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views
    - Workspaces
    - Data Management

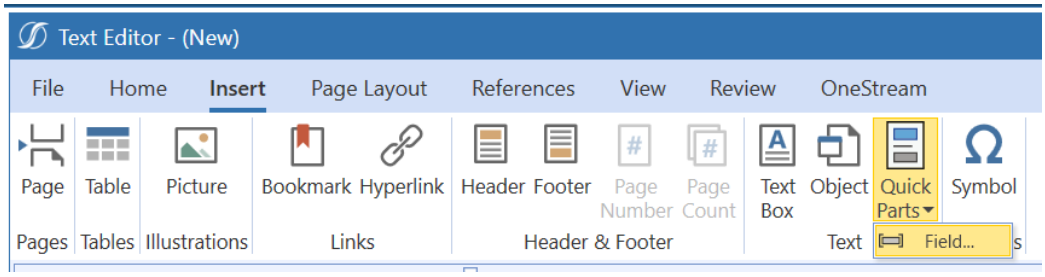


2. Once in the dialog box, select **Extensible Document Settings**. Expand **Insert Content using Document Variables** and then expand **Report**. Select the line of syntax and click **Copy to Clipboard** to copy the first string.

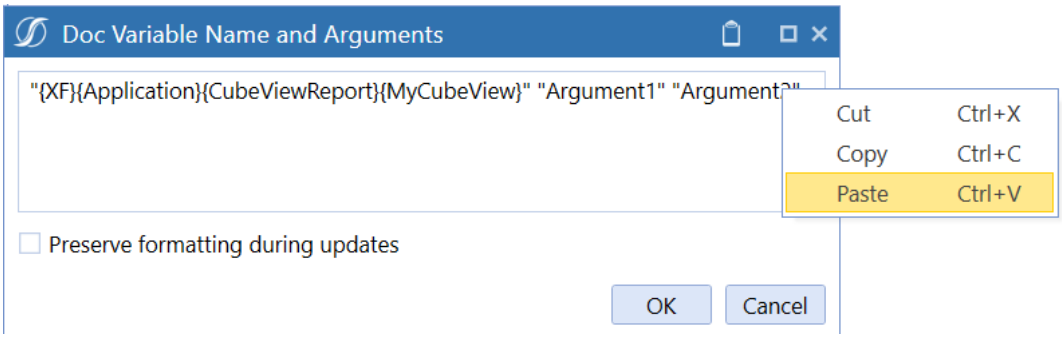


3. In the Windows Application, under Tools, click **Text Editor**. Begin with a new or existing file in Text Editor.

4. To inset your string, go to **Insert**>**Quick Parts**> **Field**.



5. A **Doc Variable Name and Arguments** dialog box appears. Highlight the placeholder text and paste the copied syntax. The string "`{XF}{Application}{Report}{ReportComponentName}`" displays in the field.

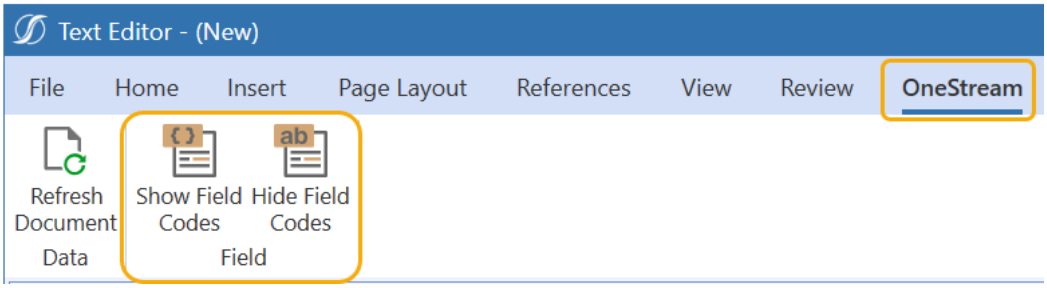


Optional: Report arguments, such as "IncludeReportMargins=False", can be pasted in "Argument1" "Argument2" sections. See Extensible Document Settings for more examples on these different item types.

> **TIP:** Under Field Options, preserve formatting during updates is unchecked. See Preserve Formatting during Updates for more information.

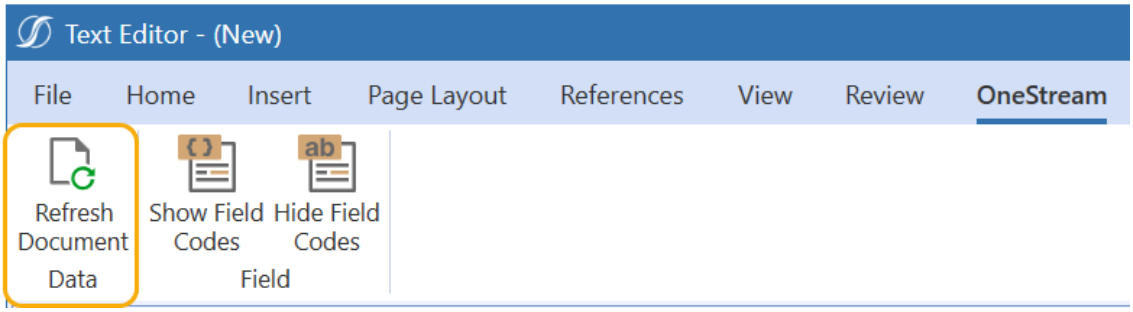
6. Update the values to reflect the report you will add. Then in the Field dialog, click the **OK** button. The final string will display in Text Editor: `{DOCVARIABLE "{XF}{Application}{Report}{Report3}"}`

7. Place your cursor on the page and press the **Alt+F9** keys. This allows for the selected value to be shown or hidden on the page. You can also use the **Show Field Codes** and **Hide Field Codes** buttons under the OneStream Ribbon to show or hide field codes such as Document Variables.



> **TIP:** If pressing **Alt+F9** keys does not work, press **Alt+Fn+F9** simultaneously instead.

8. Select **File and Save As** and select **Save As File in OneStream File System**. Select a specific folder in the File Explorer, then enter a File Name and add **.xfdoc** extension. The file name is appended with ".xfdoc.docx".
9. In the OneStream ribbon, click the **Refresh Document** toolbar button.



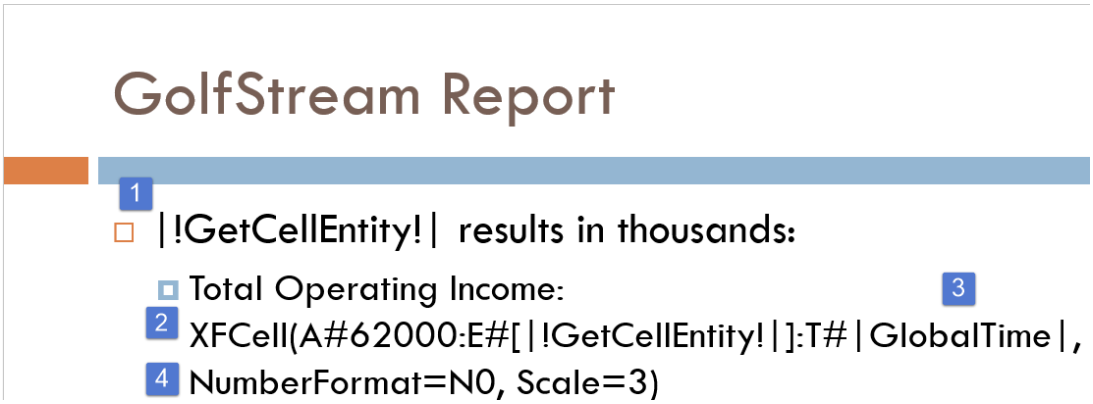
The Text Editor page refreshes and you see the embedded report.

Creating a Document in Microsoft PowerPoint

The example below is a slide from a management presentation. It shows how to complete the following:

- Use parameters
- Use substitution variables
- Use the `XFCell` retrieve function

> **NOTE:** The same format for `XFCell` is used in Microsoft Word and text files.

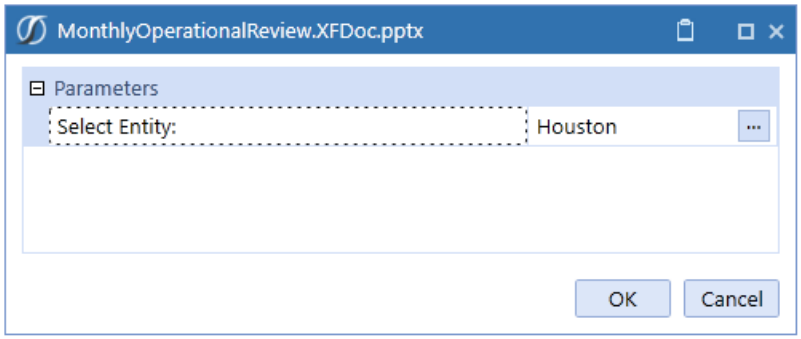


1. A parameter named |!GetCellEntity!| is created to select a specific entity on which to base the data. When the extensible document is processed at run-time, it prompts to select the desired entity.
2. `XFCell` retrieves data from a single cell in OneStream. The example above retrieves data from Account 62000 for the entity selected at run-time, for the application's global time period.

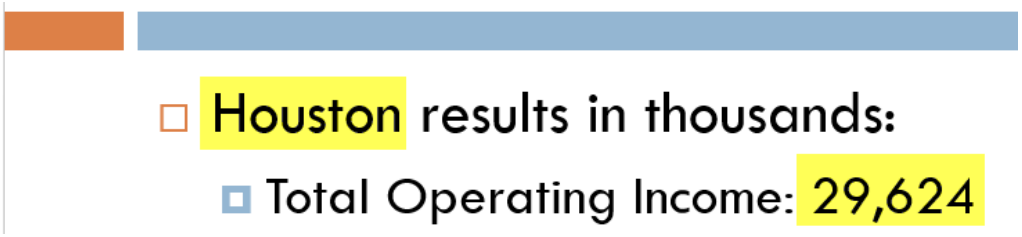
   > **NOTE:** See Extensible Document Settings for more details on `XFCell` syntax. You can also copy and paste `XFCell` syntax from a Cube View's **Cell POV Information** dialog box into a text file. See Cell POV Information in Using OnePlace Cube Views for more details on this feature.

3. A substitution variable calls the OneStream application's global time, which updates with the Global Time Period is currently set in the application.
4. Additional format settings can be included to control options such as number formatting or scaling.

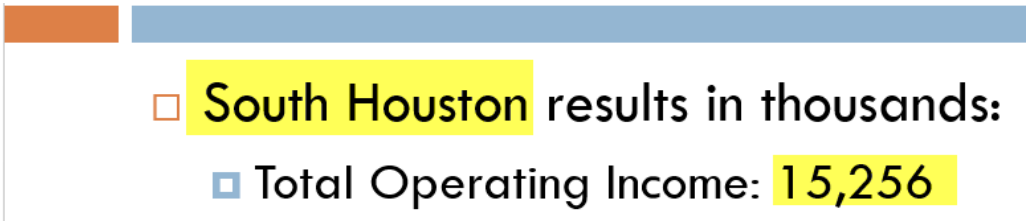
When the document is processed, it prompts to select an entity.



It then displays the updated data for the selected entity.



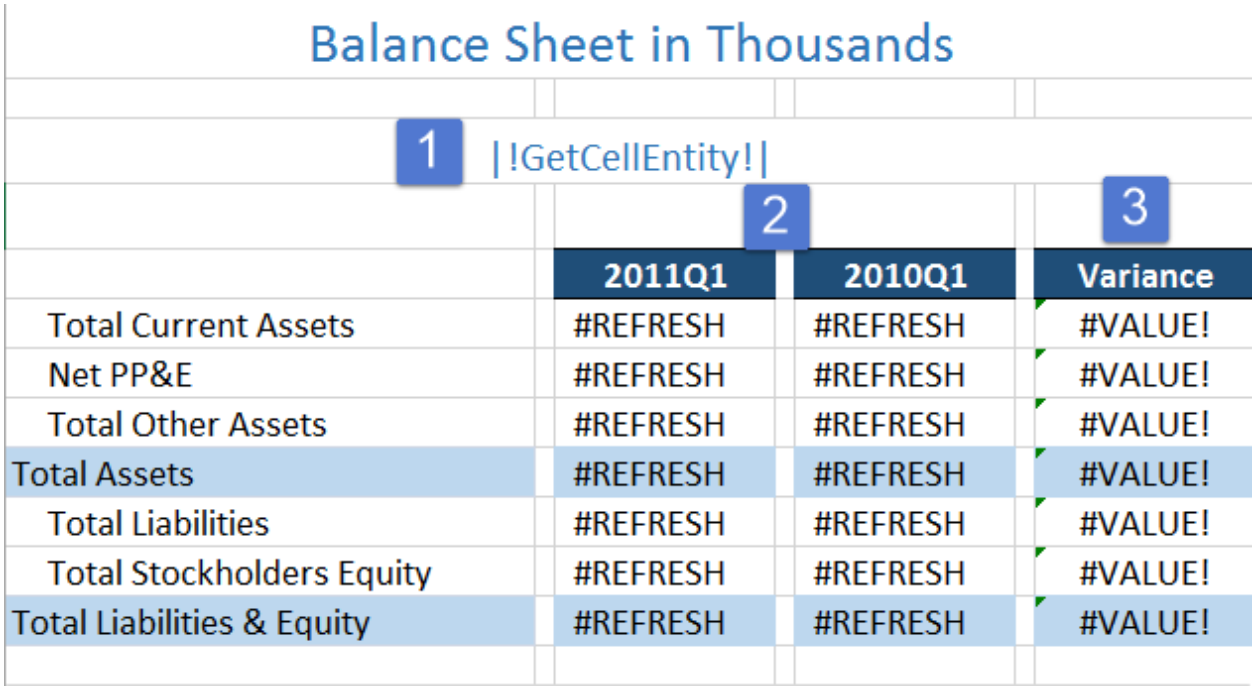
In this example, Houston has now replaced the parameter, and 29,624 is the `XFCell` function result. Data refreshes and updates when you select a different entity.



Creating a Document in Microsoft Excel

The example below is a summary balance sheet built in Excel. It shows how to:

- Use a parameter
- Use the `XFGetCellVolatile` retrieve function.
- Use the Excel `IF` function.
- Incorporate an Excel chart derived from the functions in an Excel spreadsheet.



Balance Sheet in Thousands

1 |!GetCellEntity!|

	2 2011Q1	2010Q1	3 Variance
Total Current Assets	#REFRESH	#REFRESH	#VALUE!
Net PP&E	#REFRESH	#REFRESH	#VALUE!
Total Other Assets	#REFRESH	#REFRESH	#VALUE!
Total Assets	#REFRESH	#REFRESH	#VALUE!
Total Liabilities	#REFRESH	#REFRESH	#VALUE!
Total Stockholders Equity	#REFRESH	#REFRESH	#VALUE!
Total Liabilities & Equity	#REFRESH	#REFRESH	#VALUE!

In this example:

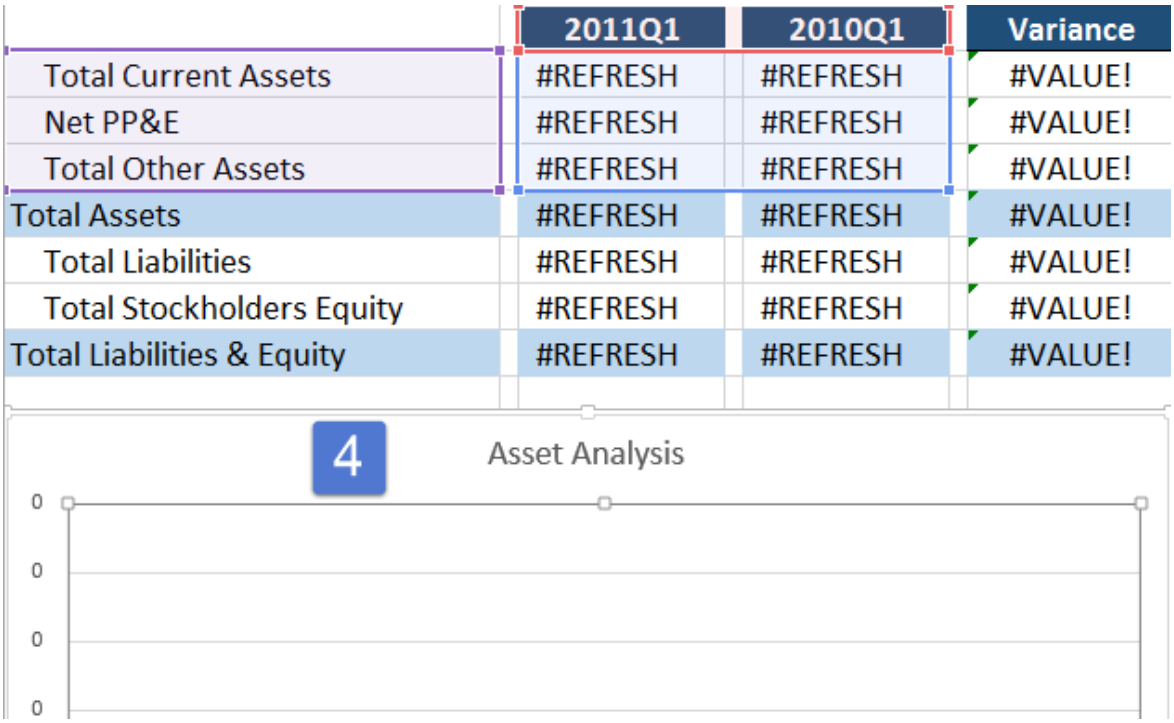
1. A parameter named |!GetCellEntity!| was created in order to allow the user to select a specific entity on which to base the data for this spreadsheet. When the extensible document processes at run-time, it prompts to select an entity.

2. The `XFGetCellVolatile` function was used to retrieve specific data from OneStream and update the Excel chart once the data is refreshed. Excel requires a volatile function for proper refreshing when using charts that reference calculated cells. This `XFGetCellVolatile` formula derives from the |!GetCellEntity!| prameter and display updated data when selecting an entity at run-time.

   > **NOTE:** See Retrieve Functions in the Excel Add-In for more details.

3. The `IF` Excel function was also used which derives from the `XFGetCellVolatile` function. This data updates when the spreadsheet launches from OneStream and the data refreshes. An example of the `IF` formula is as follows:

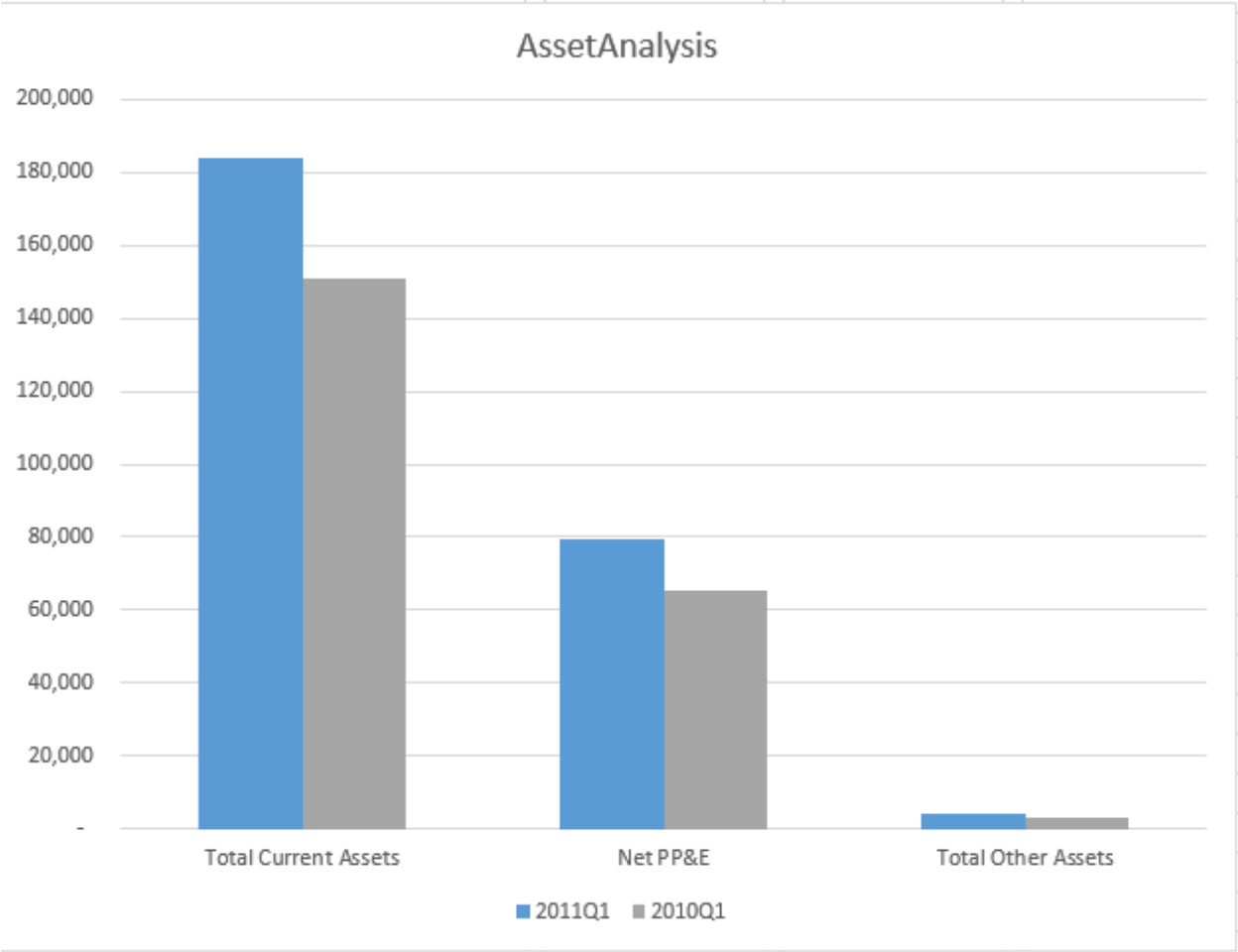
```
=IF((F6=0),"", (D6-F6)/F6)
```



4. An Excel chart inserted into this spreadsheet is driven by the data. The chart display the correct values whenever it is refreahed.

When the document processes, it prompts to select an entity and then runs the Excel spreadsheet. Log into the Excel Add-In and click **Refresh Data**to see updated values.

Balance Sheet in Thousands			
Houston			
	2011Q1	2010Q1	Variance
Total Current Assets	184,172	150,887	22.1%
Net PP&E	79,573	65,250	22.0%
Total Other Assets	3,867	3,171	22.0%
Total Assets	267,613	219,308	22.0%
Total Liabilities	196,919	207,433	(5.1%)
Total Stockholders Equity	70,694	11,876	495.3%
Total Liabilities & Equity	267,613	219,308	22.0%

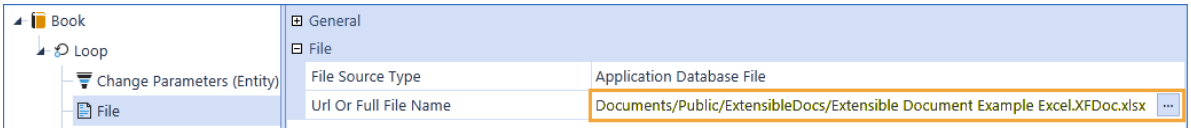


Using Extensible Documents in OneStream

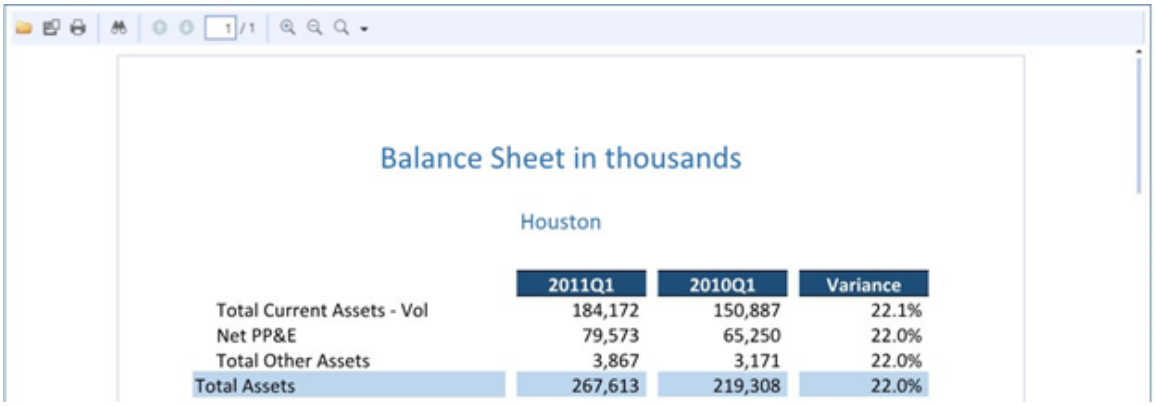
You can use extensible documents throughout OneStream. Each extensible document file is stored in the file share and can be kept private for specific users or placed into public folders for other users to see and use.

Extensible Documents in Report Books

Add extensible documents to report books by selecting a File Book item and assigning the extensible document file.



The following shows the assignment result:



You must have access to the extensible document to run this book or the file displays blank pages. See Presenting Data Using Report Books in Presentation for more details on this feature.

Extensible Documents in Dashboards

View extensible documents in dashboards by assigning them to a File Viewer dashboard component that opens and processes the extensible document at run-time. You can also view extensible documents when assigned to a book that is assigned to a Book Viewer dashboard component. See Dashboards in Presenting Data With Books, Cube Views and Dashboards for more details on dashboard components.

> **NOTE:** If a book contains an extensible Excel document using the `XFGetCell` function, you do not need to login to the Add-In to see updated values.

Extensible Documents in Data Management

Run extensible documents using a data management sequence by selecting the Export File step type and assigning the specific extensible document file to it. The extensible document exports to the OneStream file share when this step processes. See Data Management in Application Tools for more details on this feature.

Extensible Document Limitations

Extensible documents are powerful, but there are some limitations when using different file types. Extensible documents cannot reference themselves.

Text File Limitations

- Does not support images.

Microsoft Word Limitations

- Word charts cannot be converted to PDF.

Microsoft PowerPoint Limitations

- PowerPoint cannot be converted to PDF.
- When using the Rich Text box component with any Extensible Document or Dashboards, the following display format properties will be ignored in OneStream application and Modern Browser Experience:
    - FontFamily
    - FontSize
    - Bold
    - Italic
    - TextColor

- BackgroundColor
- BorderColor

- You can still process and open an Extensible PowerPoint Document that contains an Alt Text - Document Variable with MeasurementUnit, PageWidth, and PageHeight settings that are equal to < 1 inch.
- The out-of-range settings will automatically be converted to the minimum values equivalent to MeasurementUnit=Inch, PageWidth=1, and PageHeight=1. This includes:
  - When MeasurementUnit=Document and PageWidth < 300 or PageHeight < 300.
  - When MeasurementUnit=Centimeter and PageWidth < 2.54 or PageHeight < 2.54.
  - When MeasurementUnit=Inch and PageWidth < 1 or PageHeight < 1.
  - When MeasurementUnit=Millimeter and PageWidth < 25.4 or PageHeight < 25.4.
  - When MeasurementUnit=Point and PageWidth < 72 or PageHeight < 72.

> **NOTE:** You can process and open an Extensible PowerPoint Document that contains a measurement equal to an integer (for example, MeasurementUnit=123). The integer will be treated as an invalid value, such as MeasurementUnit=invalid, and be automatically converted to the default Inch setting.

Microsoft Excel Limitations

- You can only have 500 or less rows and 50 or less columns. If you include an Argument for an Excel Sheet or Excel Named Range that exceeds the maximum limit, you will receive an error message when attempting to process.

- The center across columns function is not supported.
- Excel Spark lines is not supported.
- Word wrap is not supported.
- 3D charts are not supported.
- Modifications to colors of individual chart items in a series does not carry through to a PDF.
- Fitting columns to a page is not supported.
- Repeating columns and rows is not supported.
- Cell comments do not display on a PDF after processing.
- Background images do not display after processing in Excel or PDF.
- Shapes do not display after processing in Excel or PDF.
- Smart Art does not display after processing in Excel or PDF.
- Macros are not supported.

The following Excel functions are not supported in extensible documents.

- ACCRINT
- ACCRINTM
- AGGREGATE
- AMORDEGRC
- AMORLINC
- BAHTTEXT
- BETA.DIST
- BETA.INV

- BINOM.DIST
- BINOM.DIST.RANGE
- BINOM.INV
- BINOMDIST
- CHISQ.DIST
- CHISQ.DIST.RT
- CHISQ.INV
- CHISQ.INV.RT
- CHISQ.TEST
- CONFIDENCE.T
- COUPDAYBS
- COUPDAYS
- COUPDAYSNC
- COUPNCD
- COUPNUM
- COUPPCD
- COVARIANCE.P
- COVARIANCE.S
- CRITBINOM
- CUBEKIPIMEMBER
- CUBEMEMBER

- CUBEMEMBERPROPERTY
- CUBERANKEDMEMBER
- CUBESET
- CUBESETCOUNT
- CUBEVALUE
- DDB
- DEVSQ
- DISC
- DURATION
- EXPON.DIST
- EXPONDIST
- F.DIST
- F.DIST.RT
- F.INV
- F.INV.RT
- F.TEST
- FACTDOUBLE
- FILTERXML
- FISHER
- FISHERINV
- FORECAST

- FVSCHEDULE
- GAMMA
- GAMMA.DIST
- GAMMA.INV
- GAUSS
- GCD
- GETPIVOTDATA
- GROWTH
- HARMEAN
- HYPGEOM.DIST
- HYPGEOMDIST
- INTRATE
- ISPMT
- KURT
- LCM
- LIMEST
- LOGEST
- MDURATION
- MULTINOMINAL
- NEGBINOM.DIST
- NEGBINOMDIST

- ODDFPRICE
- OFFFYIELD
- ODDLPRICE
- ODDLYIELD
- PERCENTILE.EXC
- PERCENTILE.ING
- PERCENTILERANK.EXC
- PERCENTILERANK.INC
- PERMUT
- PERMUTATIONA
- PHI
- POISSON
- POISSON.DIST
- PRICE
- PRICEDISC
- PRICEMAT
- PROB
- QUARTILE.EXC
- QUARTILE.INC
- RANK.AVG
- RANK.EQ

- RECEIVED
- ROMAN
- RSQ
- RTD
- SERIESSUM
- SKEW.P
- SLN
- STANDARDIZE
- STDEVPA
- STEYK
- SYN
- T.DIST
- T.DIST.2T
- T.DIST.RT
- T.INV
- T.INV.2T
- T.TEST
- TBILLEQ
- TBILLPRICE
- TBILLYIELD
- TREND

- TRIMMEAN
- UNICHAR
- VARA
- VARPA
- VDB
- WEBSERVICE
- WEIBULL
- WEIBULL.DIST
- XIRR
- YIELD
- YIELDDISC
- YIELDMAT
- Z.TEST
- ZTEST

Implementing Security

This section describes the four-pronged approach to managing security, which consists of:

- Workflow security
- Entity security
- Account security
- Security roles

Security can be implemented on accounts or dimensions allowing you to control who can review specific dimension members. Security is determined through users and groups, with users given specific roles to determine what data can be accessed or edited.

Application Security

A four-prong approach to application security consists of:

- Workflow security
- Entity security
- Account security
- Security roles

Once you identify entities and assign them to workflow profiles, data loaders and certifiers can be determined for each entity. Data loaders load data into the system, therefore they need read/write access to entities. Data certifiers review and sign off on the loaded data, so they need read access to entities. Security can also be done on the account or any other dimension to control who can review specific dimension members.

Security is determined through users and groups. Users are given specific roles to determine what data is accessed or edited. For example, if a user is given the ModifyData role in an application, he/she will have write-access to any data in it. Users are also put into security groups. Groups can support native groups, exclusion groups, or groups of groups. For example, a user can be put into an entity's read/write data group for read/write access to the entity's data.

Every object has access and maintenance security rights, with the exception of Task Scheduler. Access allows the security group to view the object, while maintenance allows the groups to edit the definition of the object. This system applies to most objects, such as cube views, dashboards, transformation rules, and workflow profiles.

Security Best Practices

Object Security

There needs to be different levels of access for object types and groups of objects, such as Cube Views, Dashboards or Business Rules. Application and System Roles can be granted to User Groups which create subject area administrators and by giving certain rights to a group, such as ManageCubeViews, pseudo administrators are created for these actions. This provides the most power for a specific object type.

A Maintenance Group is the middle level of power for an object at the group level. For example, a Maintenance Group assigned to a specific Entity Transformation Rule Group allows the assigned users to create, edit, and delete rules within that Transformation Rule Group.

An Access Group is the lowest level of power for an object at the group level. This means the object can be used, but its definition cannot be edited.

Confirmation Rules

Confirmation Rule Groups are assigned to Confirmation Rule Profiles which are then assigned to Workflow Profiles. The run time access to these Confirmation Rules depends on to which Workflow Profile they have been assigned. If a user has Workflow Execution Access, he/she will be able to execute them.

The best way to control Confirmation Rules is to set Access to Everyone and Maintenance to Administrators for both Confirmation Rule Groups and Profiles.

Certification Questions

Certification Question Groups are assigned to Certification Question Profiles which are then assigned to Workflow Profiles. The run time access to these Certification Questions depends on to which Workflow Profile they have been assigned. If users have Workflow Execution Access, they will be able to execute them.

The best way to control Certification Questions is to set Access to Everyone and Maintenance to Administrators for both Certification Question Groups and Profiles.

Data Sources

Data Sources are assigned to Workflow Profiles. The run time access to these Data Sources depends on to which Workflow Profile they have been assigned. If a user has Workflow Execution Access, he/she will be able to execute them.

The best way to control Data Sources is to have to the ManageDataSources Application role, and no security settings at the object level.

Transformation Rules

Transformation Rule Groups are assigned to Transformation Rule Profiles which are then assigned to Workflow Profiles. In this case, an appropriate user group needs to be assigned to Access and Maintenance because users will be able to right-click on an Import Workflow Profile and view/edit their Transformation Rules. The user groups should include the users assigned to execute the Workflow Profiles to which the Transformation Rule Profile has been assigned.

The best way to control Transformation Rules is to set Access to Everyone and Maintenance to Administrators for most core, shared, or corporate Transformation Rule Groups For some specific Transformation Rule Groups, such as an Account Transformation Rule Group that applies to a specific location, assign the appropriate user groups to Access and Maintenance. Block access to the Maintenance screen for anyone except administrators because this could potentially allow users more access than they need.

Form and Journal Templates

Form or Journal Groups are assigned to Form/Journal Profiles which are then assigned to Workflow Profiles. The run time access to these Forms or Journals depends on to which Workflow Profile they have been assigned. If a user has Workflow Execution Access, he/she will be able to execute them.

The best way to control Form and Journal Templates is to set Access to Everyone and Maintenance to Administrators for both Form/Journal Groups and Profiles.

Cube Views

The best way to control Cube View Groups is to set Access to Everyone and Maintenance to Administrators and anyone else building a Cube View. To keep the assignment of Cube View Groups to multiple Cube View Profiles flexible, the Cube View Groups need to remain smaller in size. For Cube View Profiles, set Access to anyone who will need to see these Cube Views in OnePlace, Excel, or assign them to Workflow Profiles, Forms, or Dashboards. Set Maintenance to anyone who needs to change the assignment of the Cube View Groups to Cube View Profiles.

OneStream recommends setting the Can Modify Data, Can Calculate, Can Translate, and Can Consolidate properties to False. This can be pre-set for all new Cube Views by creating an example or Cube View Template which can be copied to create new ones. Some examples of when this will not be needed is if the Cube Views are going to be read by administrators only, the Cube Views will be used as a data entry form or are only going to be visible in a formatted report or chart.

System and Application Dashboards

When assigning Dashboard Groups to Profiles, the Visibility is extremely important. For example, if a user has access to a Dashboard Profile in OnePlace, but not to a certain Dashboard Group in that Profile, the user will not be able see the Dashboards in that group. Also, if a user has access to the Dashboard Groups, but not the Profiles, he/she will not be able to see the Dashboards in OnePlace. If a Cube View is assigned to an Application Dashboard, and the user only has access to the Cube View, he/she will not see the Dashboard. Lastly, if a Dashboard is pointing to an Entity, Scenario, or Cube Data to which the user does not have access, he/she will see one of the following: NoAccess in Data Explorer for the cells the user cannot see, a blank cell in the Data Explorer Report, or No Data Series if he/she is viewing a chart.

The best way to control Dashboard Groups is to set Access to Everyone and Maintenance to Administrators and anyone else building a Dashboard. In order to keep the assignment of Dashboard Groups to multiple Dashboard Profiles flexible, the Dashboard Groups need to remain smaller in size. When assigning Maintenance for Dashboard Profiles, give access to anyone who needs to see the Dashboard in OnePlace, assign it to a Workflow Profile, or change the assignments of Dashboard Groups to Dashboard Profiles.

Use multiple Dashboard Maintenance Units in order to keep them a reasonable size making it easier to manage multiple objects and access. Dashboard Parameters can also be used across all Dashboards and do not need to be copied across all Maintenance Units. Security has no bearing on the use of Parameters.

Workflow Security

Security groups for Workflow Execution, which is the ability to process a Workflow for a specific Workflow Profile, Certification Signoff and the separate ability to Process, Approve and Post Journals, exist for all Workflow Profiles. In certain cases, the user simply needs the Access and Workflow Execution Group Access to run Workflow. For example, the user does not need Access or Maintenance Group access to Data Sources or Transformation Rules in order to run through the Import Workflow.

In some cases, having access to certain objects is necessary along with Workflow Execution Group Membership. The Manage App Role has to do with creating, reading, updating, and deleting Journal and Form Templates (metadata) themselves, not just instances of these objects at run time. It is expected that 90% of Workflow users will not have any of the Application Roles, but their access will be controlled by the Access Group for those Journal/Form Template Groups and Profiles. Workflow users also need Workflow Execution Group access in order to perform import, forms, and journal actions. The user does not have to be in the Manage Application role to create a Journal or enter data in the Form. Workflow security governs access to the forms. If the user is in the ManageJournalTemplates Application Role group, he/she can create any Journal needed for the Workflow Profiles to which they have proper execution access.

Users need to have at least Access Group privileges to the Cube Root Workflow Profile node to edit Workflow Profiles with having the ManageWorkflowProfiles role. Otherwise they will not be able to see any Workflow Profiles under the Cube Root Workflow Profile.

The order to follow when assigning access to Workflow Profiles and data is to first assign Read and Read/Write Groups to the Entities involved. Next, create an Access Group, Data Group, and Approver Group for each Workflow Profile and include the appropriate Entity groups.

Import

First, determine whether the users can load data to the Workflow for the assigned Entities and then determine whether they load both GL (BS and PL) and Supplemental data, or one or the other. Next, decide if the users for the assigned Entities can certify the loaded data as part of the Workflow.

Forms

First, determine whether the users can manually input data into a form and certify it as part of the Workflow for the assigned Entities.

Adjustments

First, determine whether the users can manually input data into a journal and certify it as part of the Workflow for the assigned Entities.

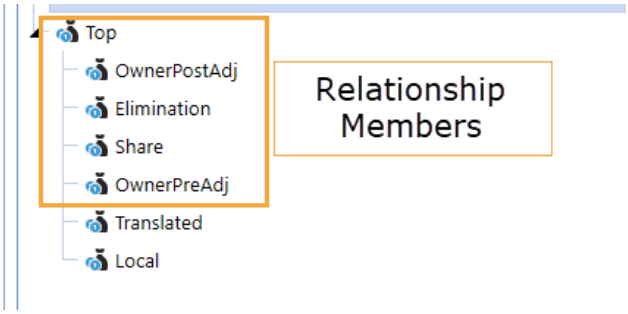
Entity Security

Entity Security controls the overall read/write access to Entity data and controls whether Cube Security should be used. When creating Entity security groups for the Read Data Group and the Read/Write Data Group, the groups should be named in a logical convention such as XXXX_View or XXXX_Mod. The Entity Read/Write Data Group should be designed first because it is needed for data loading in Workflows. The Workflow Execution Security Group should be assigned to all the Entities' Read/Write Security Group for the Workflow to gain loading access to the Entities.

When setting up View Security Groups for Entities, first consider how users need to view their data whether it is by segment or region. Determine whether it makes more sense to have one Entity View Group per Entity, or to create one Entity View Group per segment and apply one Entity View Security Group to many Entities' Read Data Group. All the Entities' View Groups below the Parent must be assigned to the Parent Level Entity View Group in order to gain access to data at the Parent Level Entity and View Entities below it. Try to minimize the amount of View Entity Security Groups where possible.

Relationship Security

You can change the security model to allow who controls viewing or modifying the relationship members in the consolidation dimension.



For the Use Parent for Relationship Consolidation dimension Members functionality:

- If set to False, the user's entity rights control their rights to all members of the consolidation dimension. This is the default security model.
- If set to True, the user's rights to the relationship members of the consolidation dimension are determined by their rights to the current entity's immediate parent.

Users have either read or read/write access to view or modify their entity so they can see their entire entity from all of the relationship members.

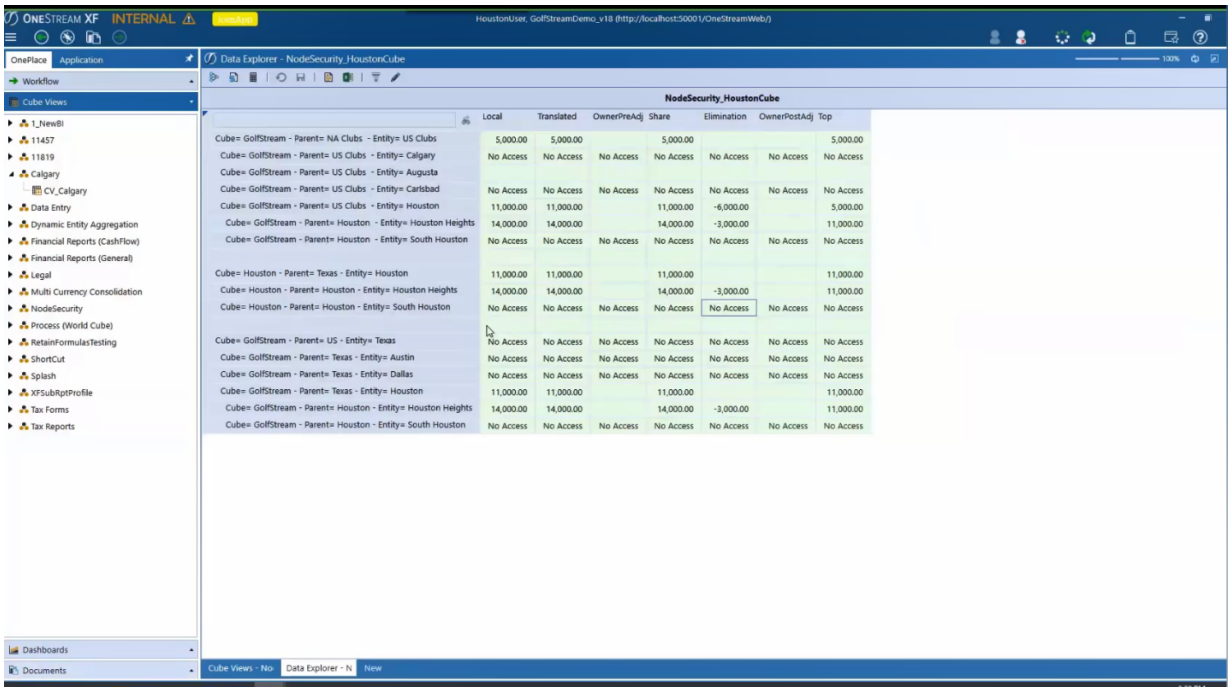
You can allow the relationship security portion of the consolidation dimension to be controlled by the access to the parent entity in the immediate hierarchy.

> **NOTE:** This is strictly a parent and child relationship.

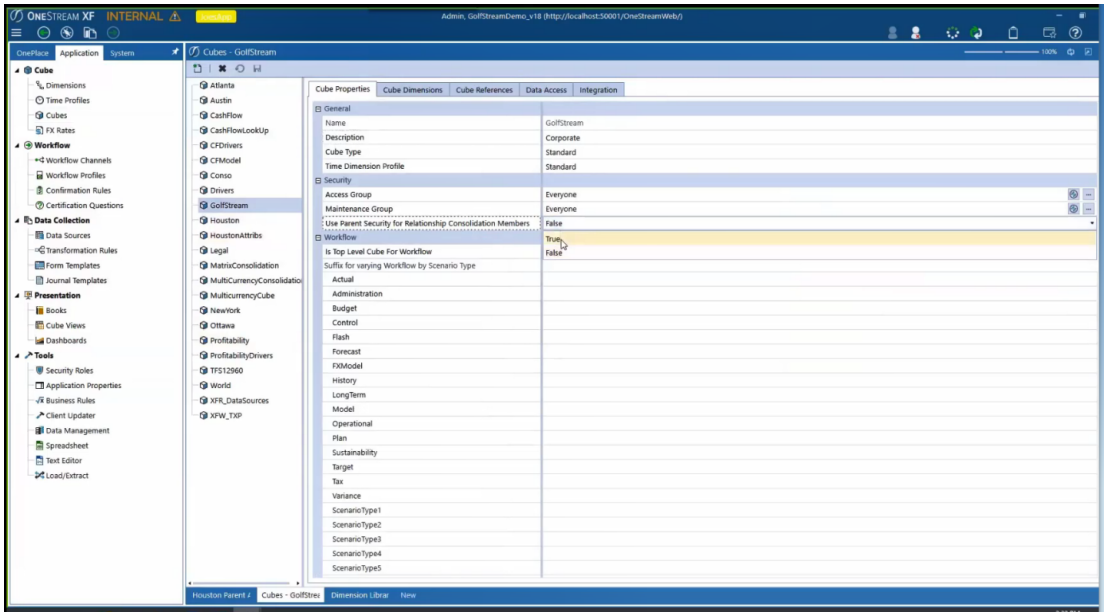
Set this feature to True when:

- A user's rights to the non-relationship consolidation dimension members (local and translated) are determined by the user's rights to the entity itself.
- A user's rights to the relationship consolidation dimension members is determined by the user's rights to the immediate parent entity.

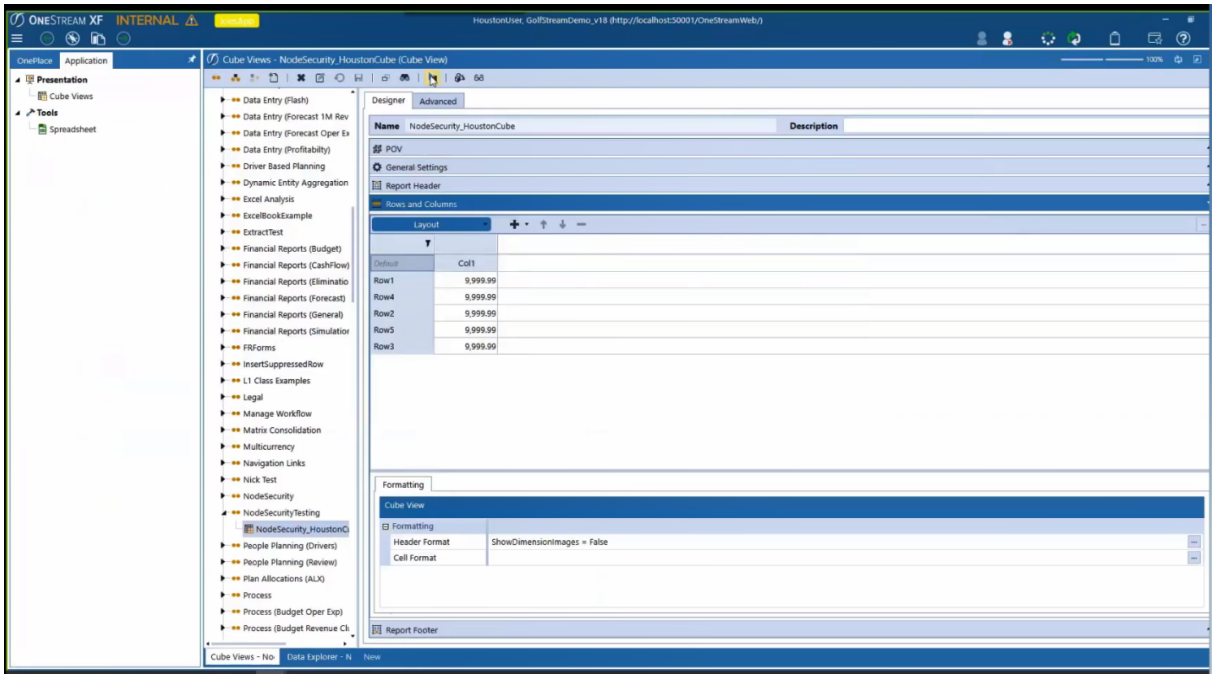
Change the Relationship Security



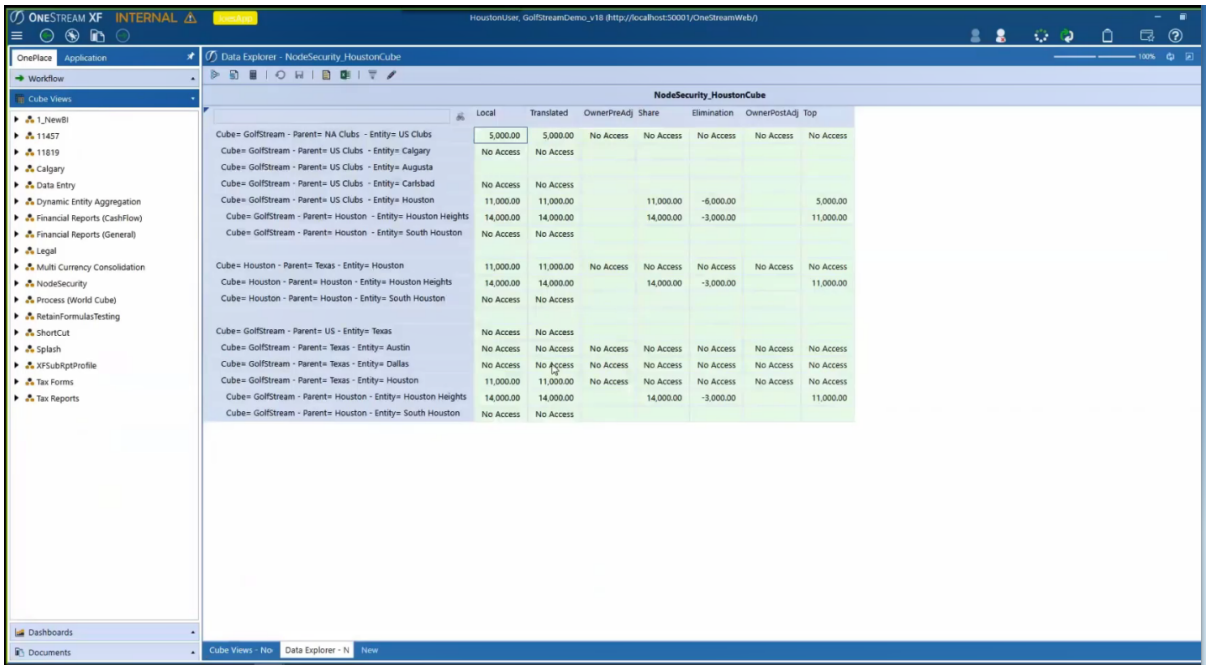
1. From the Application tab, under Cube, click **Cubes**.
2. Select a Cube.
3. In the Cube Properties tab, in **Use Parent Security for Relationship Consolidation Dimension Members**, select **True**.



4. Click **Save** to save the cube properties. You have now changed the security model for the read or write relationship for the consolidation dimension members to be based on the security rights to the immediate parent entity.
5. Click **Refresh** to refresh the application.
6. In Cube Views, go to the Designer tab and click **Open Data Explorer**.



All of the security is changed.



Notice the following changes to security:

- In the first screenshot, all the consolidation dimension members are available across Houston.
- When you switch on the new security, in the previous screenshot, the user has rights to `USClubs` then looking at the Houston member there, all the consolidation dimension members are available.
- Looking at the last group, the user has no rights to Texas, based on the NoAccess to Local and Translated. In Houston you can see the relationship members changed to NoAccess.
- The reason why USClubs turns to NoAccess is because the user apparently has no rights to the parent NAClubs, which is not displayed on the report.

Data Security

Data security controls the overall read/write access to analytic models. There are several steps to see if you have access to data. First, it that you have the ModifyData role in the application, otherwise you do not have write access to data. Next, ensures you have either the access group or maintenance group for a cube, or you do not have access to any of the cube's data. It will then check the scenario and Entity's Read Data Group and Read Write Data Group to ensure access. Finally checks the Data Cell Access Security at the cube level. If you have access to the data and use Excel or any other method to import data, it ensures the entity being written to belongs to a workflow profile with an active forms channel that is not locked or certified. If any of these steps result in no access, the process stops.

- There are several ways to guarantee data is secure. Throughout the dimensions, different security groups are available, and an administrator can decide what users belong to each group.
- The scenario dimension has both Read Data Group and Read and Write Data Group.

- The entity dimension uses both security groups from the scenario and adds display member group and Use Cube Data Access.
- The display member group only refers to the member display access level, not the data access level. Use Cube Data Access is used if slice filters are being applied at the cube level to apply additional layers of security for specified security groups.
- The account, flow, and all user-defined dimensions also use display member group.

At the cube level, more security is put in place through groups, member filters, and complex layered security. A cube slice filters a data entry form to the right member set, such as choosing the cost center in which a user can enter data. For slice filters to take effect, the user must first be granted access to the cube, scenario, and entity. Slice security cannot increase access to data that was not administered first through Users and Groups. An administrator can also lock down more than one dimension by using cube data access and can control user visibility by only giving access to certain accounts. Finally, data cell conditional input and data management access security can be used. These are not security settings but can still control how a dimension can be used for input and how a cube is modified.

Application and System Security

dimensions, cubes, business rules, data management, and File Explorer each have an access and maintenance group. OneStream recommends limiting access to these groups and assigning the screens to administrators. It is also recommended to give the ManageData role to administrators for data management and to give ManageTaskScheduler and TaskScheduler roles in Task Scheduler. The ManageFileShare role should also just be given to administrators. Anyone granted access to a folder in File Explorer has access to every file and folder through OnePlace. For FX rates, limit access by giving administrators the ManageFXRates role and selecting a set of power users.

Security Configurations

Restrict Users to an Application

Setting up access to an application can be done within the application security roles. When an application is first created, the OpenApplication defaults to `Everyone`. If specific people have access to log into OneStream, but only need access to specific applications, then security groups can be assigned to the OpenApplication role. Once the security group is created, users can be assigned to it. This can be done by performing the following steps:

1. Go to **System** > **Administration** > **Security**.
2. Create a new security group.
3. Assign all users who should have access to the application to the new security group.
4. Refresh the application in order for this new security group to appear in all drop down menus.
5. Go to **Application**> **Tools** > **Security Roles** > **OpenApplication**.

6. Click the drop-down and select the new security group.
7. Click **Save**.

Restrict Data Input by Origin

There may be times where data should be loaded through the Import origin but should not be loaded using the Forms origin. This can be handled in the data cell conditional input. For example, users may be able to load trial balance data through the Import origin, but other users submit statistical data through the Forms origin. The data cell conditional input ensures the statistical data does not overwrite the Trial Balance data in Actual. Perform the following steps to do this:

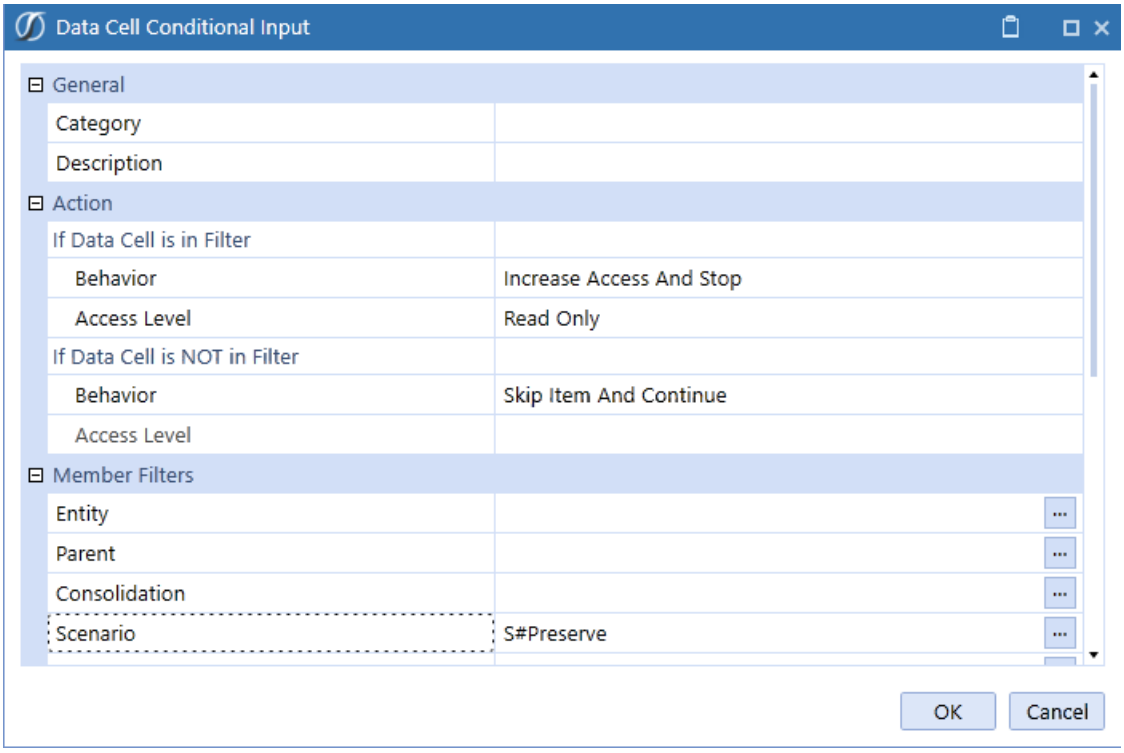
1. Go to **Application** > **Cube** > **Cubes** > **Data Access**.
2. Go to **Data Cell Conditional Input**.
3. Click ✚ to create a new line.
4. Click , or double click on the cell to make changes to the member filter. Add the dimension intersection to restrict data loading. In this case it restricts users from loading to the Trial Balance account through the Forms origin.
5. In the In Filter field, choose a behavior and choose the Read Only Access level.

Omitting Data Cell Conditional Input by Scenario

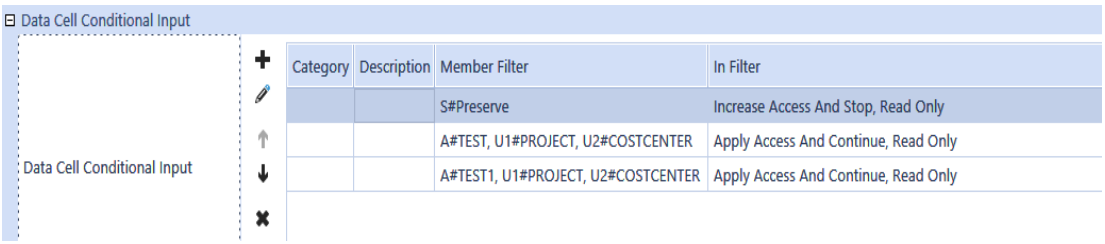
Data cell conditional input restricts access to certain intersections or slices of the cube. This behavior might not be desired for all scenarios, time periods or other elements, so this is a simple way to omit these rules for specific elements. The following method is useful when there are many data cell conditional input rules not defined by a scenario at the time of creation. After these rules are created, however, a scenario becomes a factor for historical data.

For example, there may be many read-only intersections in Actual to which users should not load data. These rules are setup in the data cell conditional input section within Application| Cube|Cube|Data Access. However, for historical purposes, there may be data in these intersections that was there prior to the filters being applied. The scenario to which the data is being copied needs to allow access because the historical data in these intersections may need to be copied for analysis.

Create a new data cell conditional input rule for the entire scenario, set the behavior to Increase Access And Stop, set the Access Level to Read Only.



Position this new rule at the top of the Data Cell Conditional Input for the scenario.



The behavior option Increases Access And Stop is being used because if the current cell matches the filter, access is being increased and all subsequent data access rules are being ignored below. In this case, the Preserve Scenario has access to everything, and the subsequent Data Cell Conditional access rules are ignored or not applied for Preserve.

System Security

System Security applies to the framework. There are some key assumptions around how the different roles or security groups work. First, it is important to understand the hierarchy of certain System Security Roles and Security Groups:

System Security Role
The System Role, such as ManageSystemDashboards, means the user has a higher privilege and does not need to be in any Maintenance Group or Access Group to see, edit, or delete all objects of that type. Having the ManageSystemDashboards System Security Role means the user can create, edit and delete any System Dashboard, System Dashboard Group, or Profile.

Maintenance Group
The Maintenance Group means users cannot only see an object, but can create new objects in Groups, edit, and delete them. Users do not need to be in the Access Group for an object if they are in the Maintenance Group. The Maintenance Group can also control the contents of Profiles.

Access Group
The Access Group means users can see the object and read its contents.

Managing a OneStream Environment

Management of all changes to the system are recommended to follow best practice procedures. Whether the changes derive from a OneStream software upgrade, or through regular application maintenance, all changes are recommended to be first deployed and tested in a development environment. There are additional benefits of making a recent copy of the production Application database, renaming it and using this as a base for these changes. Search for “Rename Application” in the Installation and Configuration Guide.

Prior to being deployed to a production environment, it is recommended to extract changes from the development environment and assess this deployment of changes in a separate test environment.

Deploying changes to a production environment should avoid times during high load and high application activity. Changes to these types of application artifacts especially should not be performed against a production environment experiencing heavy activity:

- Business Rules, whether they contain Global functions or not
- Confirmation Rules
- Metadata, especially when using member formulas

Applying changes like this while the production system is under a high level of activity may have a negative impact on servers and have the potential to cause running processes to produce an error.

Standard environments are recommended to schedule production changes during slow periods or non-work hours. Large environments should consider using the Pause functionality within the Environment tab to allow activity to wind down. These large environment managers should also consider theOneStream Solution, Process Blocker, which allows a pause of critical processes to perform maintenance on the system, without having to shut down the entire application. Process Blocker allows current tasks to be completed, while any new requests are queued, allowing the changes to be applied safely and effectively. Once these changes are in place, it is recommended to significantly limit the ability for users to make such changes during high volume.

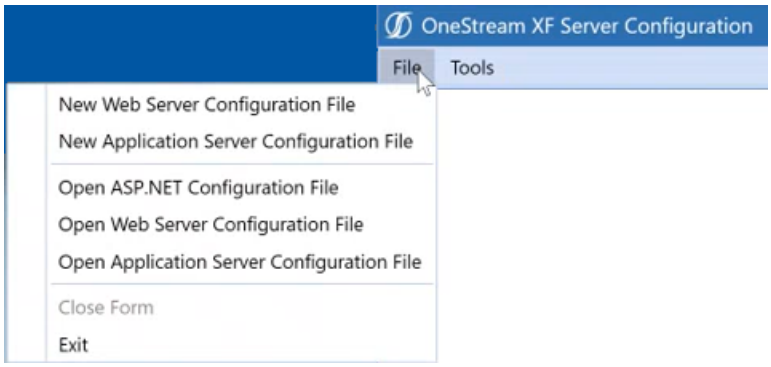
It is key that servers get a chance to recycle for good system memory health. IIS also has an Idle Time-Out setting for our OneStreamAppAppPool. This setting should be set to 0 since OneStream has other settings to recycle IIS. For active, global environments with Data Management Sequences regularly being executed, a recycle of IIS is recommended every 24 hours for these OneStream App Servers. Please discuss this situation with OneStream Support to find what is recommended, as each situation may vary.

Disable Inactive Users

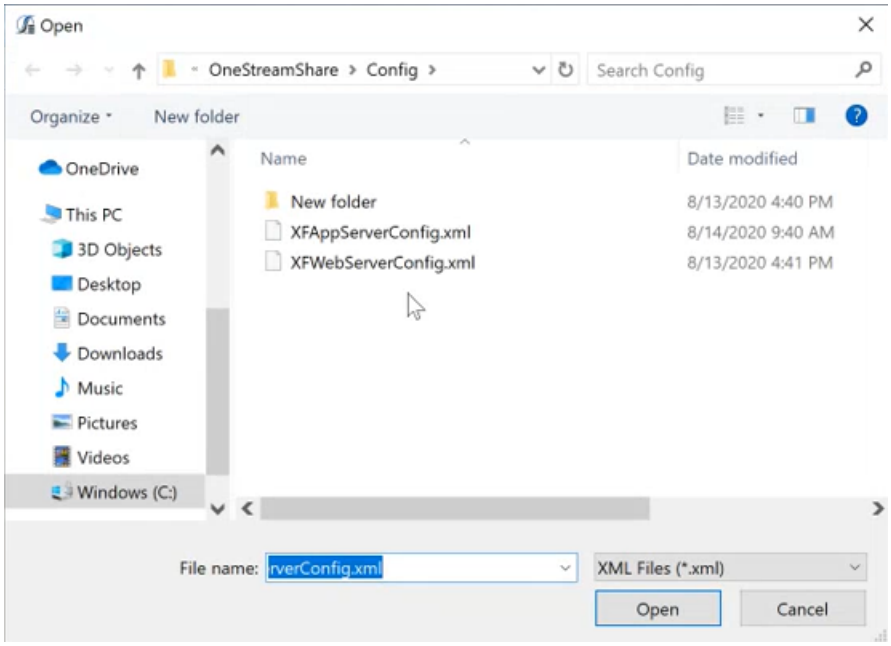
Disable inactive users gives you the ability to create an authorization policy to disable users after a specific amount of logon inactivity, keeping only active users in the system.

Set the Inactivity Threshold

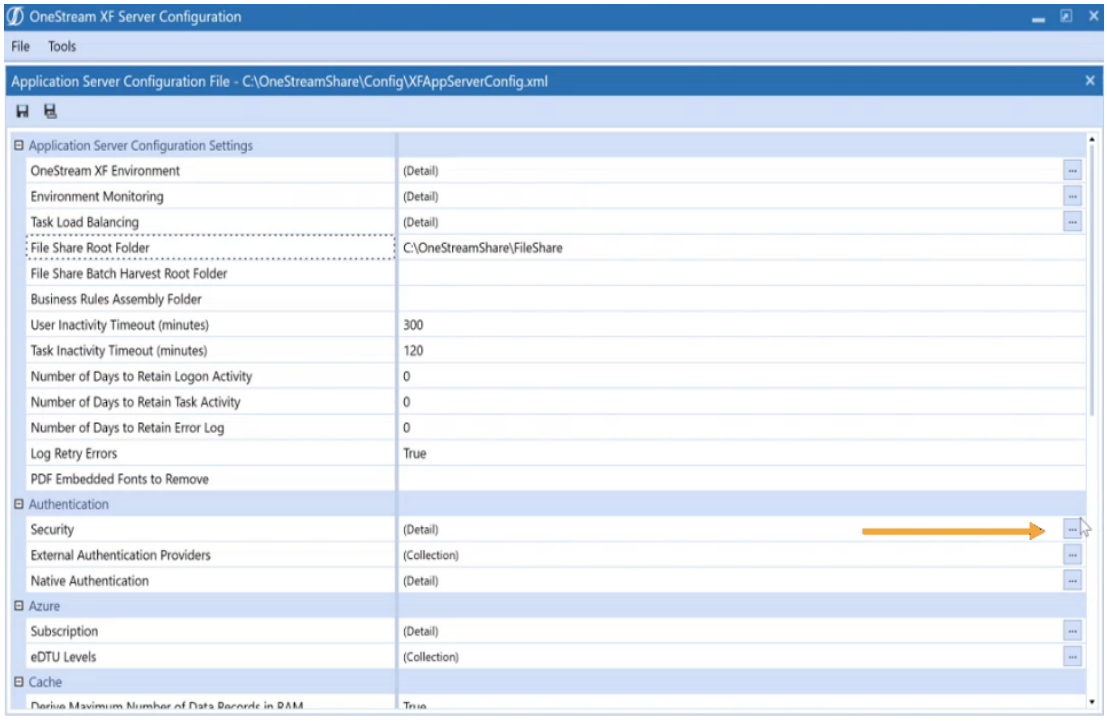
1. Click **File** > **Open Application Server Configuration File**.



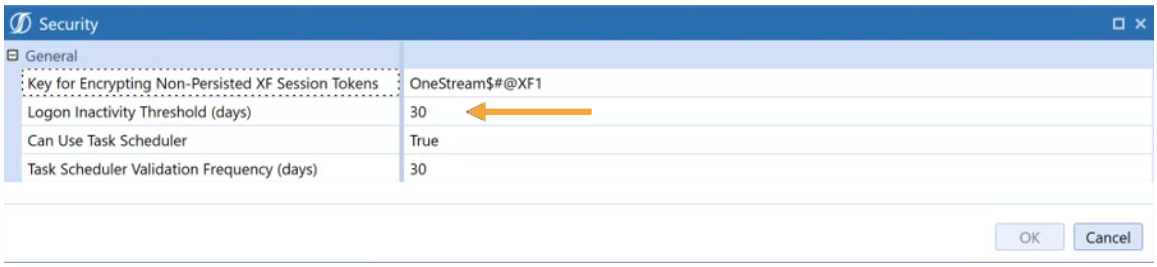
2. Open the configuration file.



3. Click **Security**.

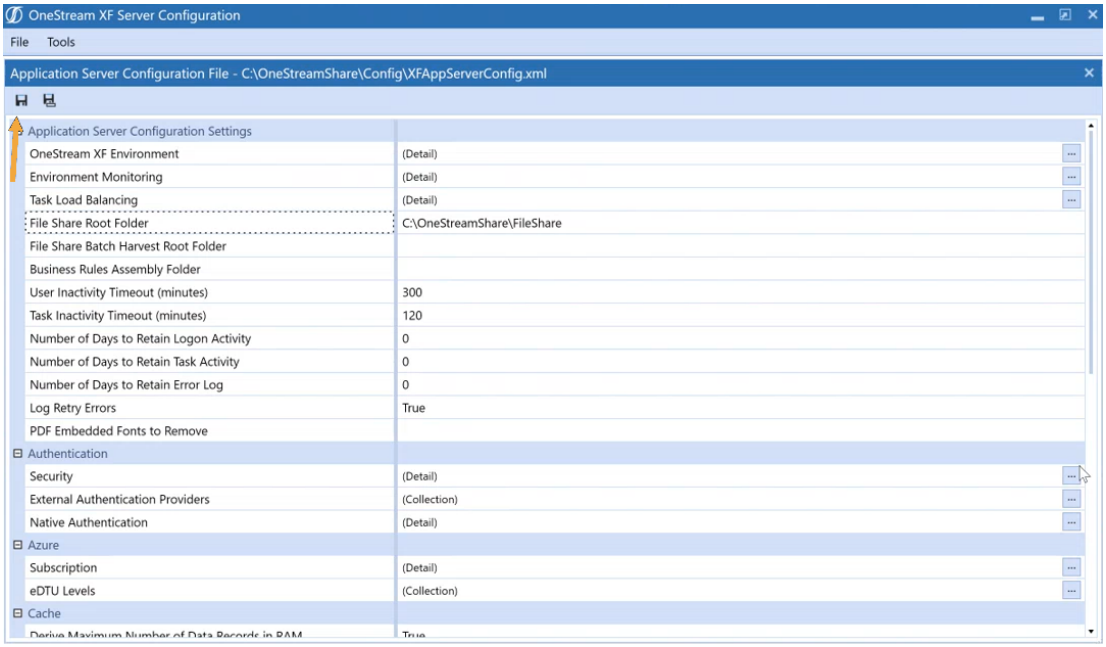


4. Set the Logon Inactivity Threshold (days) to the number of days of inactivity before the user can no longer access the system.



5. Click **OK**.

6. Click **Save**.

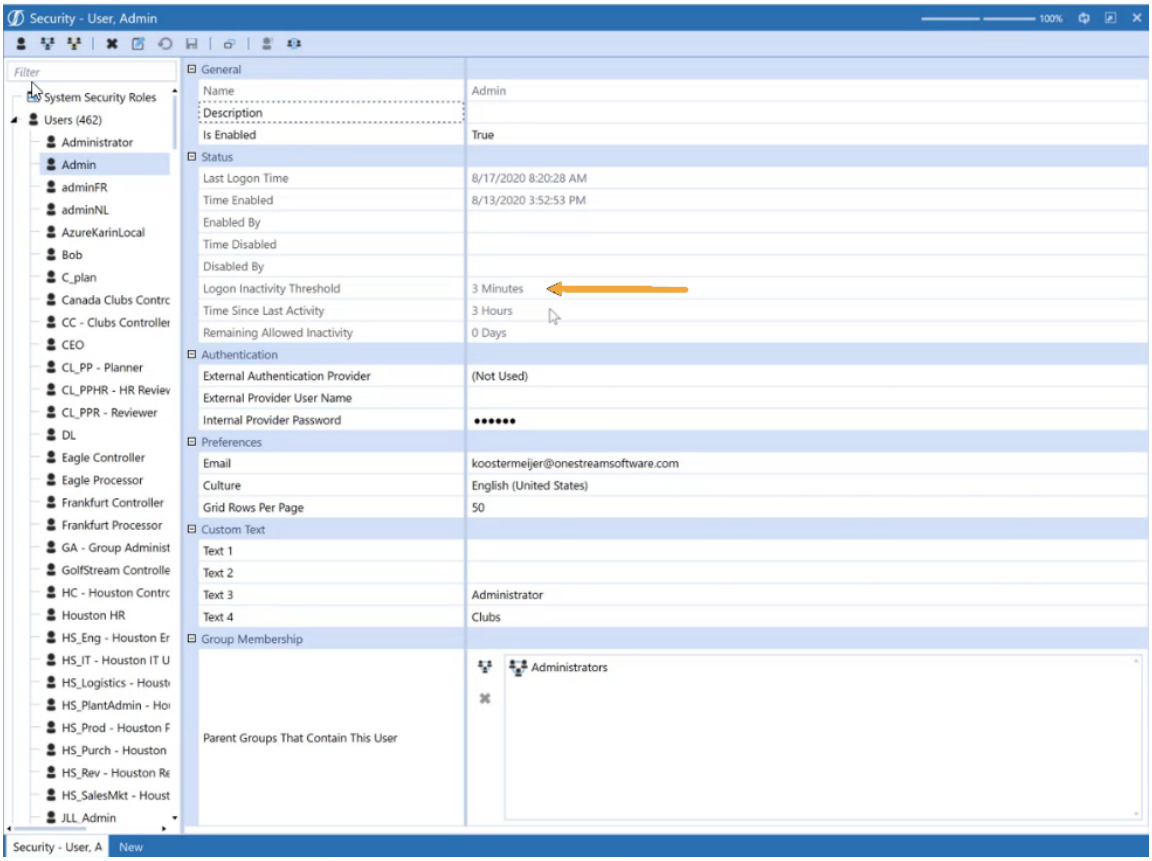


7. Reset IIS to recognize the changes.

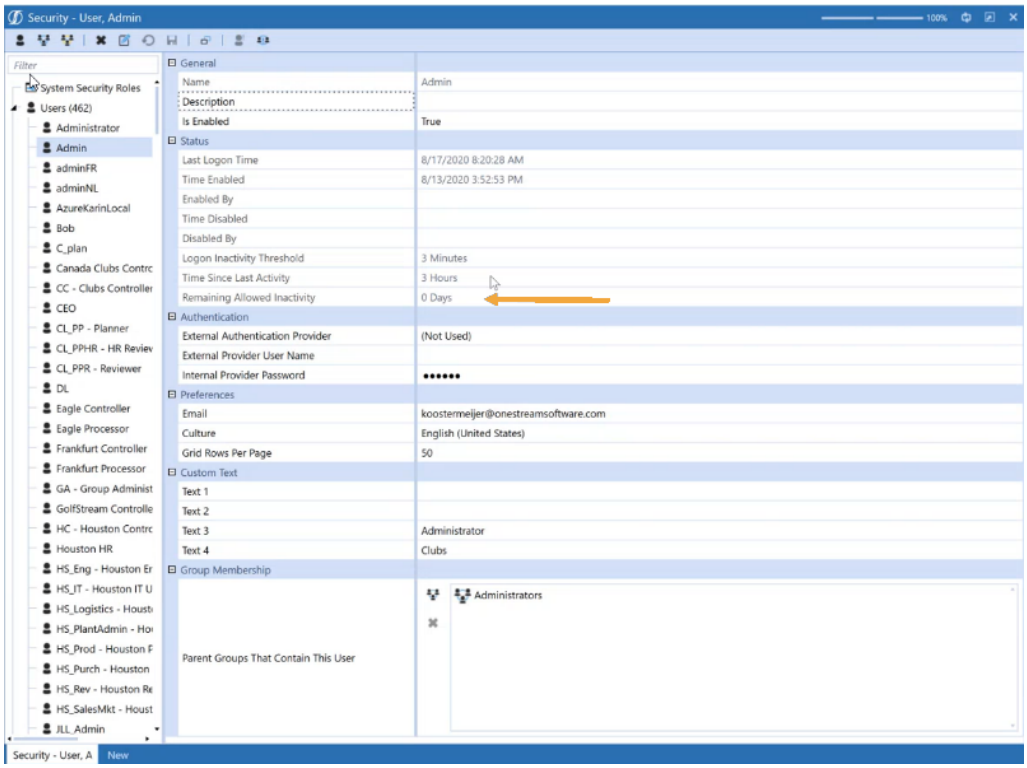
Review Settings

The Administrator is the only role that the inactivity threshold does not apply.

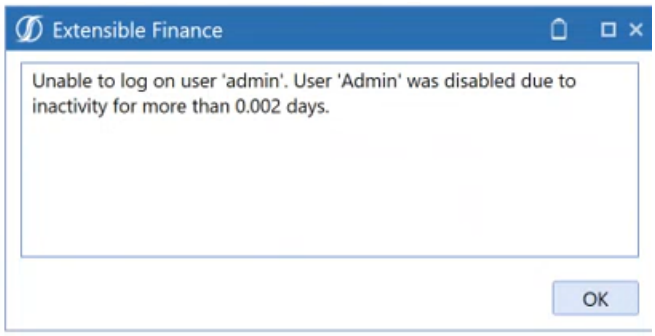
1. Click **System** > **Security**.
2. Click **Users** to review the threshold setting.



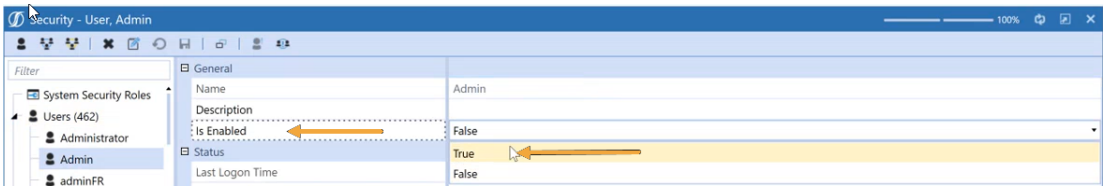
3. If the Remaining Allowed Inactivity is **0 Days**, that means the user no longer has access.



That user gets a message telling them they have been disabled if they log on.



4. To enable their access, click **System**> **Security**.

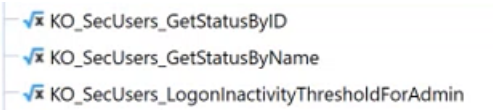


5. Select the user.
6. Click the **Is Enabled** drop-down.
7. Select **True**.
8. Click **Save**.

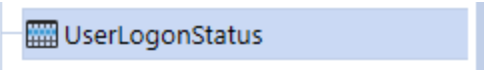
   The Remaining Allowed Inactivity field updates allowing the user to log back in within the time frame specified.

Business Rules

1. Click **Application** > **Business Rules** > **Extensibility Rules**.
2. The three new business rules are GetStatusByID, GetStatusByName, and LogoninactivityThresholdForAdmin.



System Table

1. Go to **System > Database > System Database> Tables**.
2. There is a table name UserLogonStatus 

Business Rules

A Business Rule is a VB.Net Class meaning each Business Rule is an independent object encapsulating VB.Net code. A Business Rule can be a one-line call to write a log message, or it can be a full code library containing other custom VB.Net Classes, Methods and Properties.

This section provides a detailed explanation of the following:

- Platform Engines
- Business Rule structure and fundamentals
- Business Rule Classifications
- Specific Business Rule Types
- Business Rule organization
- OneStream Business Rule framework
- Best practices for Business Rule architecture
- Business Rule organization and referencing

CAUTION: When creating custom business rules that write to the error log, use care to prevent writing sensitive information to the log. When writing to the error log, the OneStream application attempts to filter sensitive information and redact it. Information in the log that is filtered is replaced with `[Redacted]`.

IMPORTANT: Whether it be OneStream, Partner, or Customer, we strongly advise against any sensitive or confidential information be included in business rules.

See the *API Overview Guide*.

Platform Engines

The platform is comprised of multiple processing engines. These engines have distinct responsibilities with respect to system processing and consequently they expose different API interfaces to the Business Rules they call. This section provides a brief overview of each engine in the platform and describes the engine's core responsibilities.

Workflow Engine

The Workflow Engine is thought of as the controlling engine or the puppeteer. The main responsibility of this engine is to control and track the status of the business processes defined in the Workflow hierarchies. This engine is primarily accessed through the BRApi and can be called from other engines in order to check Workflow status during process execution. The Workflow Engine provides a very rich event model allowing each Workflow process to be evaluated and reinforced with customer specific business logic if required (see Appendix 2: Event Listing).

Stage Engine

The Stage Engine performs the task of sourcing and transforming external data into valid analytic data points. The main responsibility of this engine is to read source data (files or systems) and parse the information into a tabular format. This allows the data to be transformed or mapped to valid Members defined by the Finance Engine. The Stage Engine is an in-memory, multi-threaded engine that provides the opportunity to interact with source data as it is being parsed and transformed. In addition to parsing and transforming data, the Stage Engine also has a sophisticated calculation that enables data to be derived and evaluated based on incoming source data. The Stage Engine provides quality services to source data by validating, mapping, and executing Derivative Check Rules.

Finance Engine

The Finance Engine is an in-memory financial analytic engine. The main responsibility of this engine is to enrich and aggregate base data cells into consolidated multi-Dimensional information. The Finance Engine provides the opportunity to define sophisticated financial calculations through centralized Business Rules as well as member specific Business Rules (Member Formulas). It works concurrently with the Stage Engine to validate incoming intersections and works with the Data Quality Engine to execute Confirmation Rules which are used to validate analytic data values.

Data Quality Engine

The Data Quality Engine is responsible for controlling data confirmation and certification processes. This Confirmation Engine is used to define and control the sequence of data value checks required to assert the information submitted from a source system is correct. The Certification Engine is responsible for managing user certifications and determining the Workflow dependents' completion status. This engine is primarily accessed through the BRApi and may be called from other engines in order to check data quality status during process execution.

Data Management Engine

The Data Management Engine provides task automation services to the platform. This engine executes batches of commands that are organized into sequences which contain steps. Steps represent entry points or mechanisms to execute features of other engines. For example, the Clear Data Step uses the services of the Finance Engine. In addition, the Data Management Engine can execute a Business Rule Step which executes a custom Business Rule as part of a Data Management Sequence. This is an incredibly powerful capability because it provides the ability to string together any combination of predefined processing steps with custom Business Rule steps.

Presentation Engine

The Presentation Engine provides extensive data visualization services to the Platform. The Presentation Engine is made up of the following component engines: Cube View Engine, Dashboard Engine, Parameter Engine, Book Engine and Extensible Document Engine. The Presentation Engine is responsible for managing and delivering content to the end user as well as providing a development environment for custom user interface elements. This engine enables theOneStream Solutionapplication development capabilities and continues to evolve with each product release. Like the Data Management Engine, the Presentation Engine interacts with and can call the services of all other engines in the product.

Scaling Engine

This feature will be made available in a future release.

The Scaling Engine provides services that will determine whether the customer wants to Scale their Server Set or Database Elastic Pool on the Platform. This is only available to Cloud (Azure) and does not pertain to On-Premise solutions. For example, customer must be utilizing Azure Scale Set, and/or SQL Server Elastic Pool functionality. This provides the ability to create or delete a VM and/or increase/decrease database resources based on the logic that is designated in the System Extender Business Rules to meet the customer needs.

Data Management Automation With PowerShell

PowerShell is an object-oriented programming language and interactive command line shell for Microsoft Windows. It was designed to automate system tasks, such as batch processing and create systems management tools for common processes.PowerShell includes over 130 standard command line tools for functions that formerly required users to create scripts in VB, VBScript or C#.

PowerShell offers a variety of ways to automate tasks which include:

- Cmdlets : Very small .NET classes that appear as system commands.
- Scripts: Combinations of cmdlets and associated logic.
- Executables: Standalone tools.

Instantiation of Standard .NET Classes

PowerShell integrates with the .NET environment and can be embedded in other applications. Over one hundred cmdlets are included and can be used separately or combined to automate more complex tasks. You can also create and share cmdlets.

PowerShell is built into Windows Server 2008 and Windows 7, provided as an optional feature during installation. You can use Windows Task Scheduler to automate PowerShell script execution.

Using PowerShell Script Editor

To run PowerShell on Windows 7 or a later version:

- Select Windows > Start, then enter "PowerShell".
- Select All Programs > Accessories |Windows PowerShell.

You can use these programs with PowerShell:

- Windows PowerShell ISE: An integrated script editor you can use to type PowerShell commands and to edit and run PowerShell script files. These are text files with a ps1 extension.
- Windows PowerShell: A command line execution tool similar to a DOS prompt. You can use this tool to run commands or script files but you cannot create or modify scripts.

Configuring PowerShell for the Client API

Perform these steps on each machine that will use PowerShell before it can be used with the Client API:

1. Execute a PowerShell command enabling the execution of unsigned scripts.
2. Create or alter the PowerShell execution and IDE configuration files, so the script engine understands how to use the .Net Framework v4.0.
3. Install OneStream Studio on each machine executing PowerShell scripts.

Running Unsigned Scripts

Initially, this code needs to run in a PowerShell command prompt to enable PowerShell to run unsigned scripts created on the local computer:

```
set-executionpolicy remotesigned
```

Configuration for .Net Framework v.4.0

To use the OneStreamClientApi with PowerShell, PowerShell must be configured to use the .NET Framework v4.0. Perform these tasks:

1. Modify or create these files if they do not exist in C:\Windows\System32\WindowsPowerShell\v1.0:
    - powershell.exe.config
    - powershell_ise.exe.config
2. Add this required content to each file:

```
<?xml version="1.0"?>
<configuration>
    <startup useLegacyV2RuntimeActivationPolicy="true">
        <supportedRuntime version="v4.0.30319"/>
        <supportedRuntime version="v2.0.50727"/>
    </startup>
</configuration>
```

    For more information, see: http://tfl09.blogspot.com/2010/08/using-newer-versions-of-net-with.html.

3. Install OneStream Studio.

Learning PowerShell

Microsoft provides extensive resources to help IT professionals leverage PowerShell. For more information, see: http://technet.microsoft.com/en-us/scriptcenter/powershell.aspx

Using the Client API in a PowerShell Script

OneStreamClientApi specifically enables PowerShell scripts to call a function. This API exposes functions for authentication and Data Management. OneStream expanded the number of functions exposed to this API. The Client API component is installed with the Client API Installer. This API offers a simple set of functions enabling script writers to connect to the server, authenticate, execute OneStream Data Management Sequences, and perform basic data retrieval.

Client API Object Hierarchy

OneStreamClientAPI

LogonInfo

Type
LogonInfo

SI

Type
SessionInfo

Authentication

Logon

Parameters
string webServerUrl
string userName
string password
XFClientAuthenticationType clientAuthenticationType

Return Value
LogonInfo

Logoff

Parameters
None

Return Value
None

OpenApplication

Parameters
string Application

Return Value
LogonInfo

LogonAndOpenApplication

Parameters
string webServerUrl
string userName
string password
string application
XFClientAuthenticationType clientAuthenticationType

Return Value
LogoInfo

EncryptPassword

Parameters
string clearTextPassword
XFClientAuthenticationType clientAuthenticationType

Return Value
string

DataManagement

ExecuteSequence

Parameters
string sequenceName
string customSubstVarsAsCommaSeparatedPairs

Return Value
DataMgmtResult

ExecuteStep

Parameters
string dataMgmtGroupName
string stepName
string customSubstVarsAsCommaSeparatedPairs

Return Value
DataMgmtResult

DataProvider

GetAdoDataSetForCubeViewCommand

Parameters
string CubeViewName
bool dataTablePerCubeViewRow
CubeViewDataTableOptions dataTableOptions
string resultDataTableName
Dictionary<string, string> customSubstVars
bool throwExceptionOnError

Return Value
DataSet

GetAdoDataSetForMethodCommand

Parameters
XFCommandMethodTypeId xfCommandMethodType
string methodQuery
string resultDataTableName
Dictionary<string, string> cistomSubstVars
bool throwExceptionOnError

Return Value
DataSet

Exposing Data Management Automation through OneStream Web API

OneStream Web API is a RESTful web service designed to expose OneStream Data Automation functions when interacting with third-party API client applications.

OneStream Web API must be installed on a web server. It also must be configured for external authentication providers supporting OAuth2.0/OpenID Connect authorization protocol. Identity Providers currently supported are Okta, Azure AD (Microsoft Entra ID), and PingFederate.

OneStream Web API is API client agnostic. It accepts and outputs data in JSON format making it possible for every API client application that supports this format to also interact with this service. One of the most widely used API clients is Postman, a Windows app. For more information about how to configure OneStreamWeb API to interact with Postman see the autogenerated documentation at http(s)://[servername]:[port]/onestreamapi.

OneStream Web API endpoints:

Authentication endpoint. Represents a RESTful service for Authentication.

- POST api/Authentication/LogonAndReturnCookie

  Used primarily by the Enablement Team to verify Web API installation completed successfully. Returns a one-time cookie value that holds authentication state or a message indicating failure along with a proper HTTP code.

DataManagement endpoint. Represents a RESTful service of Data Management.

- POST api/DataManagement/ExecuteSequence:

  Executes a Data Management Sequence and returns a success/failure message along with a proper HTTP code.

- POST api/DataManagement/ExecuteStep

  Executes a Data management Step and returns a success/failure message along with a proper HTTP code.

DataProvider endpoint. Represents a RESTful service of Data Provider

- POST api/DataProvider/GetAdoDataSetForAdapter

  Returns a JSON representation of a DataSet a given Dashboard Adapter or a failure message along with a proper HTTP code.

- POST api/DataProvider/GetAdoDataSetForCubeViewCommand

  Returns a JSON representation of a DataSet for a given Cube View or a failure message along with a proper HTTP code.

- POST api/DataProvider/GetAdoDataSetForSqlCommand

  Returns a JSON representation of a DataSet for a given SQL Query or a failure message along with a proper HTTP code. Administrator role is required for this functionality.

- POST api/DataProvider/GetAdoDataSetForMethodCommand

Returns a JSON representation of a DataSet for a given pre-defined list of method commands used by XFDataProvider to fill a DataSet or a failure message along with a proper HTTP code. Administrator role is required for this functionality.

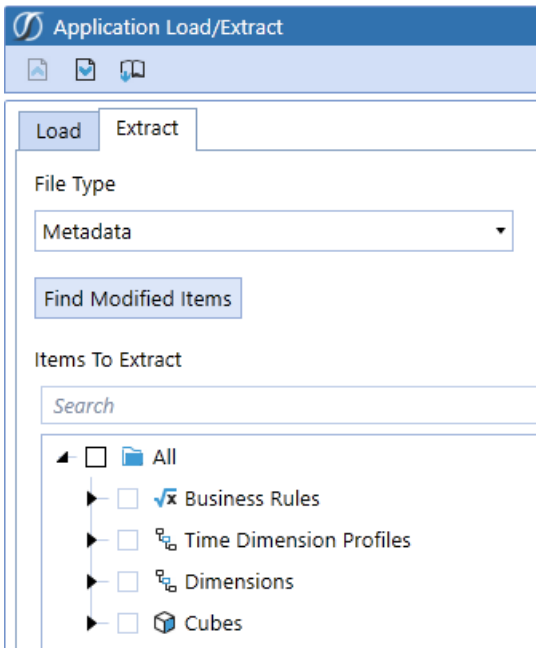
Extracting and Loading Dimensions

The Application Load/Extract can find Dimension changes and export them into an xml in order to make the metadata migration process easier.

You can extract applications as an XML or zip file. If the file size is larger than 2Gb, you must use a zip file.

When importing, you may have extracted an XML file that was larger than 2Gb in size. You must do a zip extract to successfully load the file.

NOTE: This feature only applies to Dimensions.

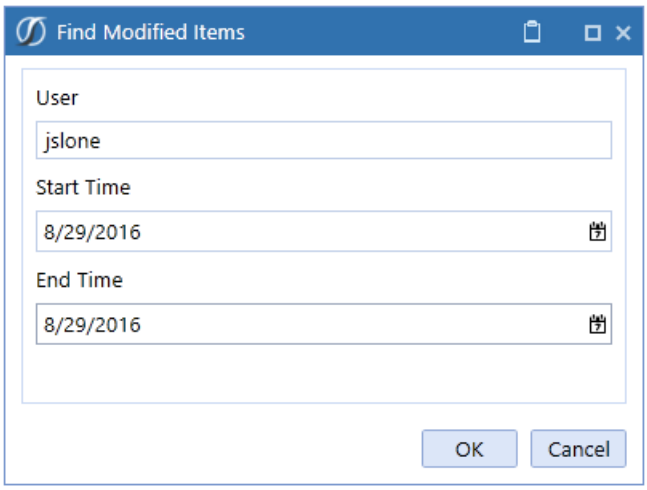


Extracting Dimension Changes

Find Modified Items

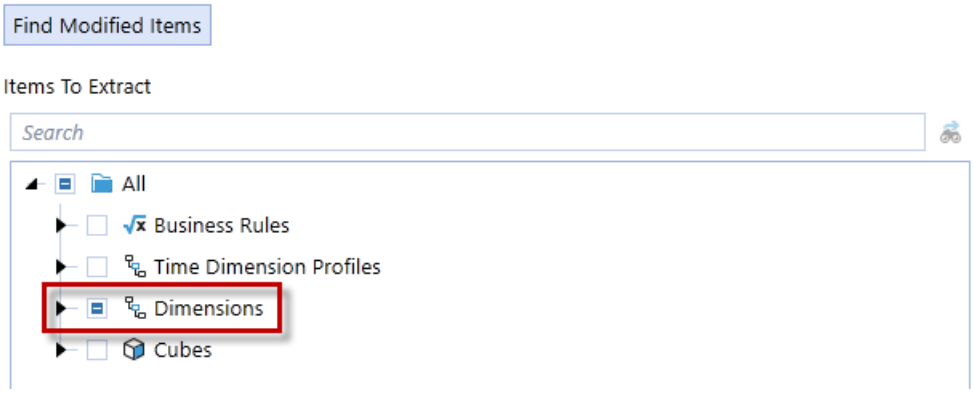
Modified Items can consist of adding/deleting Dimension Members and/or adding/deleting Dimension Relationships. Click Find Modified Items to launch the dialog. This allows users to find Dimension changes by time and user.

1. Enter a user name and a range of time in order to find Dimension Member and Dimension Member Relationship changes. Leave the User field empty in order to find changes by all users.



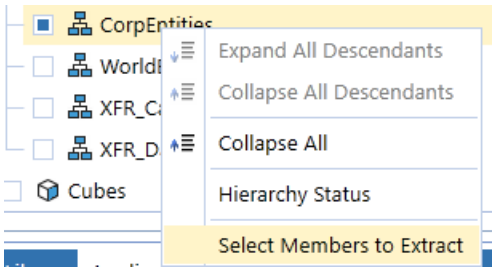
   For example, in the application's Dimension Library, a Member was created, another Member was deleted, and a relationship was deleted. Once the user and time filters are entered, click **OK**, and these changes will be highlighted.

   The hierarchy will indicate where changes were made. The example below indicates there was a partial change to the Dimensions.

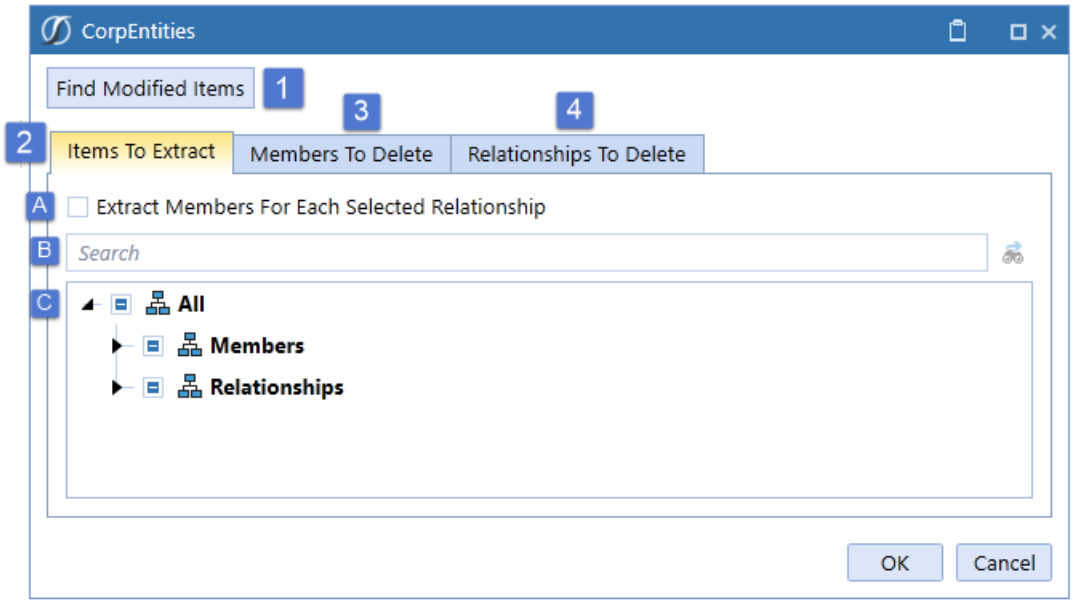


2. Expand this to see what Dimensions were changed.

3. Right-click on the Dimension name and click **Select Members to Extract**.



This launches an extraction dialog for the specific Dimension selected.



4. Find Modified Items

   By default, everything is selected. Click Find Modified Items and enter the username and desired date range in order to find the specific changes made to the selected Dimension.

5. Items to Extract Tab

   This tab displays the changes found in the hierarchy. Users can also manually select Members and Relationships to extract.

6. Extract Members for Each Selected Relationship Check Box

   Check this box in order to include the Members used in the selected relationships.

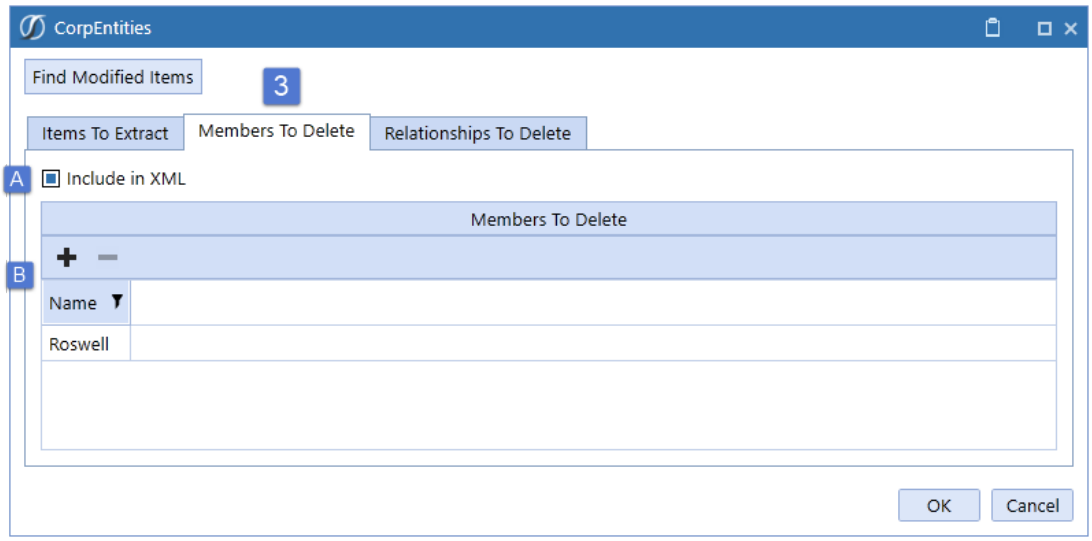
7. Search

   Search for a specific Member in the selected Dimension.

8. Dimension Hierarchy

   Scroll through the Dimension hierarchies in order to see where changes occurred and/or manually select or de-select Members or Relationships to extract.

> **NOTE:** Right-click on any Parent Member under Relationships and click Select All Descendants in order to select all Child Members within the hierarchy and include them in the extraction.



9. Members to Delete Tab

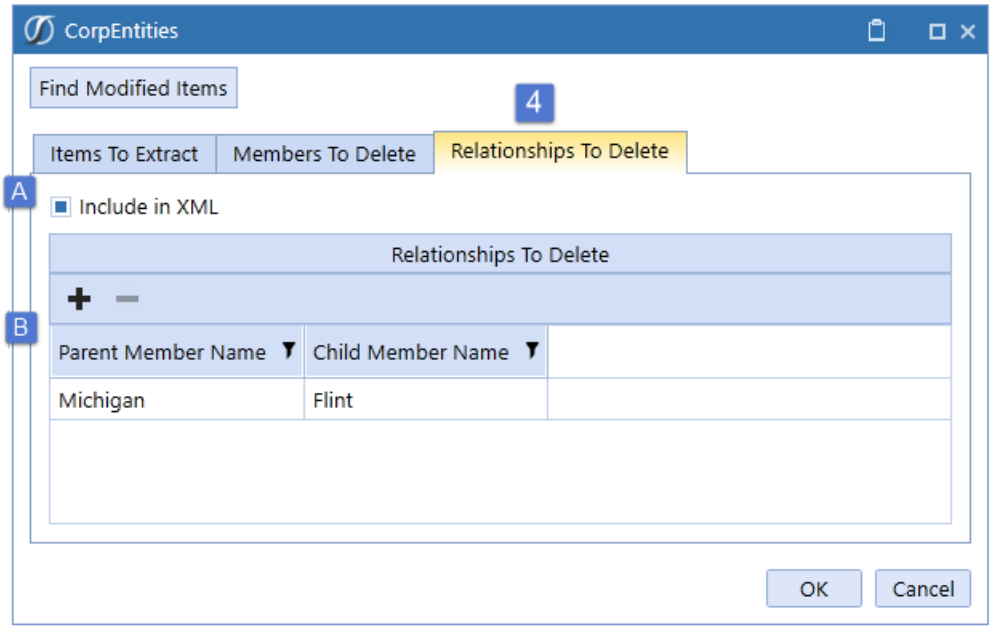
   The Members to Delete tab displays any deleted Members found within the User/Time Parameters.

10. Include in XML Check Box

    By default, Include in XML is not checked. Check this box in order to include the displayed Members in the extract. If checked, any Members to Delete will have an Action=Delete in the xml file and be deleted upon loading the xml into another application.

11. Any deleted Members found within the User/Time Parameters will display here. Click the plus sign to manually add additional Members to delete. To exclude a particular entry from the xml, select the line to exclude and click the minus sign.

> **NOTE:** Members with stored data cannot be deleted from an application.



12. Relationships to Delete
    The Relationships to Delete Tab displays any deleted relationships found within the User/Time Parameters.

13. Include in XML Check Box
    Check this box in order to include the displayed Member relationships in the extract. If checked, any Relationships to Delete will have an Action=Delete in the xml file and be deleted upon loading the xml into another application.

14. Users can manually add additional relationships to delete by clicking the plus sign and indicating the Parent and Child. To exclude a particular entry from the xml, select the line to exclude and click the minus sign.

Loading the XML File

When importing an extracted xml file, it processes in the following order:

1. New or changed Members
2. Deleted Members
3. New Relationships
4. Deleted Relationships

> **NOTE:** The processing order is important because if any errors occur during the xml file load, OneStream applies as many modifications as it can up to the point of the error. For example, an xml file contains ten Member changes, three deleted Members and one new relationship. If an error occurs when trying to delete the first Member, the ten Member changes will still take place because they are processed first during the xml load. Any modifications in the xml prior to the error will occur and any after the error will not.

If a user receives an error during the load process, the error must be resolved in order to complete the metadata migration.

Common Load Errors and Resolutions

1. Deleting a Member with data
   If a Member without data is deleted from the source application, but that same Member contains data in the destination application, an error occurs and the Member will not be deleted. To resolve the error, do one of the following:
   a. Clear the Member's data in the destination application and reload the xml file.
   b. Create a new xml extract excluding that Member from the file.
   c. Edit the xml file to exclude the Member and the action.

2. Deleting a Member without Data

   Care should be taken when deleting Entity members. Even though the member may not have data, it may be in use as an Intercompany Partner. Ensure the member is not in use on any data records, or on records as an Intercompany Partner.

3. Undefined Security Group

   If a security group is assigned to a Dimension Member in the source application, but does not exist in the destination application, an error will occur. To resolve the error, do one of the following:

   a. Create the security group in the destination application and reload the xml file.

   b. Create a new xml extract excluding this Member and its changes from the file.

   c. Edit the xml file to exclude the Member and the action.

4. Undefined FX Rate Type

   If a FX Rate Type is assigned to a Scenario Member in the source application, but does not exist in the destination application, an error will occur. To resolve the error, do one of the following:

   a. Create the FX Rate Type in the destination application and reload the file.

   b. Create a new xml extract excluding this Member and its changes from the file.

   c. Edit the xml file to exclude the Member and the action.

5. Invalid Characters in the XML File

   If the xml file was edited and invalid characters were entered for a Member name, an error will occur. To resolve the error, do one of the following:

   a. Make the Member modifications in the source application and extract an xml file without invalid characters.

   b. Edit the xml file and remove the invalid characters.

Project Extract and Load

This extract is for Application Project Designers who are building solutions that span many artifacts, such as Dashboard Maintenance Units, Business Rules, Cubes, Dimensions, Cube Views, etc. A good example is a person designing a solution to be hosted on MarktetPlace. This application Extract and Load option allows all the defined objects, such as Dashboards and Business Rules, to be collected as a single file export package and to be later reloaded as a package. XFProject is used as a convenient way to organize OneStream Solution or similar solutions into a folder structure which can be integrated with a version control system such as “Git.” Doing so could enable more than one team member to work on a solution simultaneously. The developer must create an XML file which is the definition for the contents of the Project export.

Project File

The Application Designer must first manually define an XML file to support the export of objects as aF Project File. The file is saved with the file extension of .xfProj and saved to a local project folder which could also support a version control system.

Sample File:

```
<xfProject topFolderPath="" defaultZipFileName="">
  <projectItems>
    <projectItem projectItemType="DashboardMaintenanceUnit" folderPath="" name="Corporate" includeDescendants="true" />
    <projectItem projectItemType="DashboardMaintenanceUnit" folderPath="" name="Corporate Templates" includeDescendants="true" />
    <projectItem projectItemType="DashboardComponent" folderPath="" name="btnSaveAndCalculate" includeDescendants="true" />
    <projectItem projectItemType="BusinessRule" folderPath="" name="FXRates" includeDescendants="true" />
  </projectItems>
</xfProject>
```

File Structure:

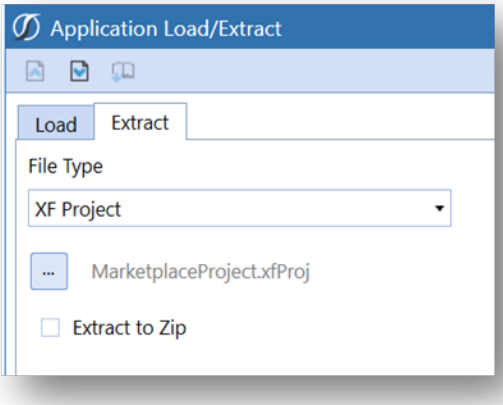
- xfProject: The root node to start a .xfProj which contains two attributes:
    - TopFolderPath: Will create and define the starting folder location of where the specified files are extracted to.
    - DefaultZipFileName: Will create a standard default file name for .zip file extracts.
- projectItems: A list structure containing the project items needed to extract (no attributes needed).
- projectItem: The item reflecting what is needed to extract from OneStream or load from the file system. It has 4 attributes:
    - ProjectItemType:
        - BusinessRule
        - Cube
        - CubeViewGroup
        - CubeView
        - CubeViewProfile
        - DashboardMaintenanceUnit
        - DashboardFile
        - DashboardString
        - DashboardParameter
        - DashboardGroup
        - DashboardAdapter
        - DashboardComponent
        - Dashboard

        - DashboardProfile
        - DataManagementGroup
        - DataManagementStep
        - DataManagementSequence
        - DataManagementProfile
        - DataSource
        - Dimension
        - TransformationRuleGroup
        - TransformationRuleProfile
    - FolderPath: The name of the sub folder where the project item type is extracted to.
    - Name: The name of the project item.
    - IncludeDescendants: Default is true and only affects these project item types:
        - CubeViewGroup
        - DashboardGroup
        - DashboardMaintenanceUnit
        - DataManagementGroup

File Extract

The .xfProj file is placed in a local folder, such as the user's desktop. The defined folderpath folders will be generated here as the target location for application exports and loads. There are two file extract options available on the Windows App.

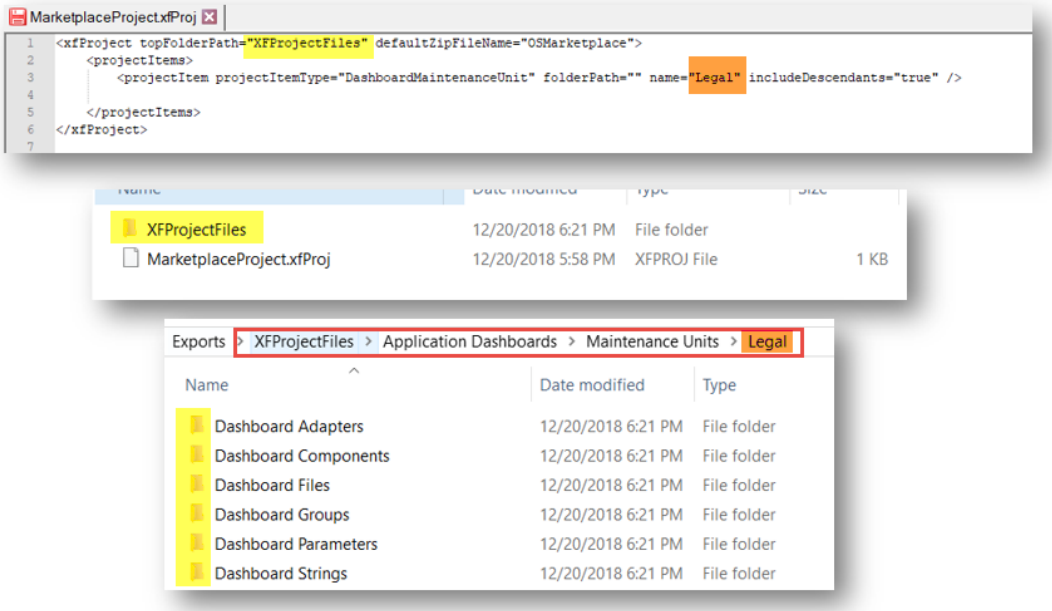
- .zip: The export option will collect all the objects defined in the .xfProj file as a Zip file to the location of the .xfproj file.
- File: The standard export will export all the objects defined in the .xfProj file to the folderpath locations defined in .xfproj file.



1. Navigate to **Application > Load > Extract**.
2. Select the **Extract**tab.
3. Browse and select the **.xfProj** file.
4. (Optional) From the OneStream Windows Client, select or de-select **Extract to Zip** as required.
5. Click the **Extract toolbar** button.

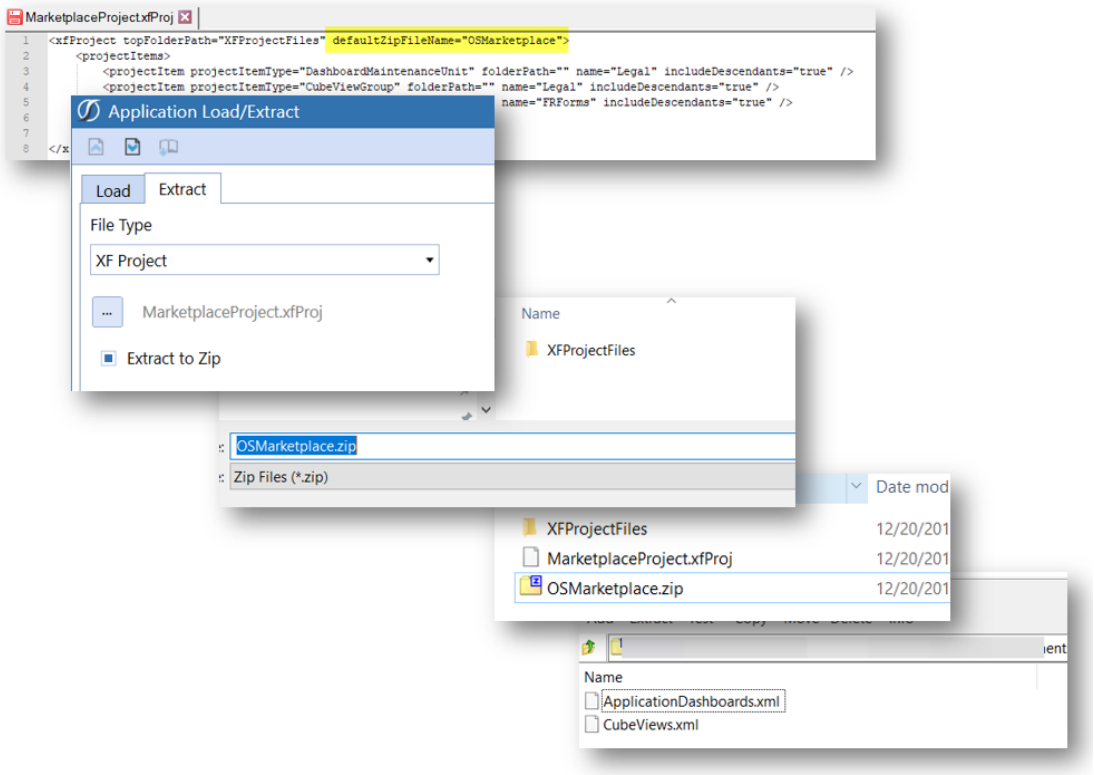
Example:

The contents will be generated in the defined folder paths.



Zip Extract

The zip file extract will create an application zip file containing all the objects defined.

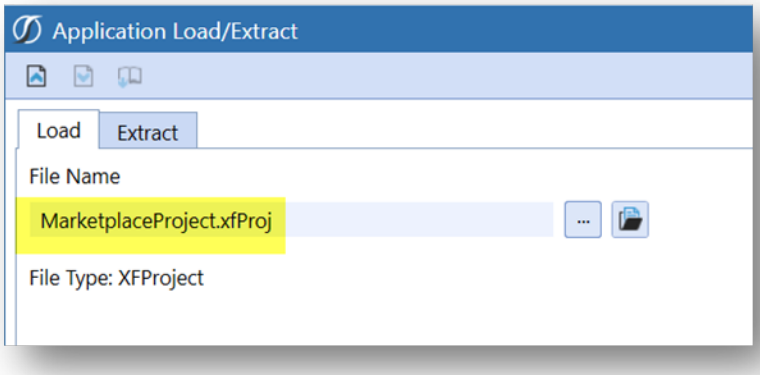


File Load .xfProj

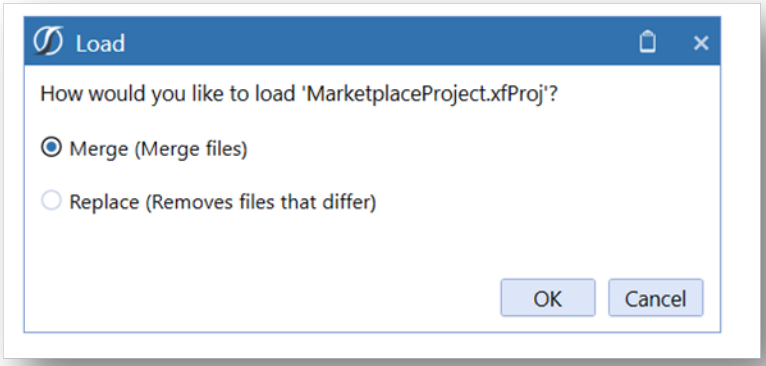
The file load using the defined .xfProj file provides a seamless link to the project files. When loading an .xfProj file, the user is presented with options to “Merge” or “Replace” the target files. The only files affected are those defined by the .xfproj file.

> **NOTE:** If you select **Replace**, it will only remove files that differ for CubeViewGroups, DataManagementGroups, DashboardMaintenanceUnits, and DashboardGroups. For all other items (such as, business rules or extensibility rules), if you select **Replace**it will act as a Merge.

1. Navigate to **Application > Load > Extract**.
2. Click the **Load**tab.
3. Browse and select the **.xfProj** file.
4. Click the **Load toolbar** button.



5. Select the Load Method, Merge or Replace.



Zip Load

The zip file load functions as any other application file load. The contents of the file are merged into the application. The zip file load is not supported by alternative merge or replace file load options.

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.

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
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

Dimension Token	Dimension Type
T#	Time
C#	Consolidation
O#	Origin
V#	View

Configurable Dimensions

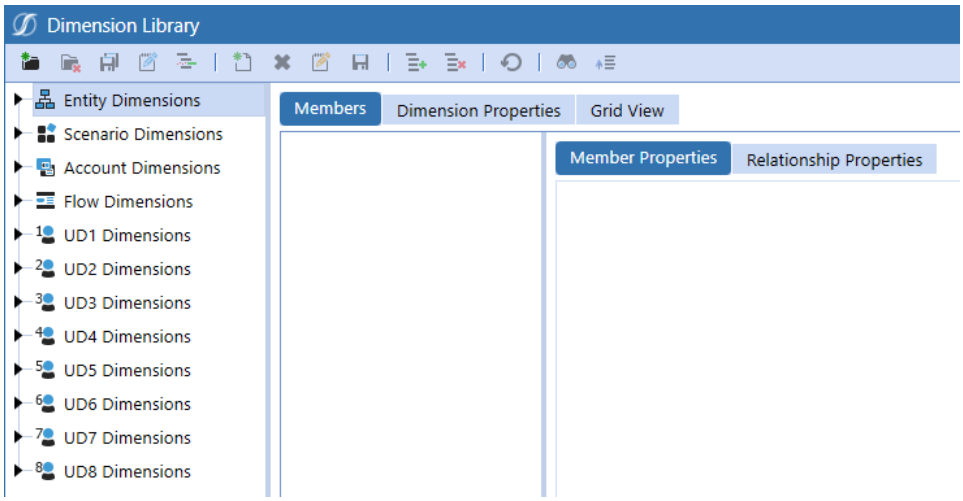
Configurable dimension types include entity, scenario, account, flow, user defined, parent, intercompany, time, and consolidation.

- 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.

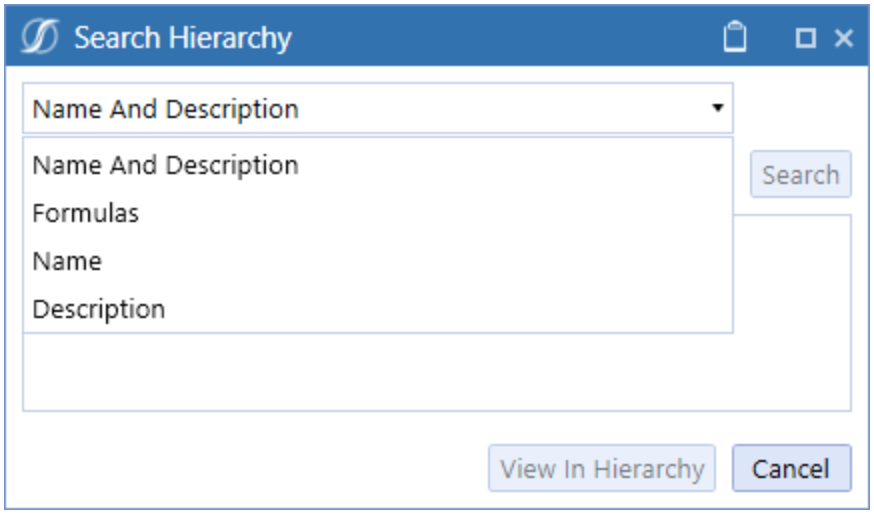
  > **NOTE:** It does not rename the member in other areas.

- **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 also produce every hierarchy in which a member appears. Search using Name and Description; Formulas; Name; or Description.



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

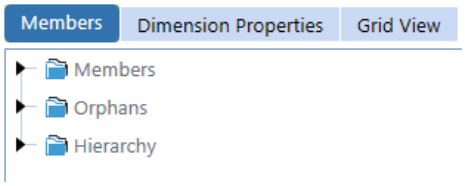
The following icon displays in all dimension security properties.

- **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, click the **Members** tab.

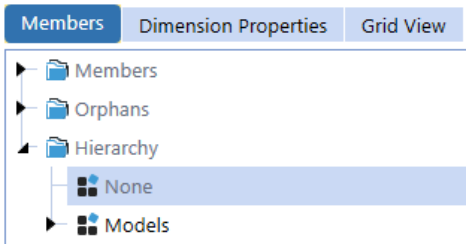
In the Members tab, dimensions can be viewed by members, orphans, or in a hierarchy.



- **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:

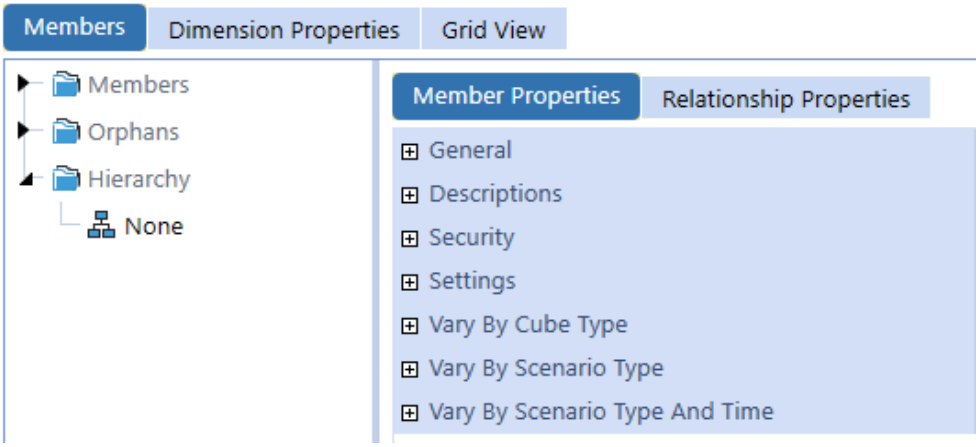
- Typing directly in the field
- Clicking the ellipsis to open a dialog box
- Clicking the **Navigate to Security** icon in the Security section
- Double-clicking in a field to generate a drop-down menu

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. Click 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.

The following properties only displays 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

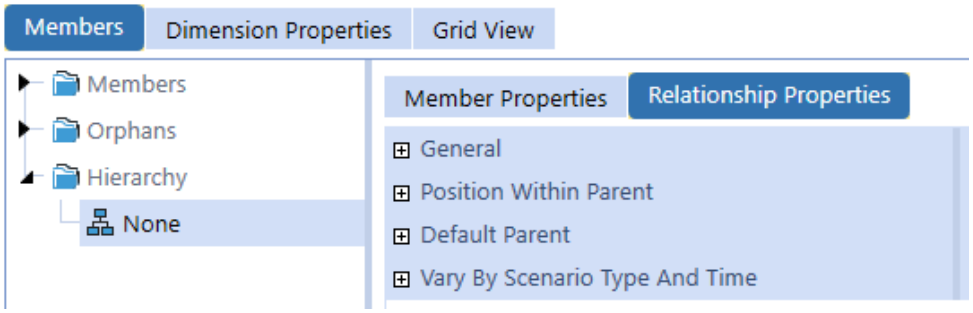
Default Description: A description of the member in the dimension. See Report Alias.

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. Click 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.

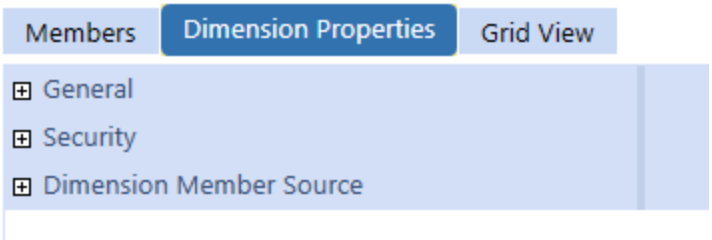
Position Within Parent

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 click 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.

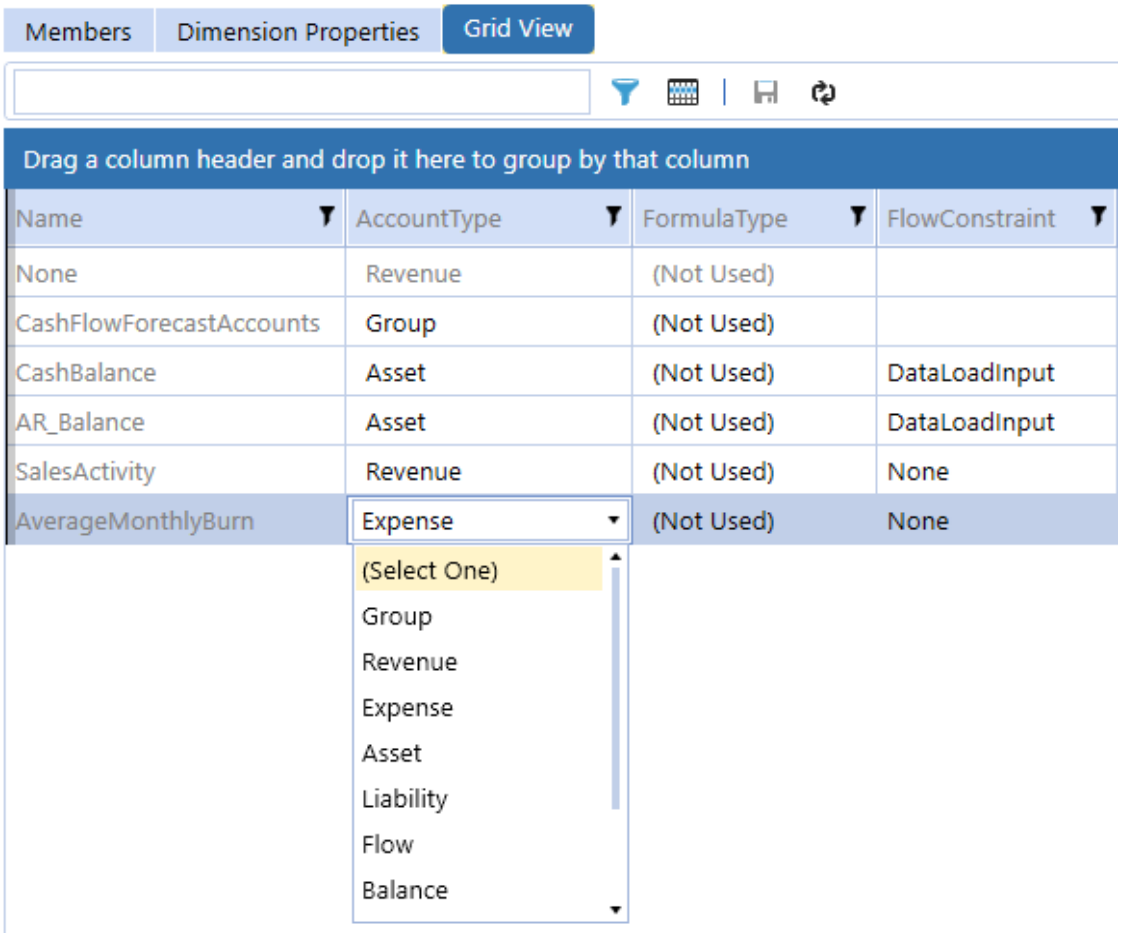
- **Source Path**: If the Source Type is Business Rule or XBRL, enter the business rule name or the full file system path for the XBRL taxonomy link definition file. For example, Internal/XBRL/Taxonomies/EntryPoints/. You can also enter a Workspace Assembly Service to run for a Dynamic Dimension. For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

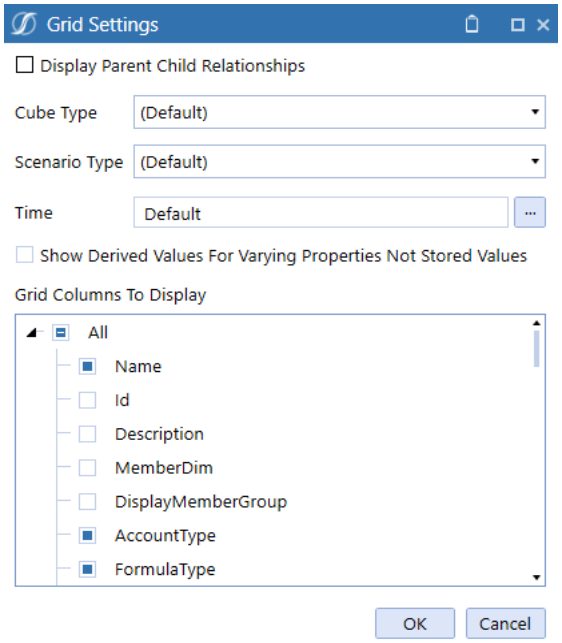
- **Member Filter**: Narrows the list to view specific members. See Member Filter Builder.
- **Grid Settings**: Changes the data displayed.
- **Save**: Saves changes.
- **Refresh**: Updates the data.

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. Click **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. Click the **Create Dimension** icon.
3. Enter a name.
4. Click the **OK** button.

Create a Member

To create a new member in the dimension:

1. Select the dimension.
2. Click 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. Click 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. Click 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. Click the **Relationships Properties** tab.
3. Click 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. Click the **Sibling Member** drop-down menu.
6. Select the sibling member.
7. Click 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. Click **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. Click **Copy Selected Members**.
3. Right-click the name of another member.
4. Select the paste option:
   - Click **Paste Relationships (Add)** to create a new relationship for the copied members.
   - Click **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. Click the **Add Relationship For Selected Member** icon.
3. Click the ellipsis next to **Child Member** to select the child member.
4. Click 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. Click 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. Click 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
\	Back slash
{ }	Brace (left and right)
[ ]	Bracket (left and right)
^	Caret
,	Comma

Character	Name and Description
=	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. So, it is recommended to not use it.
|	Pipe or vertical bar
+	Plus sign
?	Question mark
"	Quotation marks
;	Semicolon

Character	Name and Description
/	Slash

Reserved Words

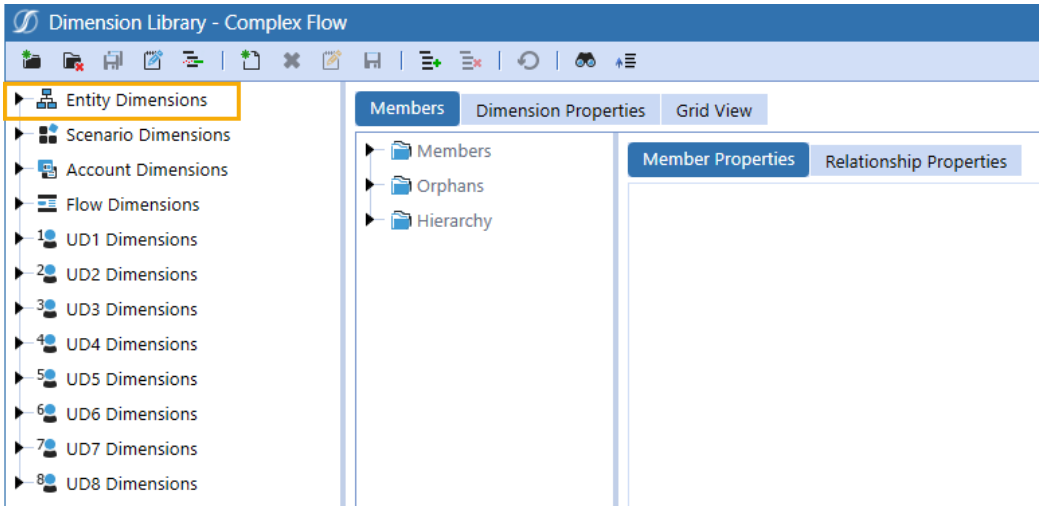
Reserved words cannot be used in dimensions or other structural application components, such as cubes.

- 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. 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
- 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.

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.

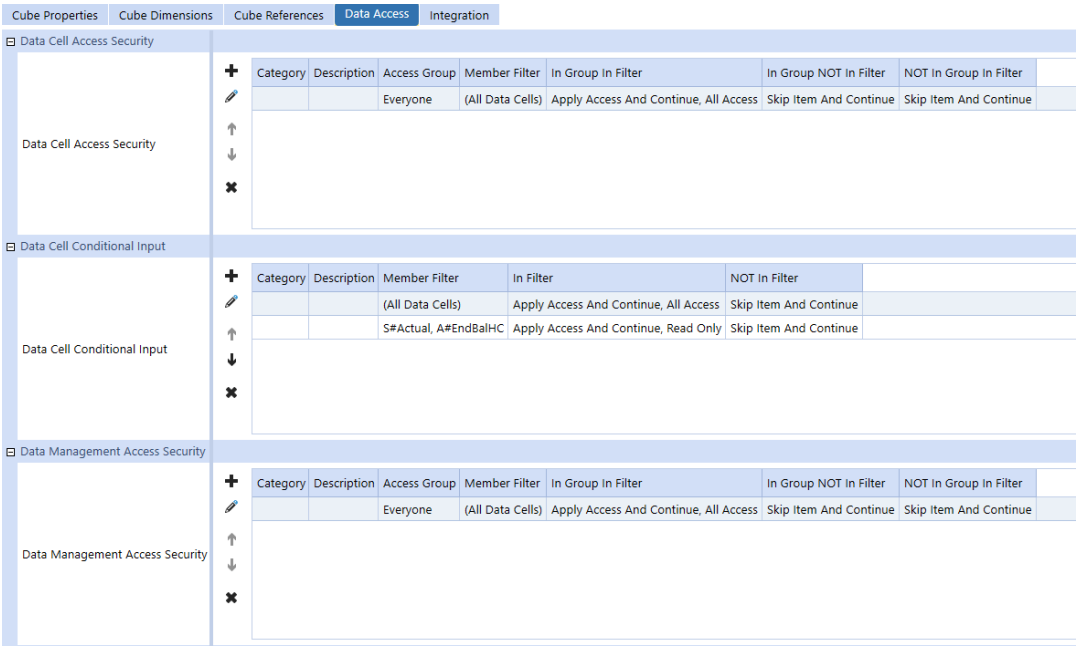
> **TIP:** Click the ellipsis button and begin typing the name of the Security Group in the blank field. As the first few letters are typed, the groups are filtered making it easier to find and select the desired group. Once the group is selected, press **Ctrl** + double-click to enter the correct name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



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.

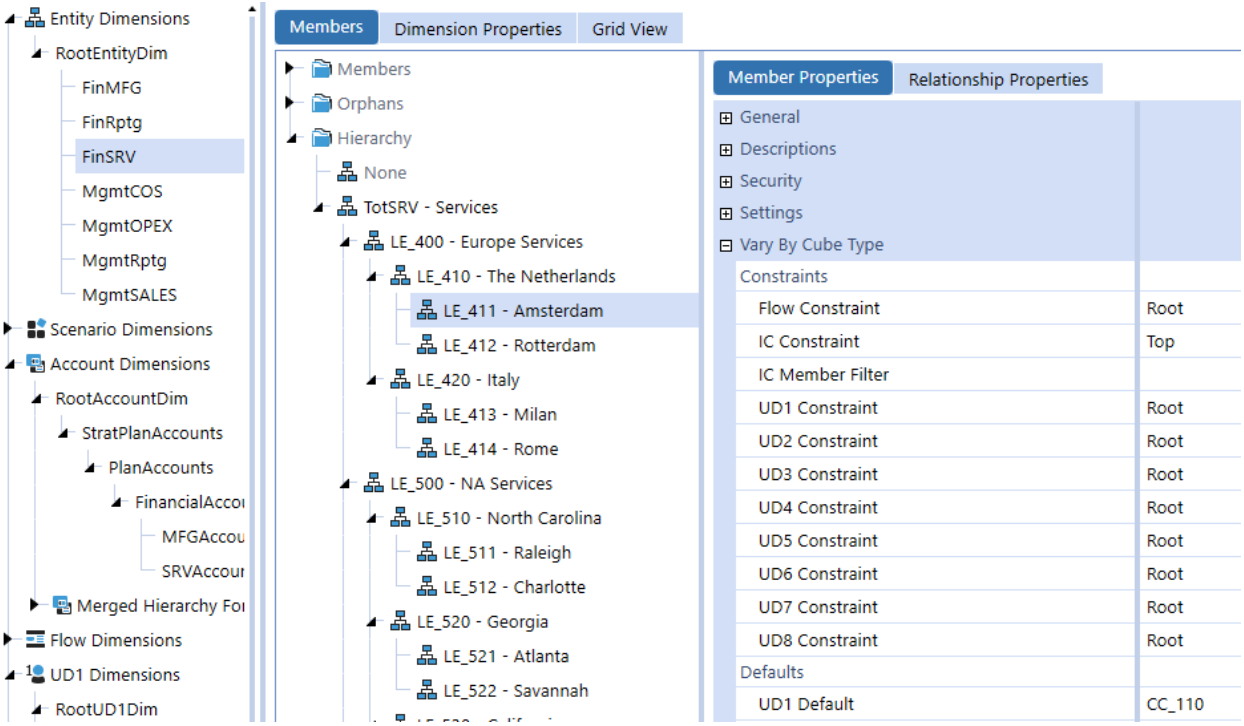
IC Constraint: This is the intercompany dimension constraint. This entity can only use the members with this child or members under a selected parent member. Setting entity constraints will define the data intersection as green, no input cells.

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. 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.

- 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
- Forecast
- FXModel
- Operational
- Plan
- Variance
- ScenarioType1-8

- Budget
- Control
- Flash
- History
- LongTerm
- Model
- Sustainability
- Target
- Tax
- 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. 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
- Workflow
- Settings
- FX Rates
- Hybrid Source Data
- Custom Settings

Security

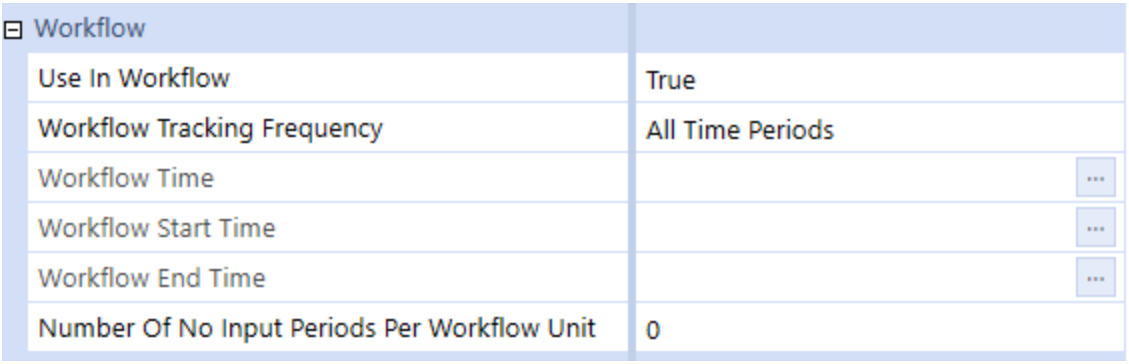
Use Security settings to control the level of access organization members have to a scenario.

Click 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. See Input Frequency.

> **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. Click 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.

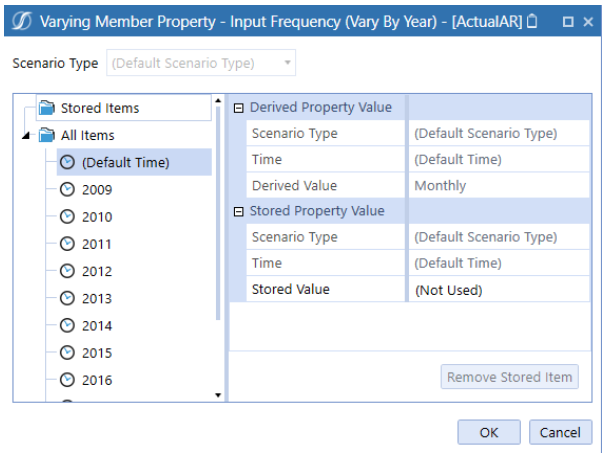
Settings

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. Click 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.



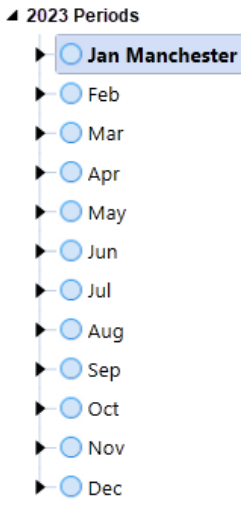
3. In the **Stored Value** field, select the frequency.
4. Click the **OK** button.

> **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. See Workflow. If the Input Frequency varies by year, the Workflow Tracking Frequency updates the workflow view accordingly. See examples below:

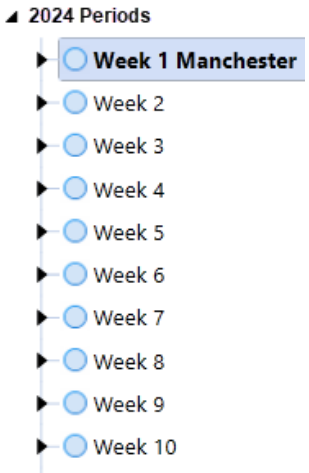
Example 1

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.

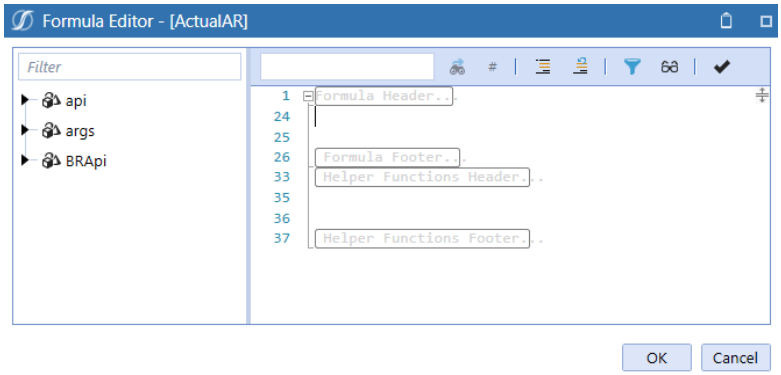
No Data Zero View For Adjustments: Determines the view when there is no data in journals for the period. Can be set to periodic or YTD and typically the setting is the same as the Input View for Adjustments.

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 About the Financial Model 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.

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.

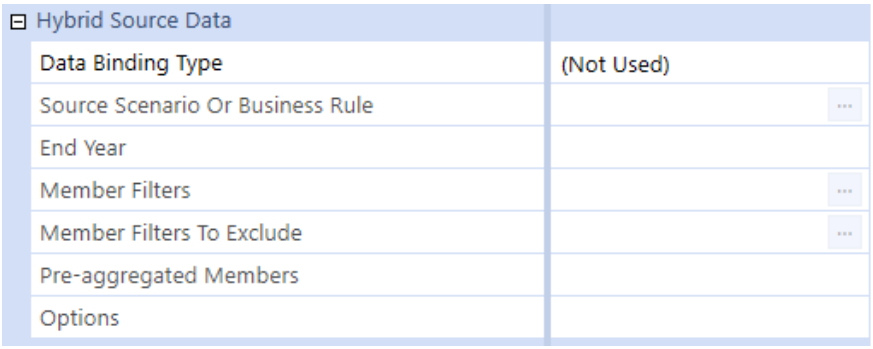
- Set to **True** to use current cube's default rate type.
- Set to **False** to use custom rate calculations.

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 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.

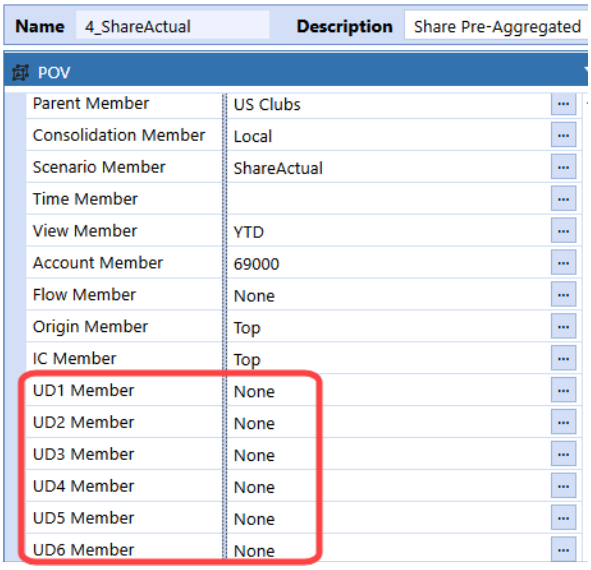
> **NOTE:** You cannot use Data Unit dimensions in the Member Filters or Member Filters to Exclude properties (Entity, Time, Consolidation, Scenario).

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.

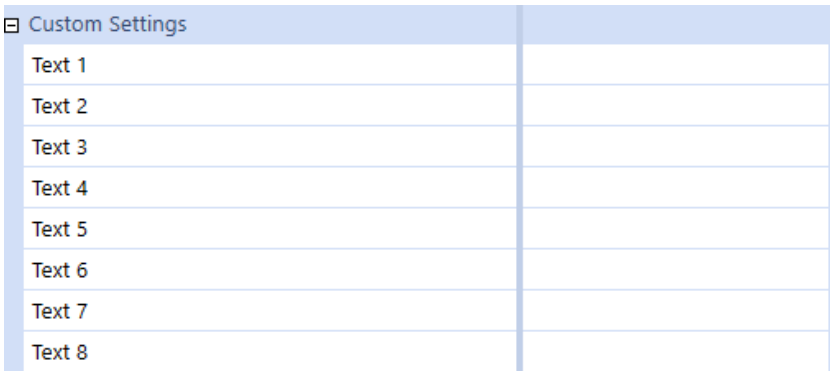
Options: Optional and variable settings that provide additional control when running a hybrid share or copy.

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. Click the **Members** tab and then the **Relationship Properties** tab.

See the following category descriptions:

- General

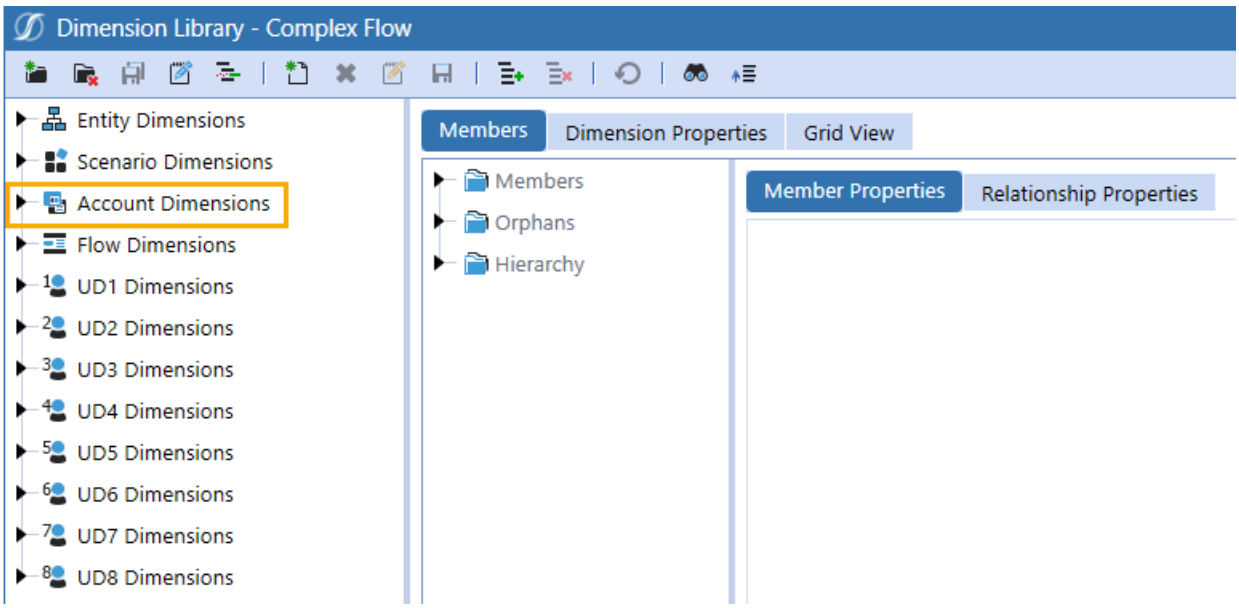
> **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

- Position Within Parent

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

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.

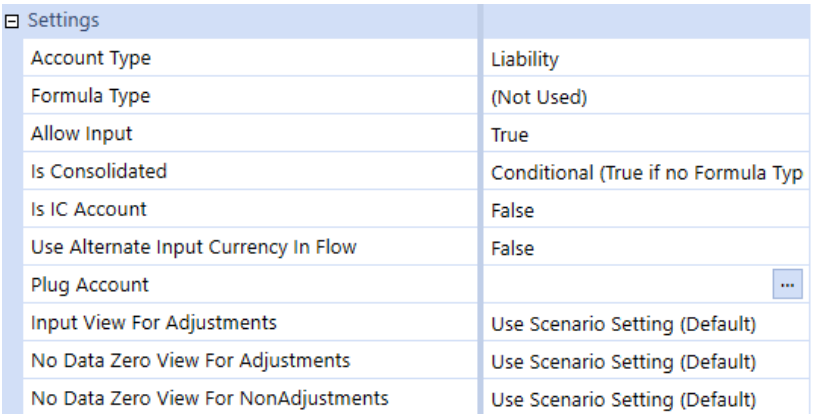
- Security
- Settings
- Aggregation
- Vary By Cube Type
- Vary By Scenario Type
- Vary By Scenario Type And Time

Security

Controls who can view the selected account dimension member.

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

Settings



Settings	
Account Type	Liability
Formula Type	(Not Used)
Allow Input	True
Is Consolidated	Conditional (True if no Formula Typ
Is IC Account	False
Use Alternate Input Currency In Flow	False
Plug Account	...
Input View For Adjustments	Use Scenario Setting (Default)
No Data Zero View For Adjustments	Use Scenario Setting (Default)
No Data Zero View For NonAdjustments	Use Scenario Setting (Default)

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 Switch Type.

- **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.

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. Click the ellipses to open the **Select Member** dialog box.
2. Click 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. Click **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

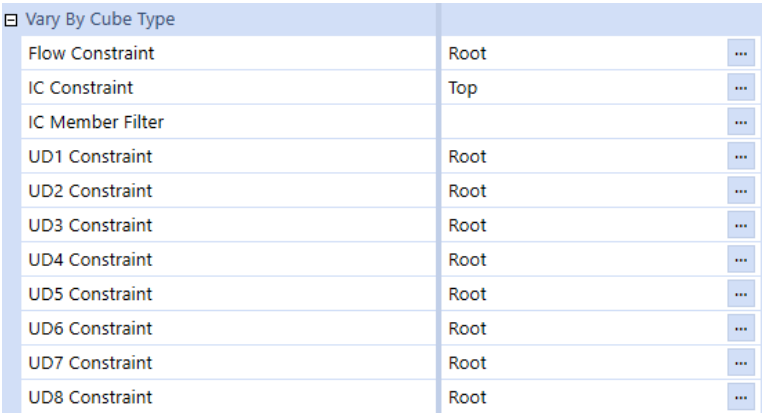
Click a cell to set each property to **True** or **False**. Set to **True** to allow aggregation of the account member in the specified dimension or **False** to turn off aggregation. Dimension members used for informational purposes are typically set to **False**.

- 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.

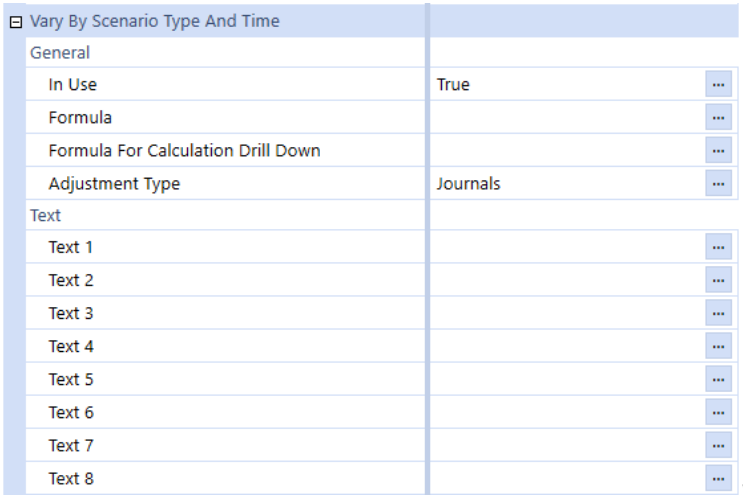
Click the ellipsis to set the **Workflow Channel** to one of the following options:

- **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.



Click 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.

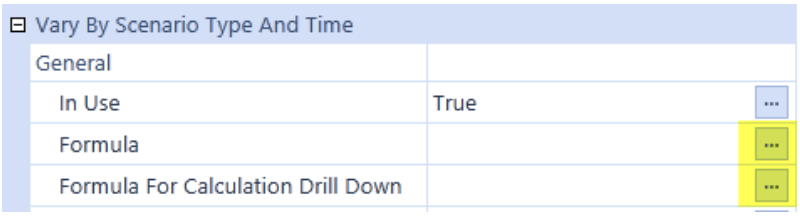
- Set to **True** (default) to have the account in use.
- Set to **False** to turn off the ability to use an account based on time.

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

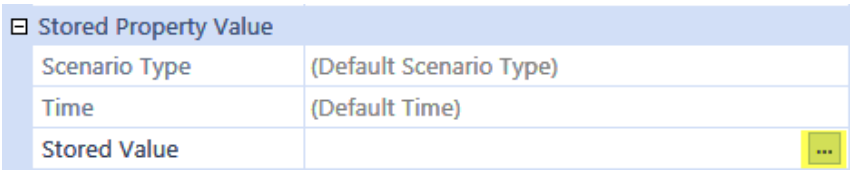
> **Example:** For instance, if a calculation changes over time, but the historical interpretation of that formula needs to be saved. For more information, see the Formula Guide, in About the Financial Model.

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**:

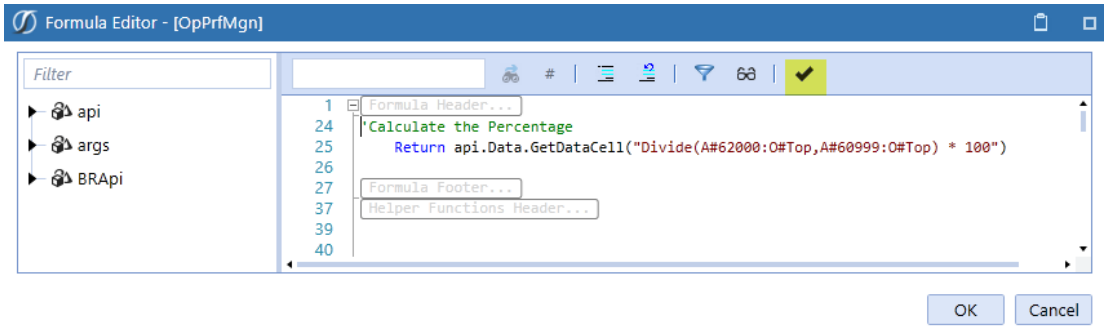
1. Click the ellipsis button located in the **Formula** or **Formula For Calculation Drill Down** fields.



2. Click 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. Click the tick icon to check that the formula was compiled correctly.
5. Click **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. Click the ellipsis to open the **Varying Member Property - Adjustment Type** dialog.
2. Click the **Stored Value** cell to generate a drop-down.
3. Choose from one of the following options:

a. **Not Allowed**: Adjustments are not allowed, at any time.

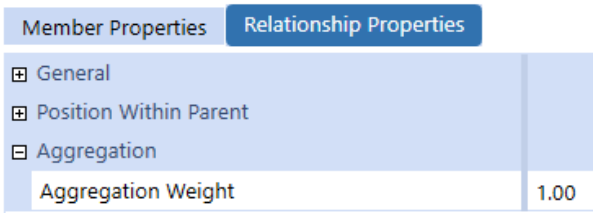
b. **Journals**: Adjustments are allowed for journal entries.

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. 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

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

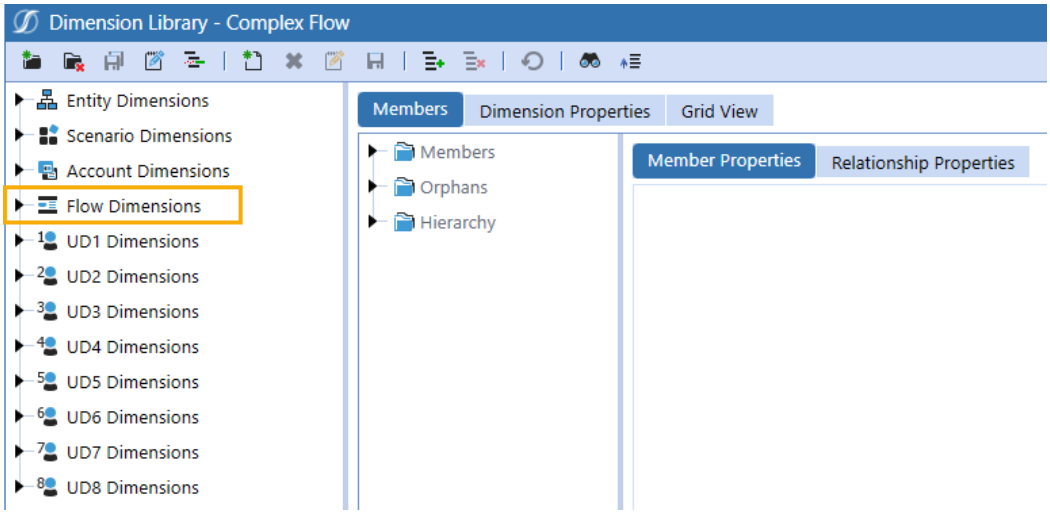
> **Example:** An aggregation weight of 0.50 means that 50% of the data stored against this member will aggregate to its parent.

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.

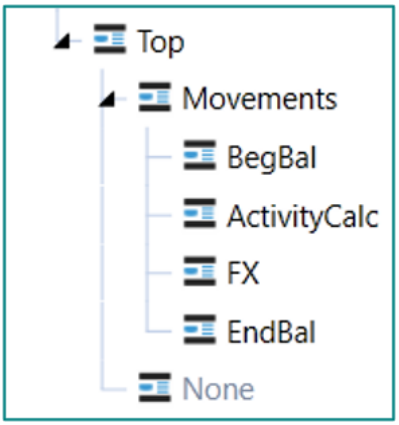
Flow Dimension

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.

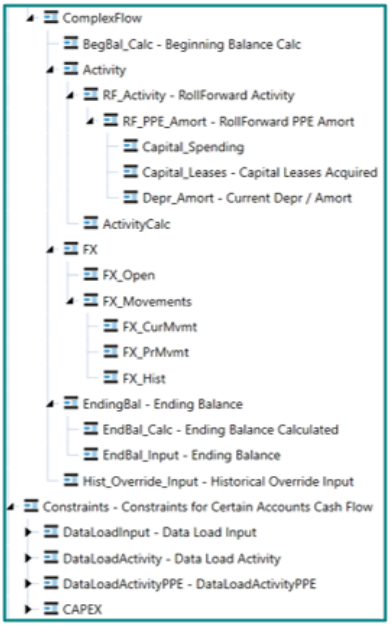


Members of this dimension aggregate up, and hierarchies can be simple or complex.

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. 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.

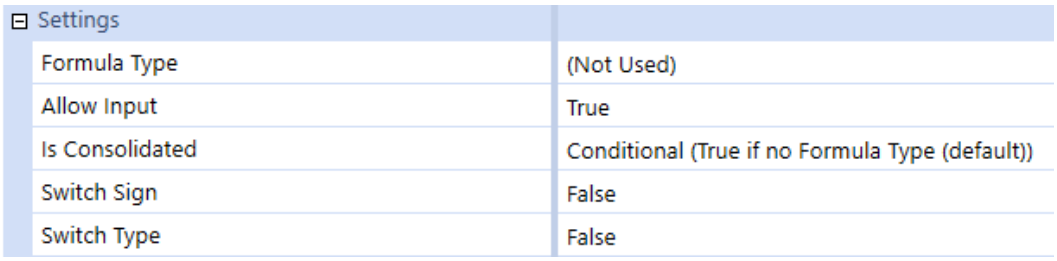
- Security
- Settings
- Flow Processing
- Vary By Scenario Type and Time

Security

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

Display Member Group: Click 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.

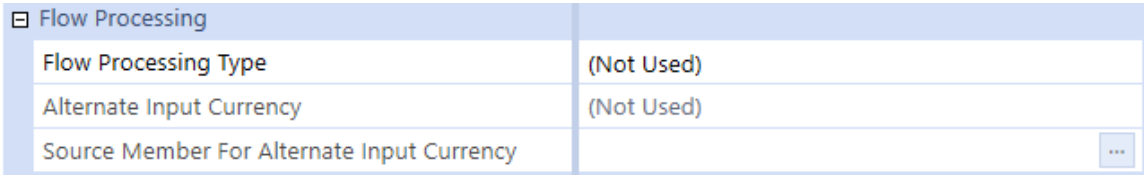
- Set to **True** to switch the sign of data for the flow member.
- Set to **False** to keep the sign as is.

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: Click within the cell to generate a drop-down list 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.

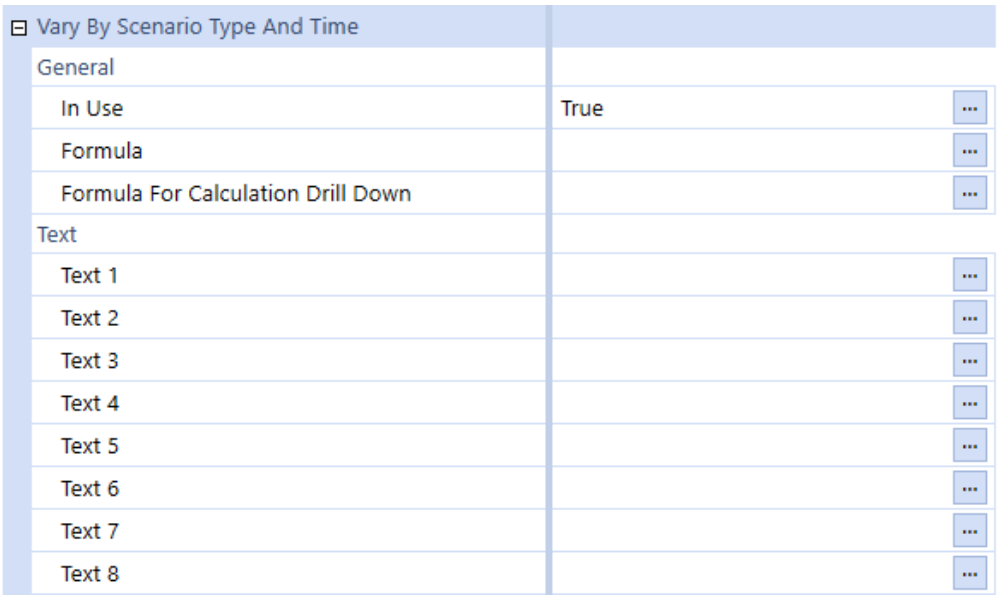
Alternate Input Currency: Lists available currencies that determine the source value override currency. This typically matches the default currency of the application.

> **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.

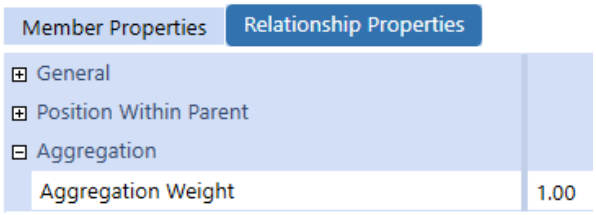
Vary by Scenario Type and Time

See Account Dimension.



Relationship Properties Tab

To view the **Relationship Properties** tab, go to **Application** > **Cube** > **Dimensions** > **Flow Dimensions** and select a flow 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

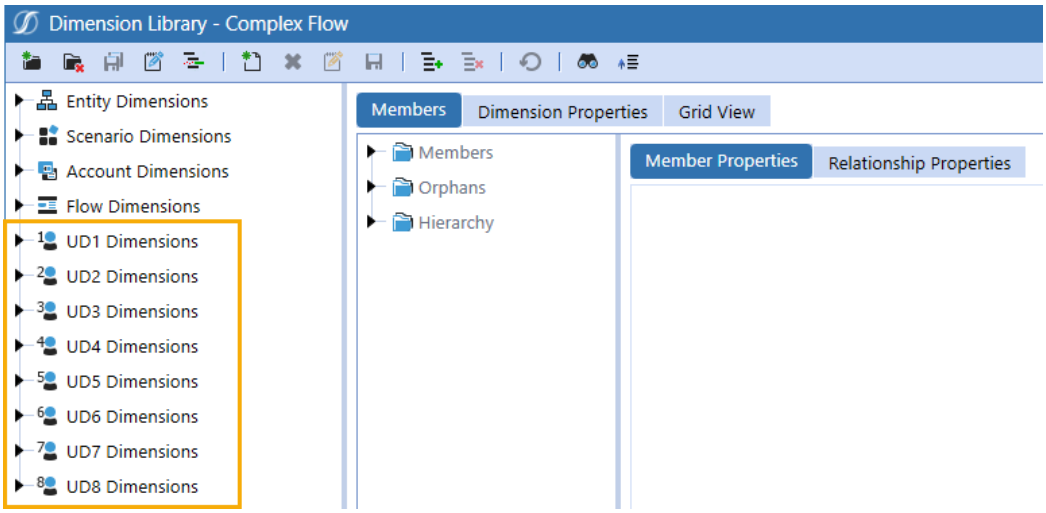
  > **NOTE:** These properties are standard across the entity, scenario, account, flow, and user defined dimensions.

Aggregation Weight

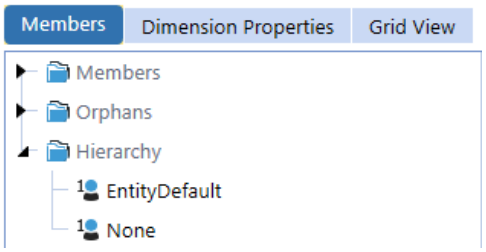
See Aggregation in Account Dimension.

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.

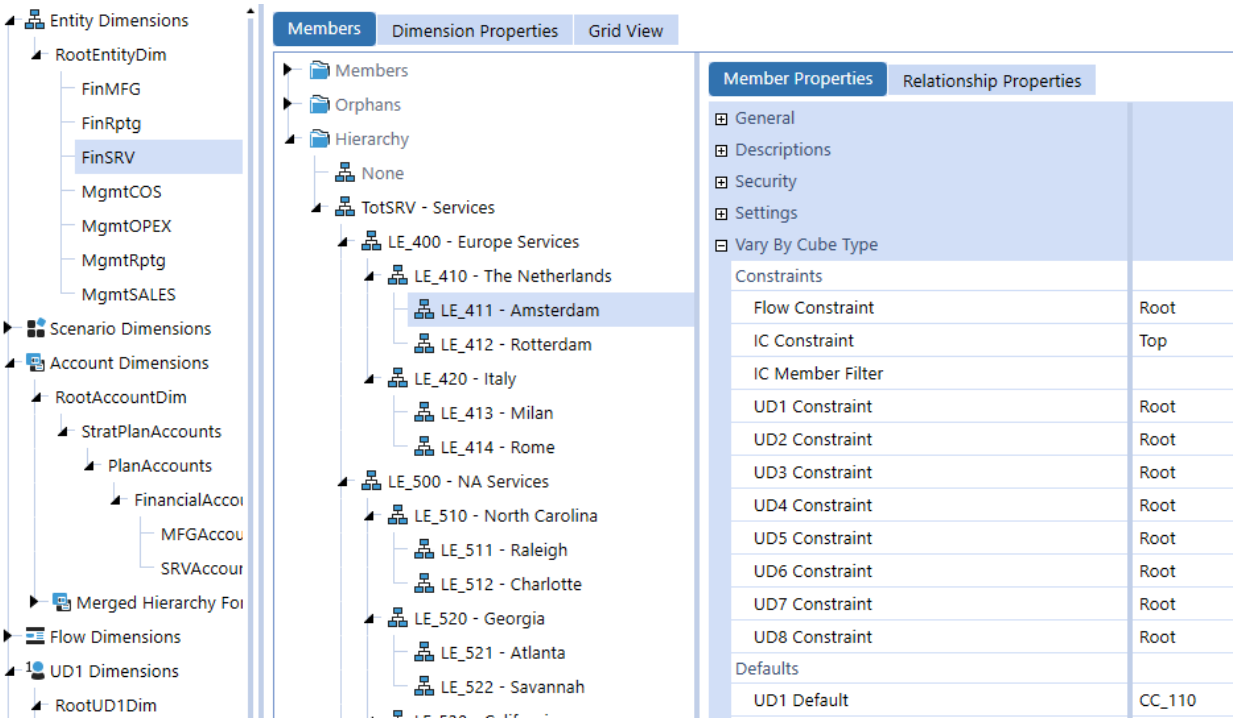


Each user defined dimension has an **EntityDefault** member that is used to assign attributes to an entity. This is set on the specific entity in the Vary By Cube Type settings for each user defined dimension. With this feature, an entity can have a specific default, which reduces the need to map every 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.

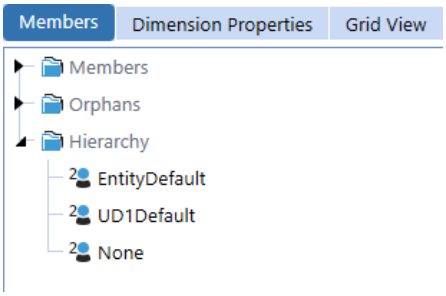


A UD1 through UD8 default membership can be defined to a user defined dimension, such as a cost center.

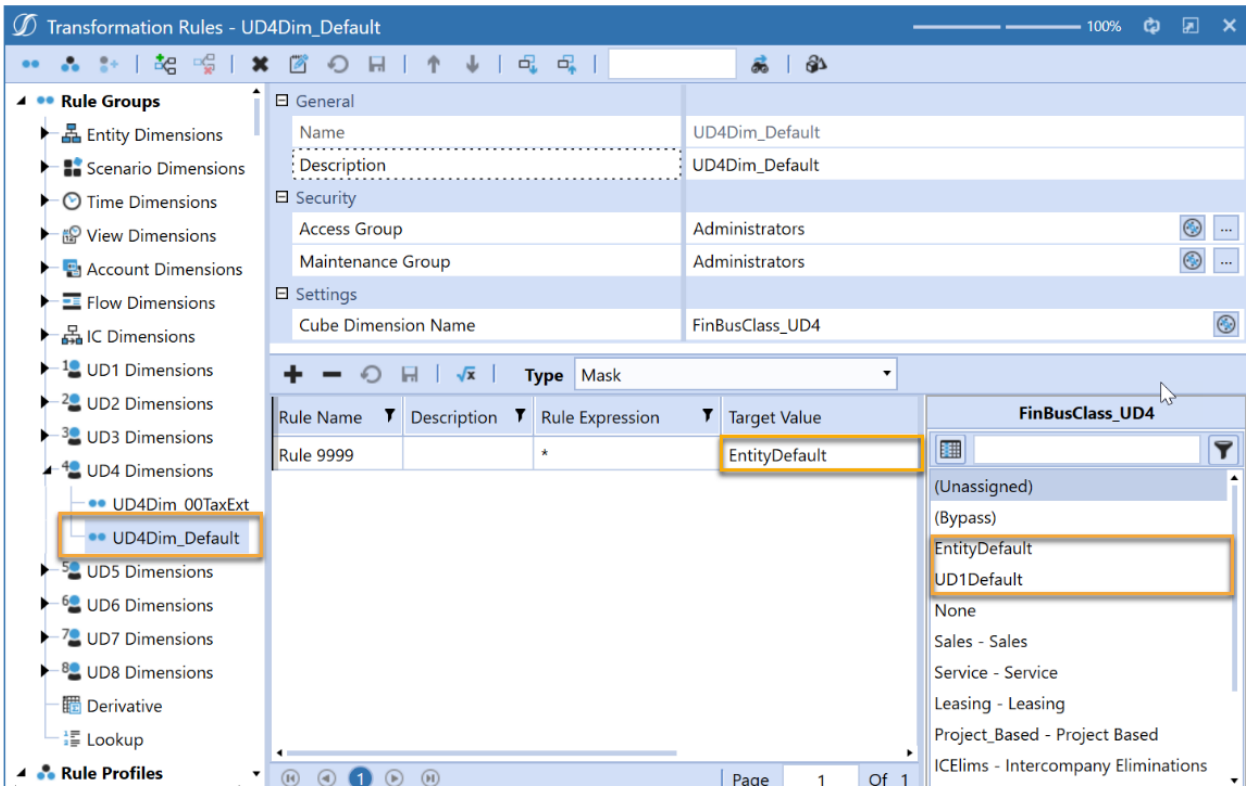
> **Example:** This entity falls in the Europe Services region and every import, form edit, and journal entry should classify this entity as being associated with the CC_110 - Sales cost center. The entity never has to explicitly map to the cost center. When data is loaded, it is directed to the EntityDefault 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 based on the settings for the related reference properties. Data intersections are not loaded directly to an attribute. 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. Being stored members, the entity and scenario, as related dimension types, do not support calc-on-the-fly. Attribute member results on entity or scenario will only be available on base members.

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.
- 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.

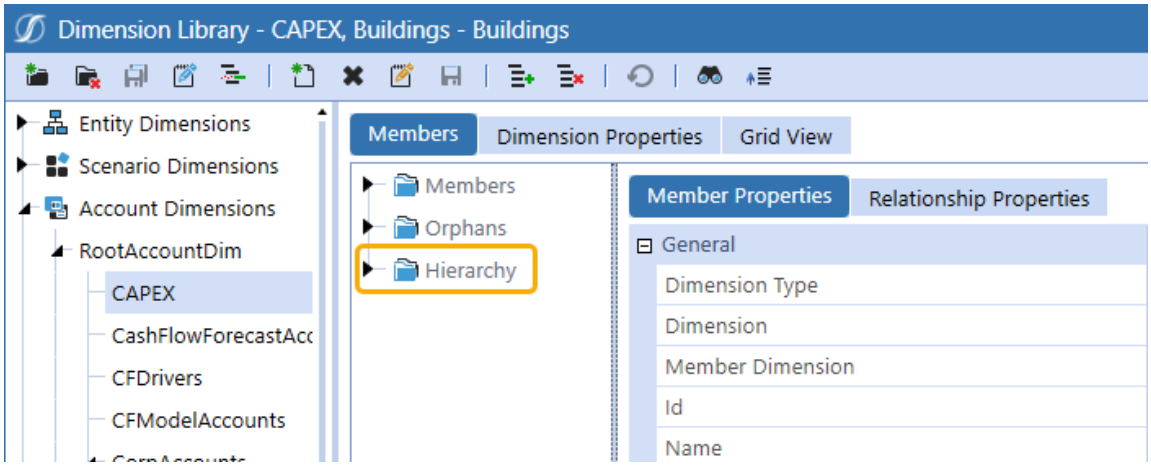
```
Dim objDataBuffer As DataBuffer =
   api.Data.GetDataBuffer(scriptMethodType,
    sourceDataBufferScript,
changeIdsToCommonIfNotUsingAll, includeUDAttributeMembersWhenUsingAll, expressionDestinationInfo)
```

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.

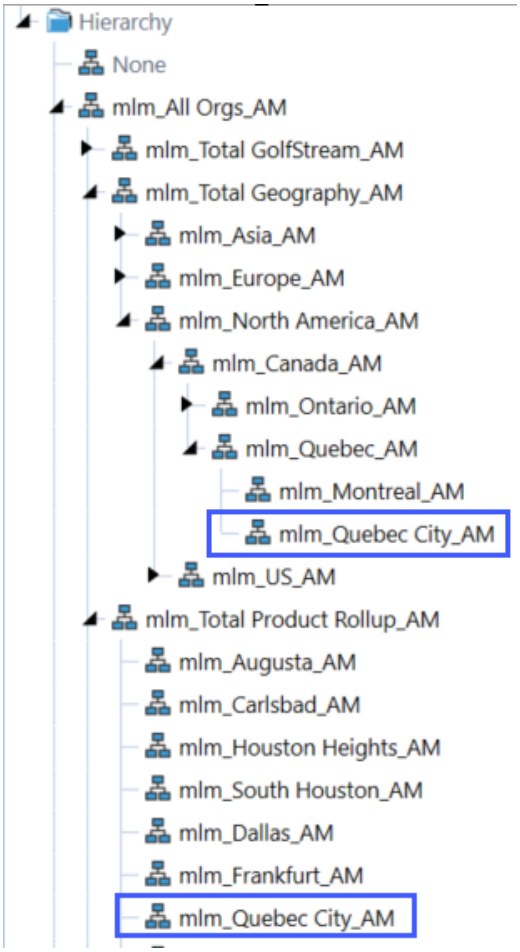
The parent dimension member is defined by the relationships within an entity dimension member hierarchy. To view the hierarchy, go to **Application** > **Cube** > **Dimensions** and select a dimension member. Click the **Members** tab and then click the arrow next to **Hierarchy** to expand the member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

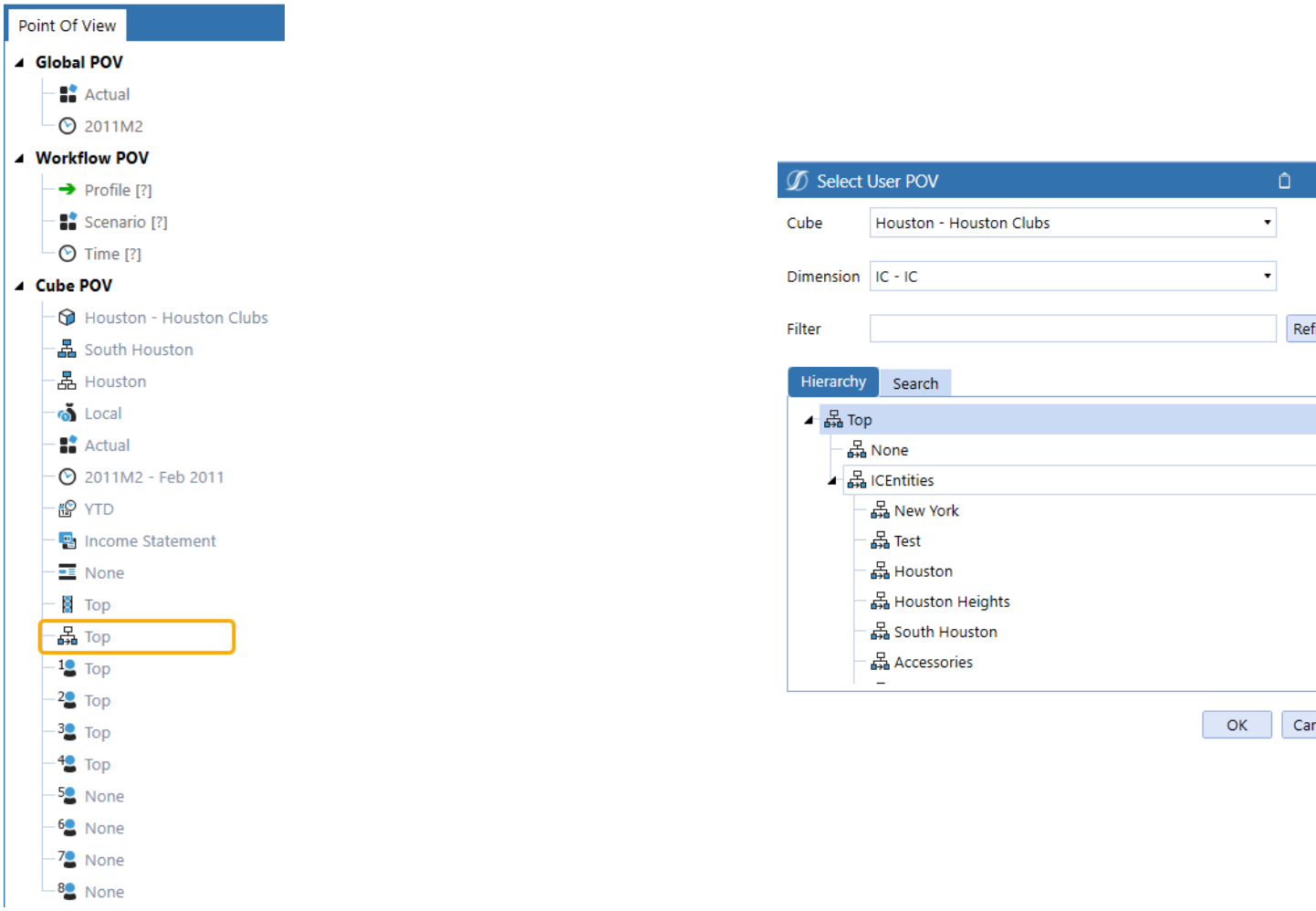
Example: Below, mlm_Quebec City_AM has two different parent-level entity members, as it is both a child member of mlm_Quebec_AM and mlm_Total Product Rollup_AM.



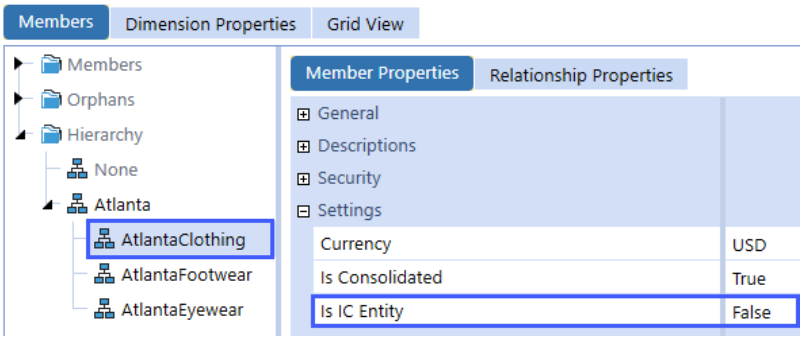
NOTE: The parent dimension does not have a hierarchy of its own as it varies based on the dimension and member selected.

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 in About the Financial Model.

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.

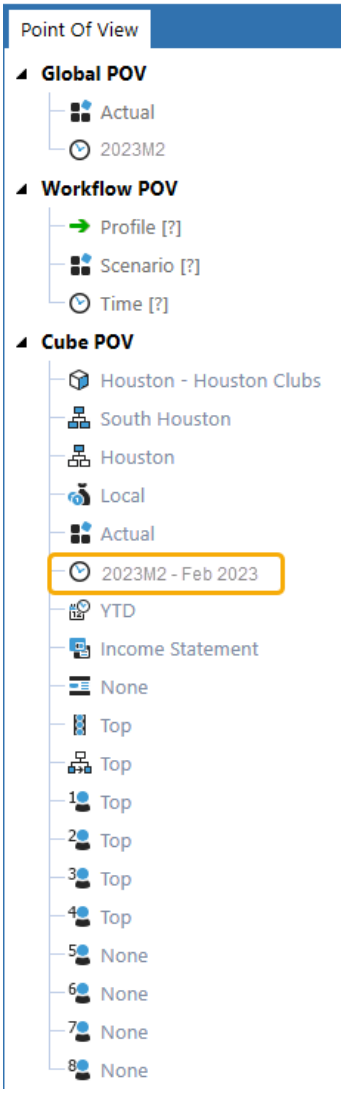
> **Example:** The 445 calendar divides a year into four quarters of 13 weeks, each grouped into two 4-week months and one 5-week month. The longer month may be set as the first (5-4-4), second (4-5-4), or third (4-4-5) unit. This drives the time hierarchy that displays in the point of view.

> **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.

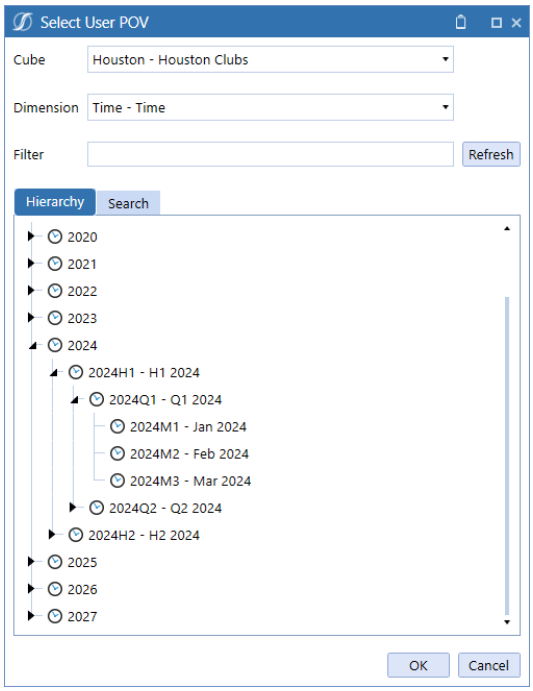
Time Dimension Hierarchy

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

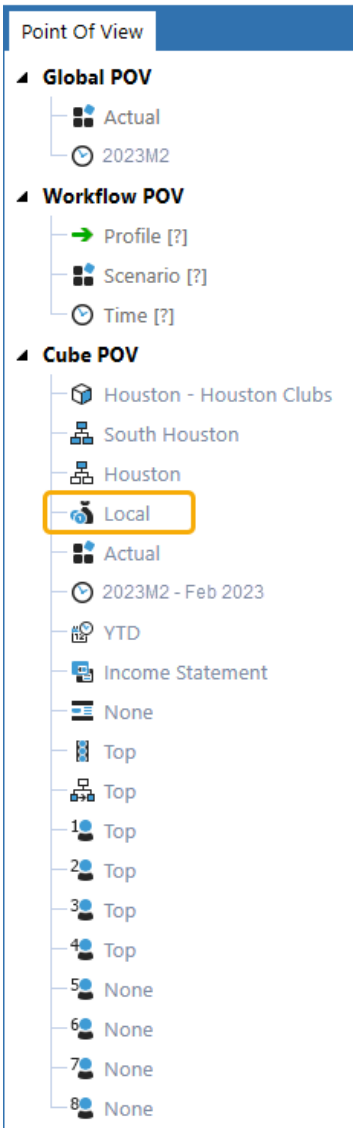


Weekly Example

> **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

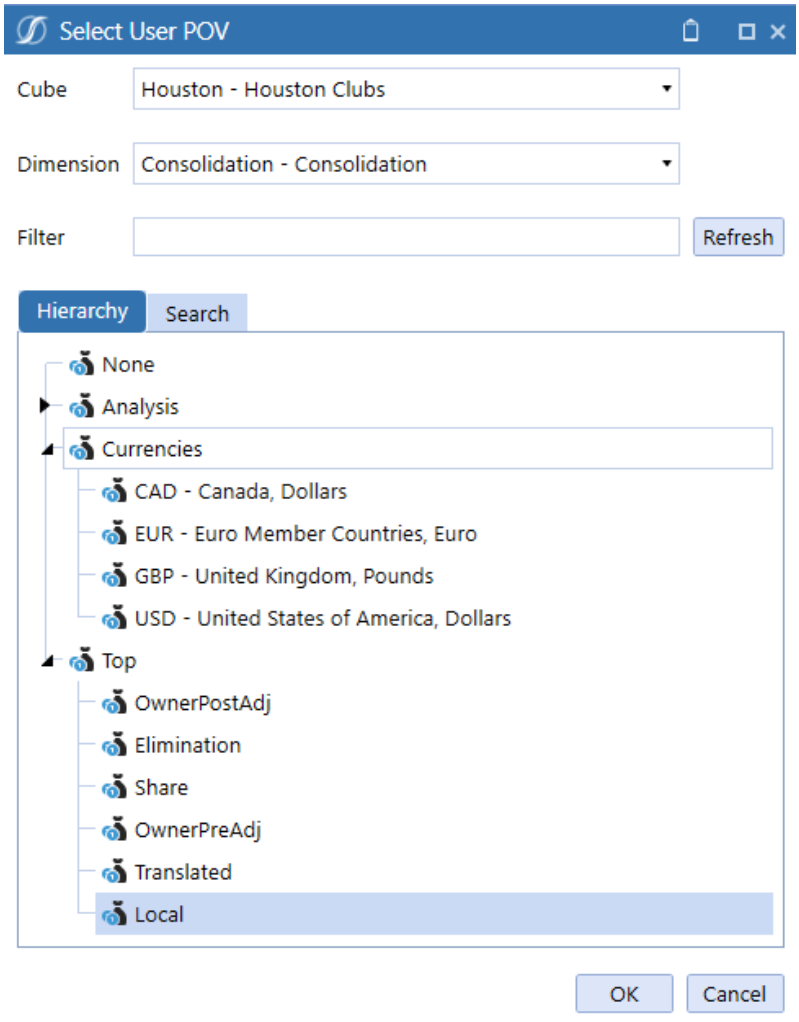
The consolidation dimension member hierarchy shows how entity data rolls up from the local currency to the final numbers that contribute to its parent entity member. This dimension is not editable through the Dimension Library but is viewable using the Point of View pane.



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 Consolidation in About the Financial Model.

System-defined Dimensions

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

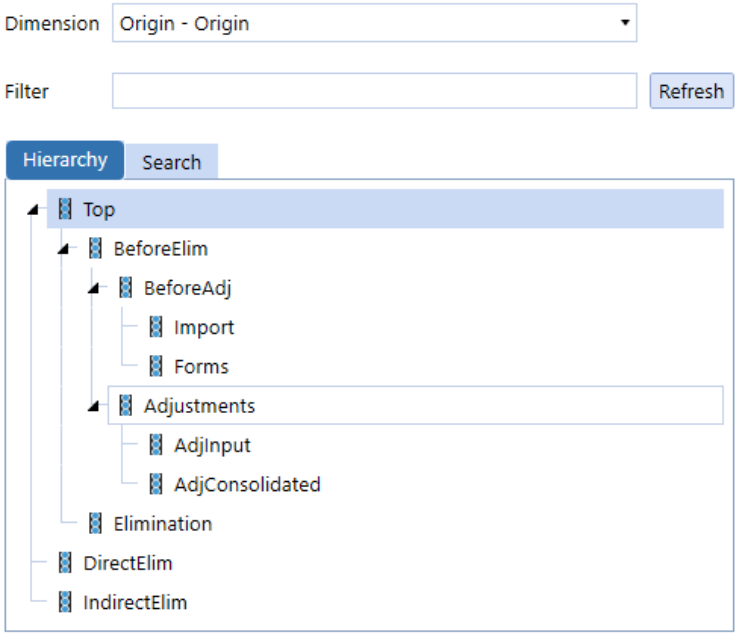
Origin Dimension

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 in About the Financial Model.

Origin Dimension Member Hierarchy

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

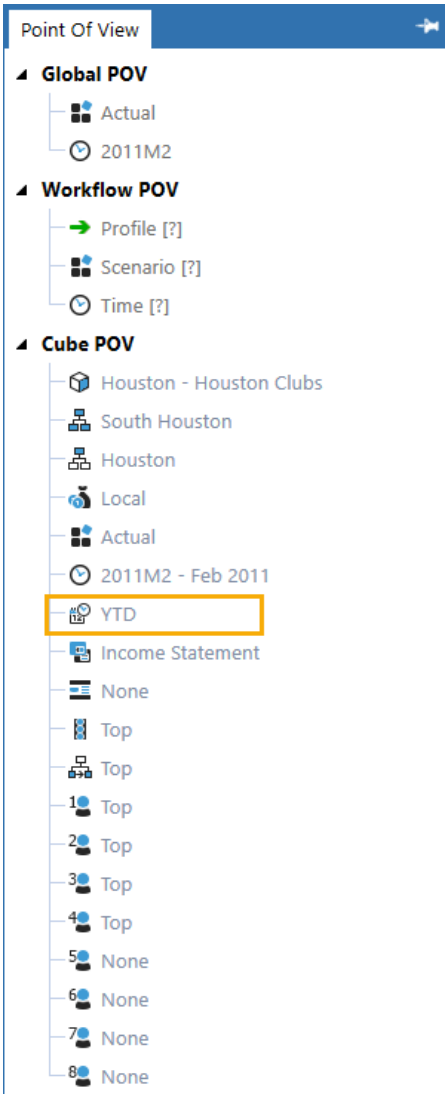


- **Top**: Shows data consolidated after eliminations.
- **BeforeElim**: Shows data consolidated before eliminations.
- **BeforeAdj**: Shows data before adjustments.
- **Import**: Populated through the data loaded during an import workflow, which enables the application to bring in data from an external 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

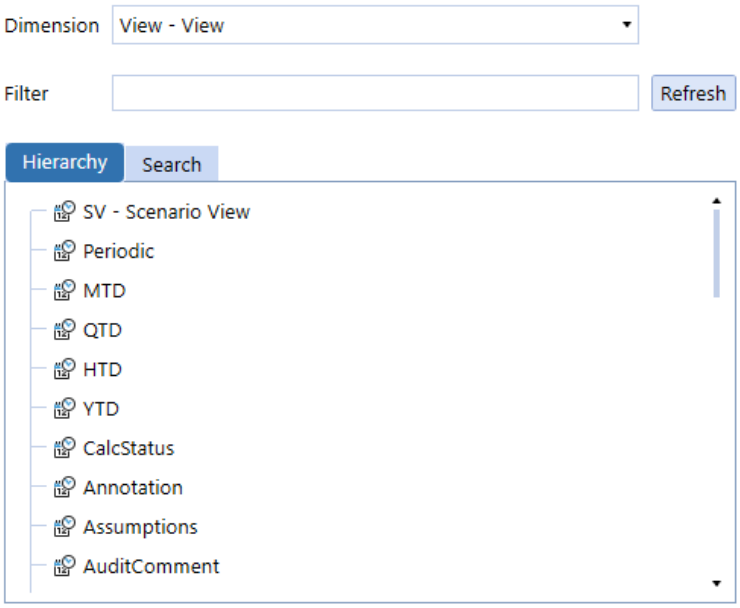
The view dimension is a system-defined dimension and is not accessible in the dimension library. Access the view the dimension using the POV pane.



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.



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.

For 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 buttons to support Time Profile functions.

1. **Create Time Dimension Profile** - Use this button to create a new Time Profile. When a new Time Profile is needed, click on this icon to start the configuration process for a new Time Profile.

2. **Delete Selected Time Dimension Profile** - Use this button to delete a selected Time Profile. The intention of this button is 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 button will 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 button to 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 button 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 button when 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 button after changing any properties in an existing Time Profile.

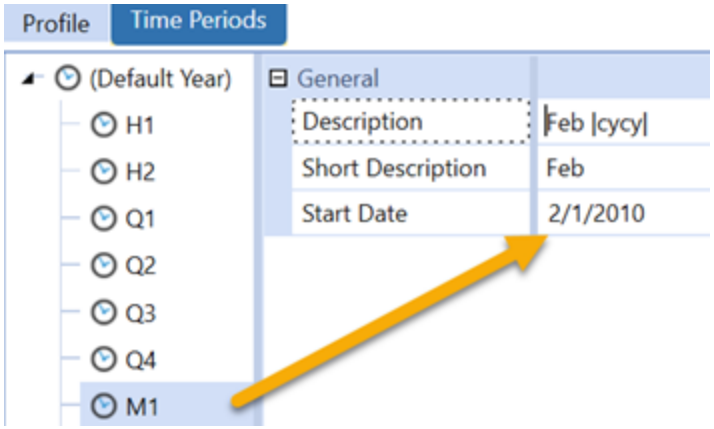
Profile

- **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. Additionally, 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 Calendar 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.

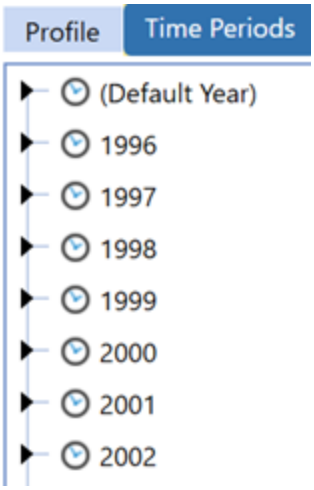
- **Fixed Weeks 4-5-4**: A weekly calendar determined by four weeks in first month, five weeks in second month, and four weeks in third month within a Quarter.
- **Fixed Weeks 5-4-4**: A weekly calendar determined by five weeks in first month, four weeks in second month, and four weeks in third month within a Quarter.

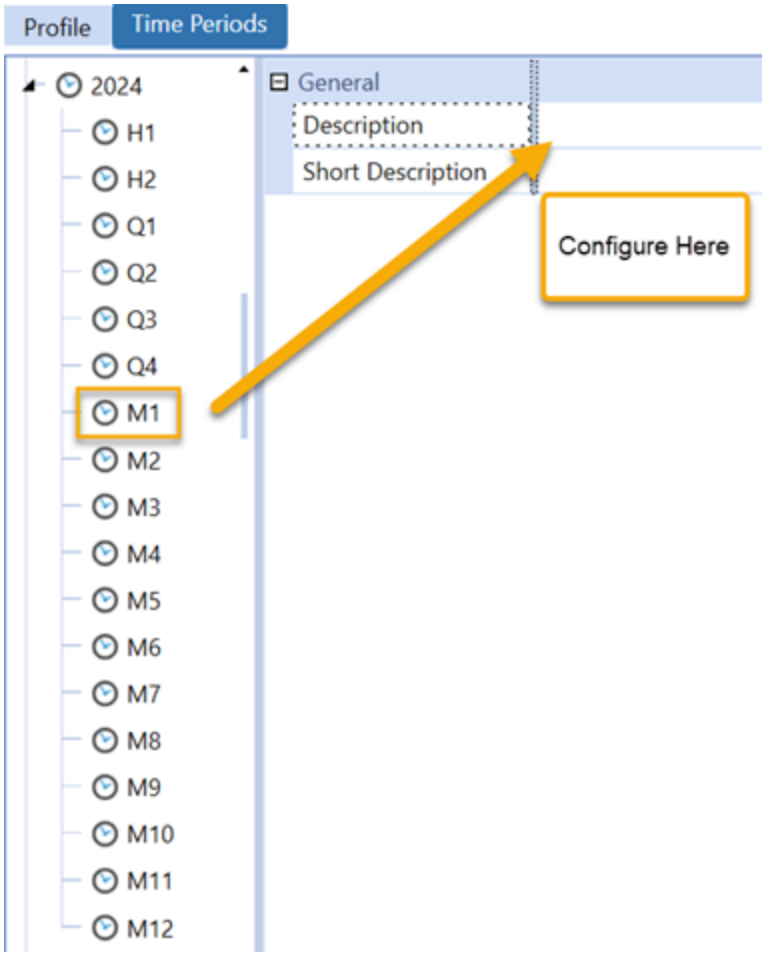
> **NOTE:** For Fixed Weeks, each month is either 28 or 35 days except for the twelfth month which will have an extra one or two days.

- **Custom Start Dates**: This option allows for 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.


Profile
Time Periods
(Default Year)
1996
1997
1998
1999
2000
2001
2002



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

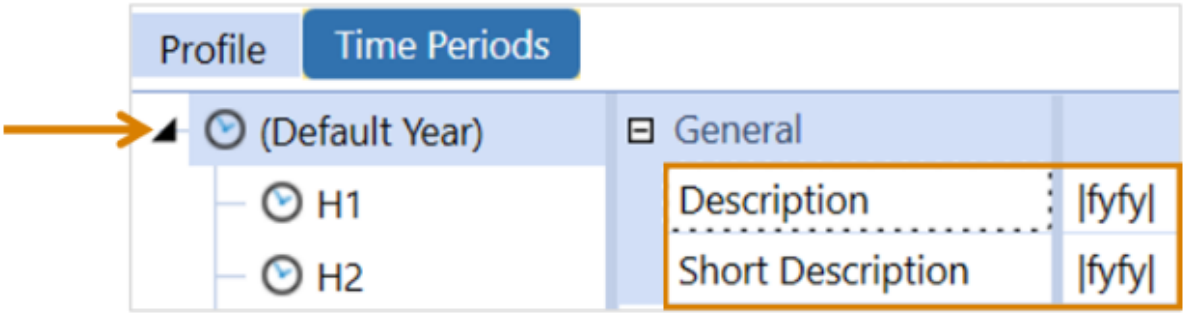
The Time Periods tab is used to align the Time Periods, which are defined by the Time Dimension configured at application creation time, with the Time Profile. Each Time Profile can be unique based on different Fiscal Year accounting periods. However, the Time Periods are defined at the application creation level. Therefore, different Fiscal Year Time Profiles would share the same Time Periods configuration. See Time Dimensions for more information.

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

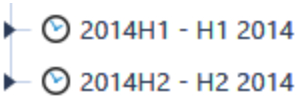
Year Description: Displays the Year as either Fiscal Year or Calendar and defaults as |fyfy|.

Short Description: Displays the Year as either Fiscal Year or Calendar and defaults as |fyfy|.

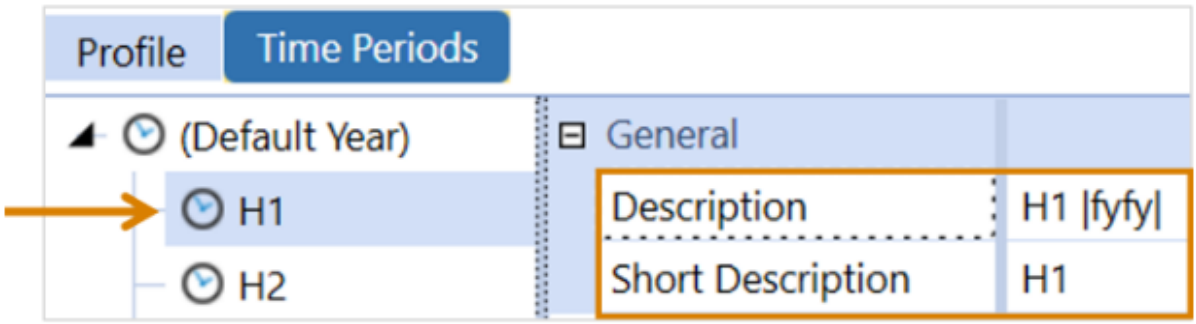


Half Years

HY1, 2 Description; Displays the Half Year as either Fiscal Year or Calendar Year and defaults as H1 |fyfy|.

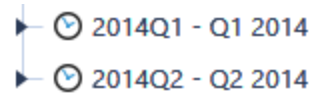


HY1, 2 Short Description: Displays the Half Year's short description and defaults as H1.

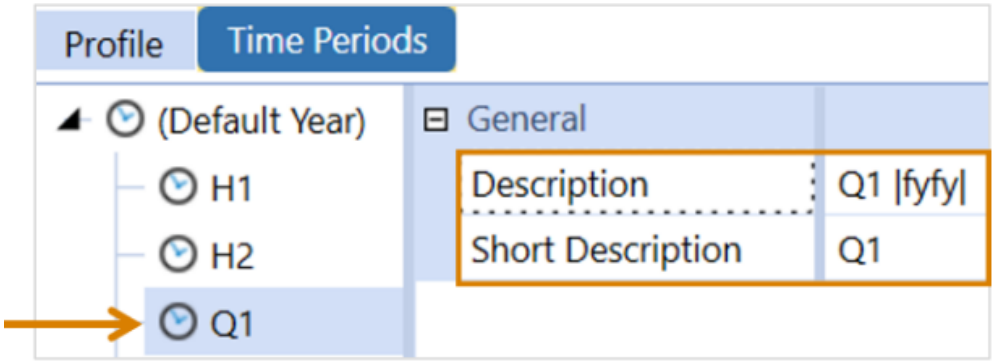


Quarters

Q1, 2, 3, 4 Description: Displays the Quarters as either Fiscal Year or Calendar Year and defaults as Q1 |fyfy|.



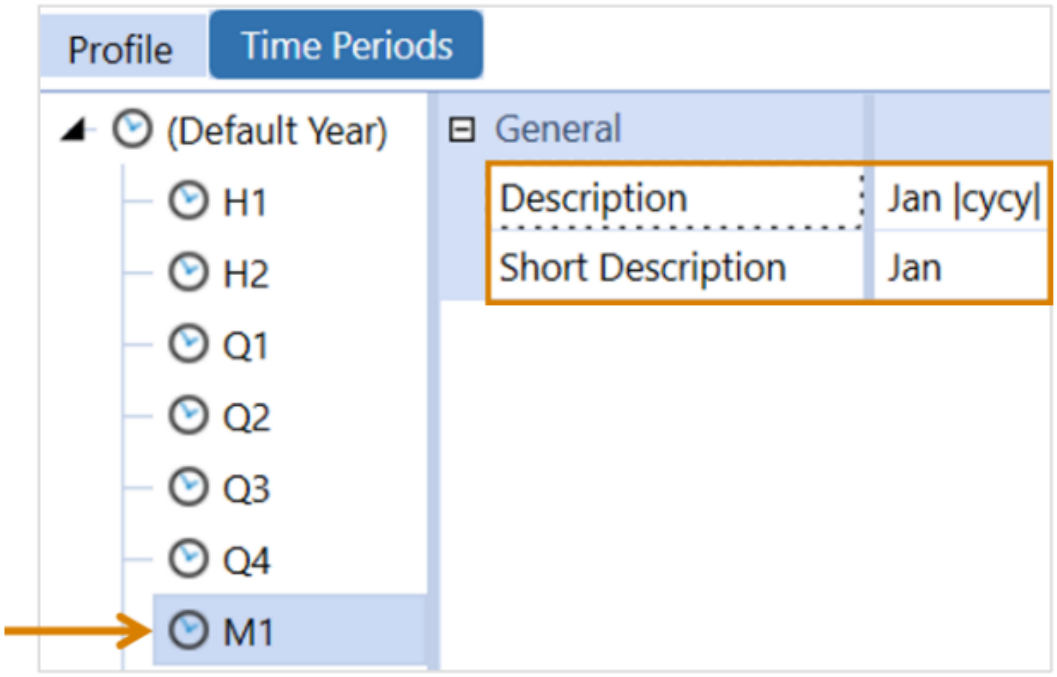
Q1, 2, 3, 4 Short Description: Displays the Quarters short description and defaults as Q1, Q2, Q3, Q4.



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

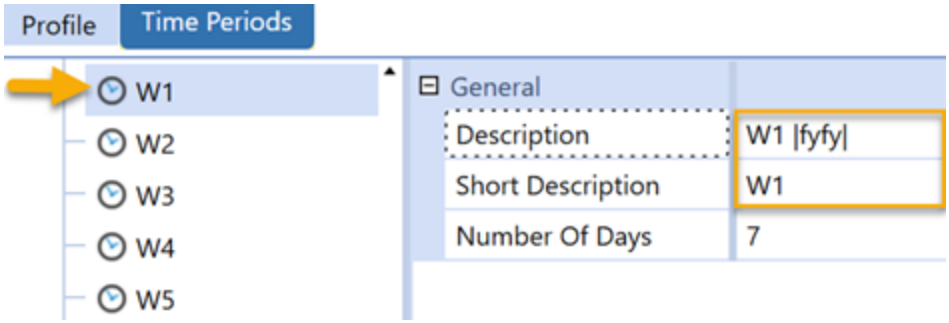
Weeks are available if the application is created to accommodate Weeks. Weeks are configured differently to Half, Quarter, and Months. Not only do the Weeks have Description and Short Description, but Weeks also contain Number of Days for configuration. If the application contains Weeks, here is how the Weeks will be displayed. See Time Dimensions and 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 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 for it to store, calculate, translate, and consolidate data.

It is flexible, designed for specific purposes, and data types in an 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 setup and controlled within the Cube profile.

Structure of a Cube

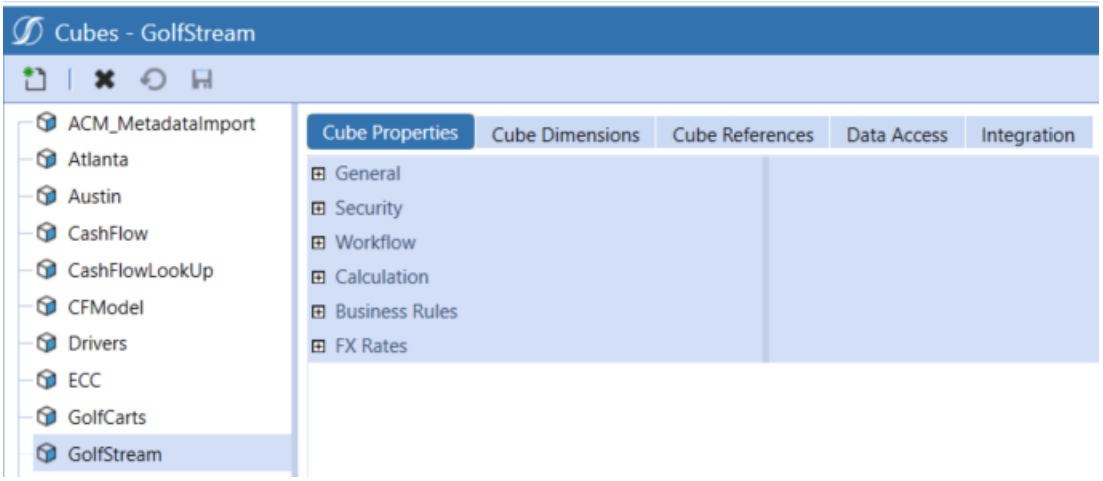
A Cube consists of multiple tabs. Each tab plays a key role supporting the structure and processing:

- Cube Properties
- Cube Dimensions
- Cube References
- Data Access
- Integration

See Dimensions Overviewon Cubes and Extensible Dimensionality.

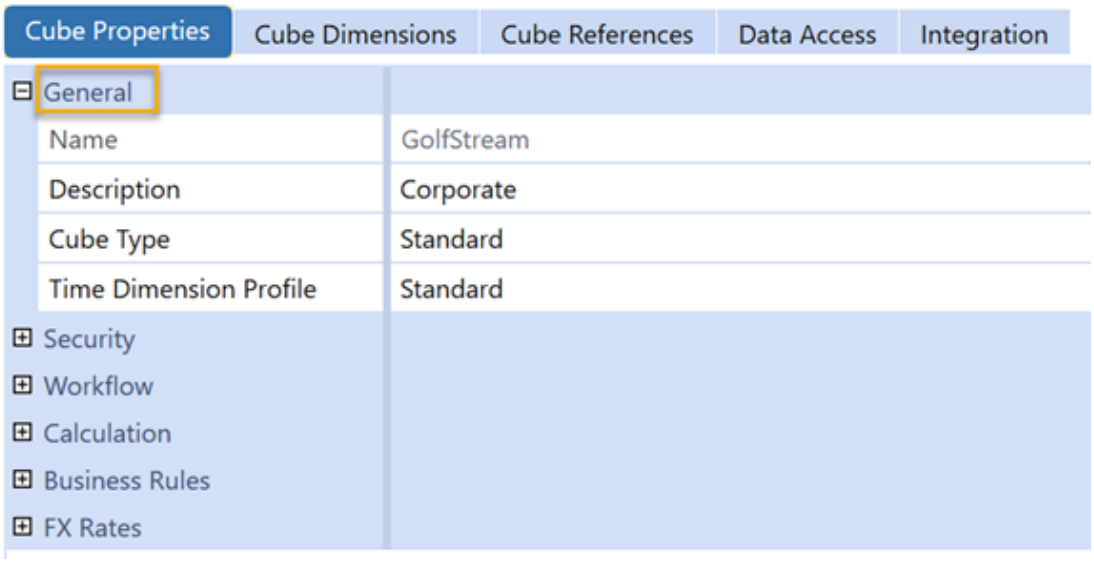
Cube Properties

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



General

Inside the Cube Properties tab is a section called General.



Name: Input 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: Input field to provide a description for the Cube and limited to 200 characters. The Cube Description can be changed at any time being created.

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 properties:

- **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

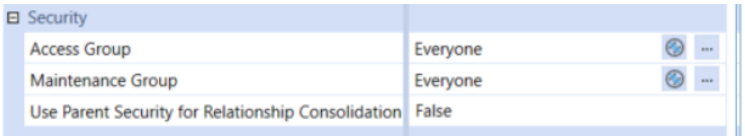
- Supplemental = 3
- What If = 4
- Cube Type 1 = 11
- Cube Type 2 = 12
- Cube Type 3 = 13
- Cube Type 4 = 14
- Cube Type 5 = 15
- Cube Type 6 = 16
- Cube Type 7 = 17
- Cube Type 8 = 18

Time Dimension Profile: This property provides a drop-down list of 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 than 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. See Security.

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. See Cube Design Options and Characteristics.

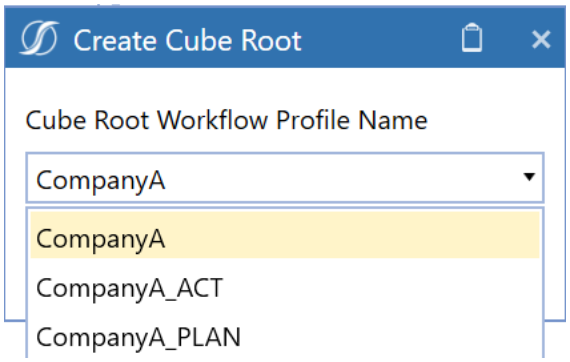
Suffix for Varying Workflow by Scenario Type: This property is used to group Scenarios by Scenario Types to make available to users in the Workflow. This can be determined on a Cube-by-Cube basis. This is active only for Cubes with Top Level Cube for Workflow set to True. There is one setting for each 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.

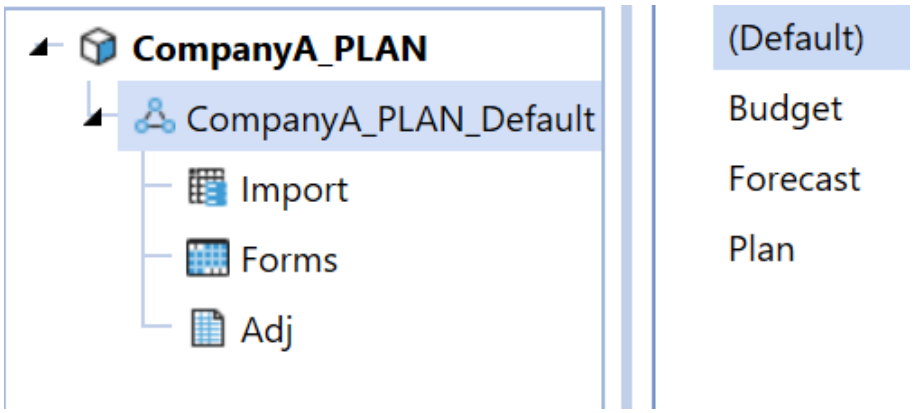
While technically feasible, this design may not offer optimal efficiency for organizing Workflow Profiles and Workflow Profile hierarchy structures. For example, different users are responsible for different processes within an organization. Accounting/Finance is responsible for Financial Close & Consolidation, 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.

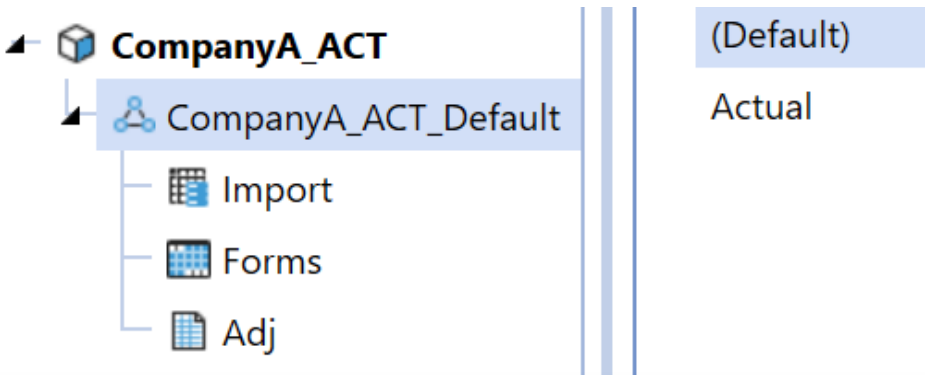
Based on the configuration on the Cube, the Cube Root Workflow Profile Names would show accordingly when creating a Cube Root:



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 for more information.

> **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 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 using OneStream'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 from OneStream'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 Algorithm 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's 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.

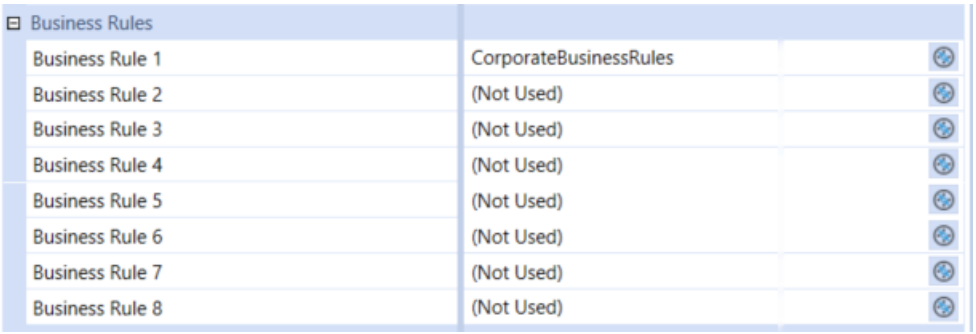
Standard Using Business Rules for FX Rates: Standard Using Business Rules for FX Rates is like Standard but allows the ability to use a Business Rule to specify translation rates for any given intersection. Any intersections not specified in the Finance Business Rule will translate based on the Standard 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, we can dynamically determine what year needs to translate 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, see Consolidation.

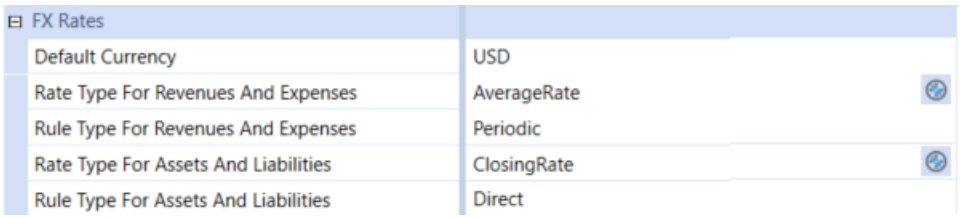


Business Rules	
Business Rule 1	CorporateBusinessRules
Business Rule 2	(Not Used)
Business Rule 3	(Not Used)
Business Rule 4	(Not Used)
Business Rule 5	(Not Used)
Business Rule 6	(Not Used)
Business Rule 7	(Not Used)
Business Rule 8	(Not Used)

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 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 Average Rate, Opening Rate, Closing Rate, and Historical Rate. Although a pre-defined selection 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 FX 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

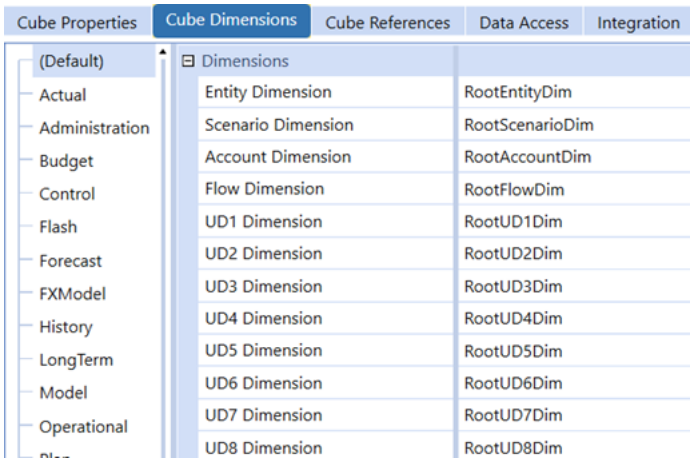
Dimensions are assigned to a cube through the Cube Dimensions tab. The dimension metadata provides structure to the cube and dictates how the cube will organize the data within the application. When assigning dimensions to a cube, it is best practice to assign the cube dimensions, except for the Entity and Scenario dimension types, by Scenario Type. Additionally, there is an option called (Default) that included with all the Scenario Types.

All dimension types vary by Scenario Type except for Entity and Scenario, which must be assigned at (Default). Since the Entity and Scenario dimension types do not adhere to the same concept of Extensible Dimensionality as the Account related dimension types, they cannot be assigned at the Scenario Type levels. When creating a new Cube, the Entity and Scenario dimension types are grayed out at the Scenario Type levels by default and cannot be changed.

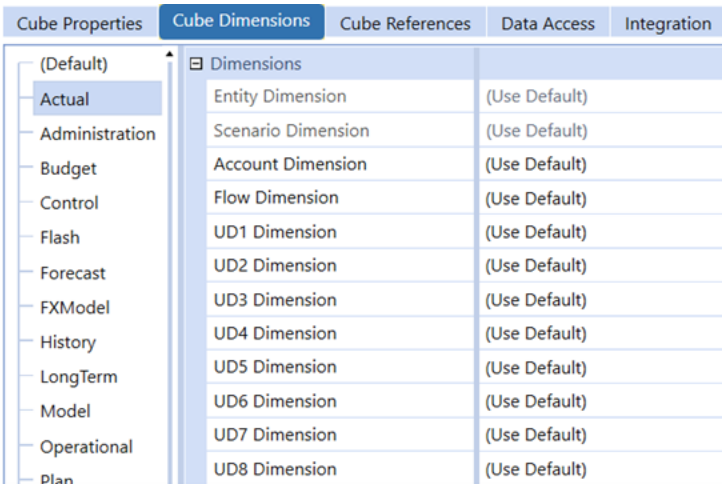
When assigning cube dimensions to the cube, managing unused dimension types per Scenario Type. As dimension types are configured with appropriate dimensions, any unused dimension types at the individual Scenario Type levels will have (Use Default) populated. Instead of using (Use Default) as the option, change (Use Default) to RootXXXDim for the Dimension Types.

> **NOTE:** XXX represents the name of any of the Dimension Types.

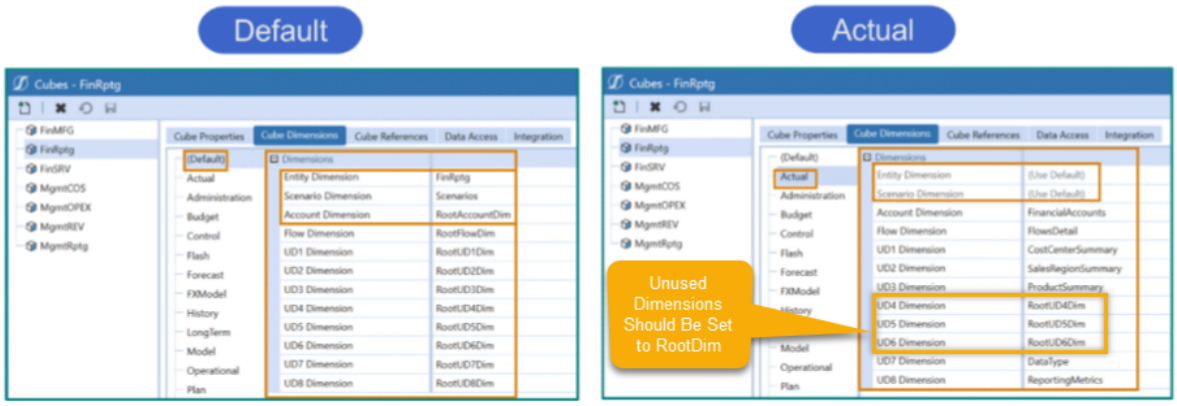
When a new cube is created, the option (Default) defaults to the configuration below for Cube Dimensions. At (Default), only the Entity and Scenario Dimensions will be configured and all other dimensions remains at RootDim.



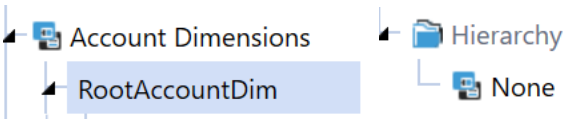
For actual and each Scenario Type, the Entity and Scenario dimension types are grayed out and not configurable. All other dimension types will be set to (Use Default). As dimensions are assigned to dimension types at the Scenario Type level, any dimension types that are not defined need to be set to the appropriate RootXXXDim.



Example: The example below shows the proper cube dimension configuration at (Default) and the Actual Scenario Type. At (Default), the Entity and Scenario Dimension Types are assigned. All other Dimension Types are set 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.



Unused Dimension Types are set to RootXXXDim at the Scenario Type level . Each Dimension Type contains a RootXXXDim. Inside of the 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. By the nature of the relationship, the new Dimension inherits RootXXXDim and the none member.

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. This is a crucial configuration concept as it allows flexibility if the Dimension Type needs to be used in the future.

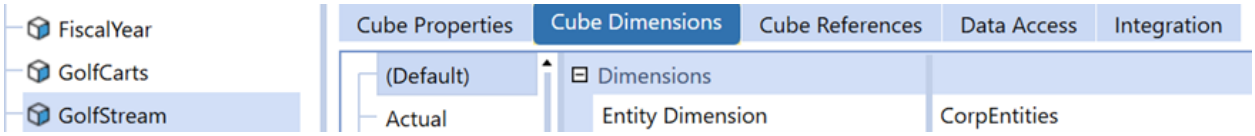
If RootXXXDim is assigned to the Dimension Type and data is loaded to the Cube, all the data for that Dimension will be in the Dimension's 'None' member, which is inherited from RootXXXDim. If a new Dimension is created for the Scenario Type in the future, the Dimension can be changed from RootXXXDim 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 already 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. See About the Financial Model for more information.

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 for more information.

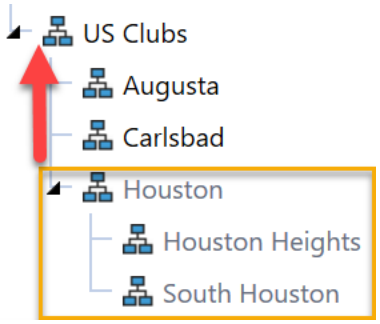
Cube Reference

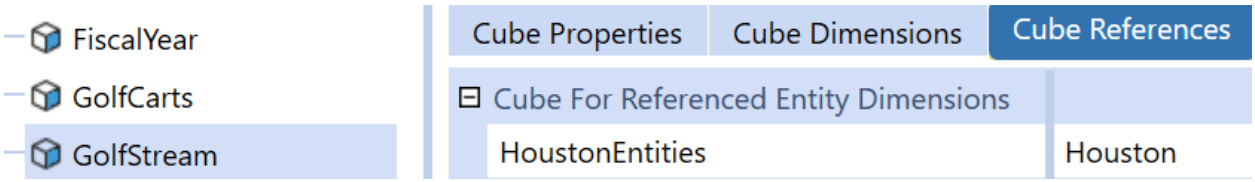
The Cube References tab is used to bring Linked Cubes together. By using Cube References, one Cube is defined as a parent Cube. The parent Cube will have the Cube References configured to include all referenced Entity Dimensions from child Cubes. When a Cube is created, the Cube Reference tab is blank. 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 for more information.

> **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.

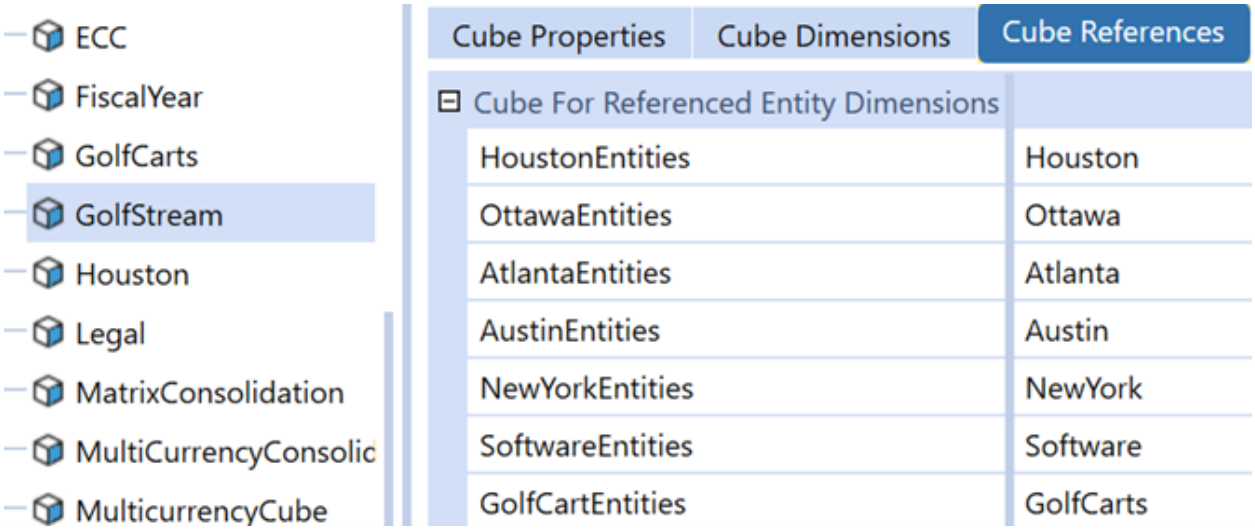


When an Entity hierarchy from another Entity Dimension, such as HoustonEntities, has a relationship with an Entity in the CorpEntities Entity Dimension, HoustonEntities will appear in the Cube Reference tab under Cube for Referenced Entity Dimensions for the Golfstream Cube.





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 on Cube Properties > Security section, 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

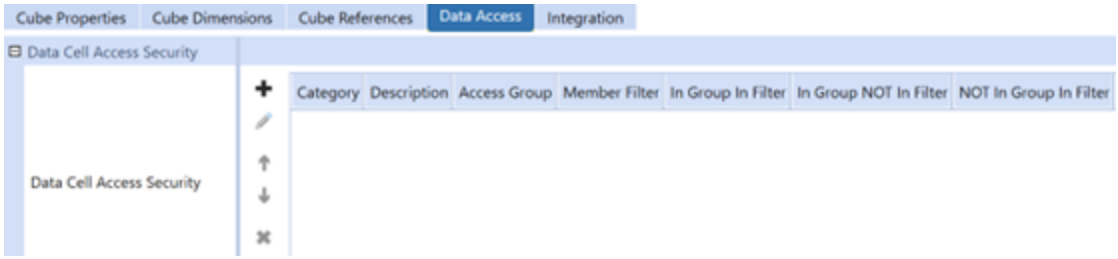
Data Cell Access Security

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.

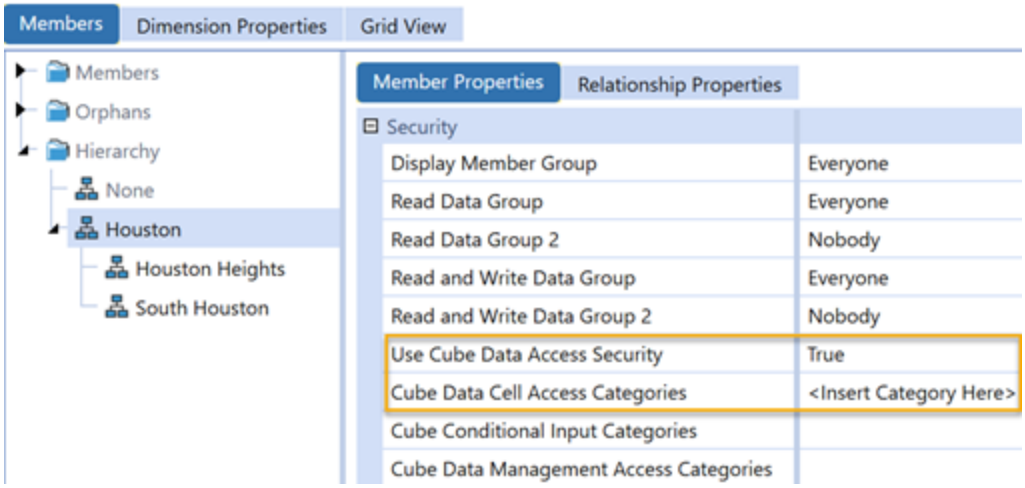
To get access to Entity data, you must have Read/Read Write access to the Entity through Entity security. See Entity Security for more information. When granted access to the Entity data, you get access to every single data cell for the Entity within the Cube. If there is a situation where the user should 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.

First, 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. See Security for more information.

- **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. See Security for more information.

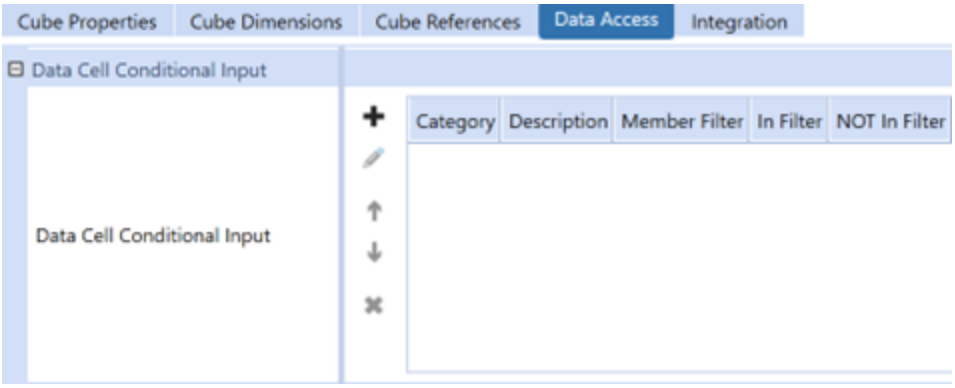
Data Cell Conditional Input Security

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

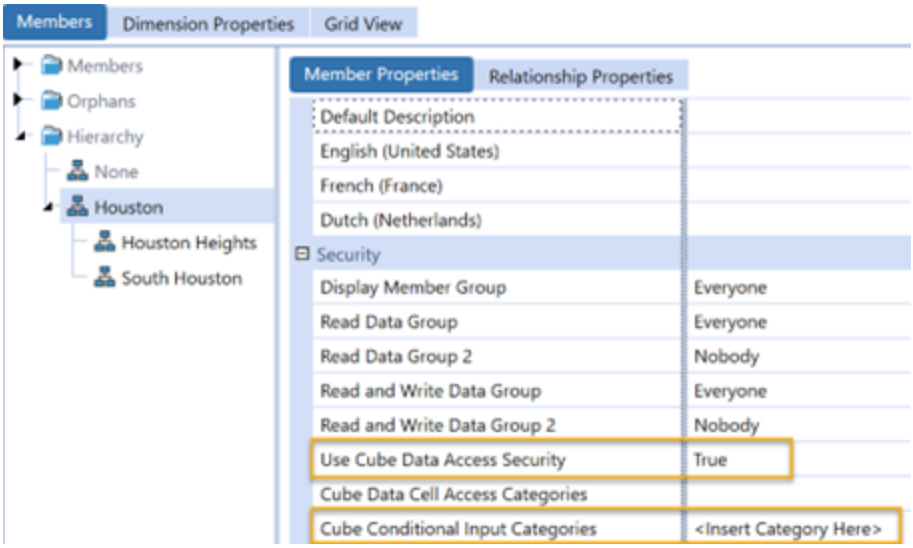
Data Cell Conditional Input shares the same properties and behaviors as Data Cell Access Security. However, Data Cell Conditional Input does not have an Access Group property. When configuring the Member Filters for a Data Cell Conditional Input rule, all users will either have access or not have access 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)

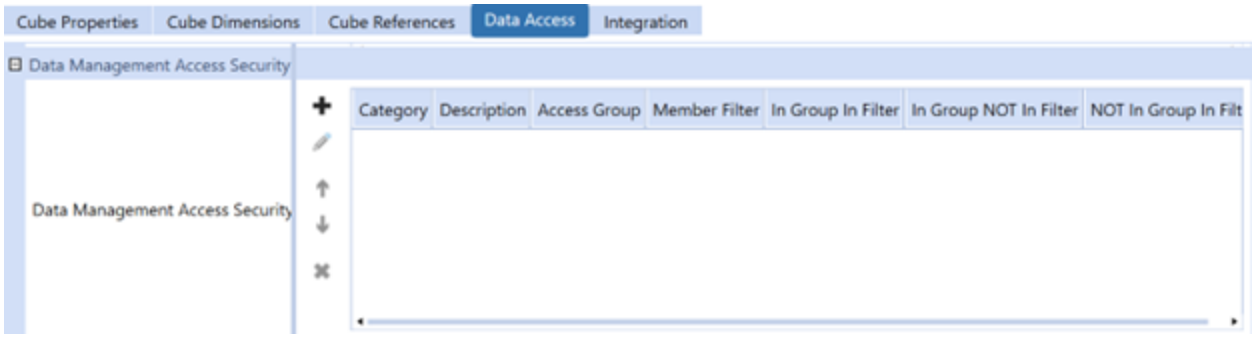
Data Management Access Security

This section provides a level of Cube security when running processes from a Data Management Sequence or Step. See Data Management for more information.

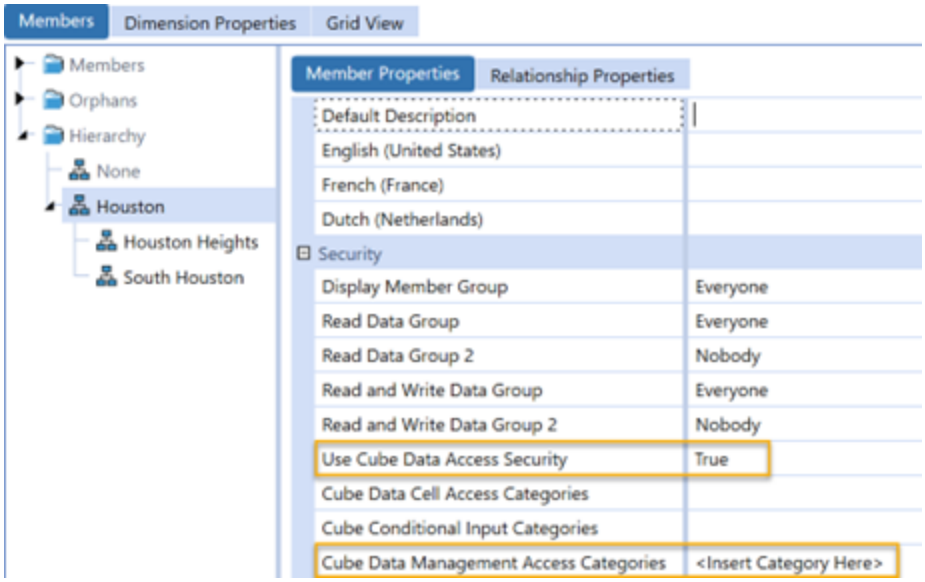
Data Management Access Security shares the same properties and behaviors as Data Cell Access Security. The purpose of Data Management Access Security is to control who has access to read or modify Cube data when using a Data Management process. This could be a Data Management process intended for running 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. See Data Unit, Data Cell Access Security, and Data Unit for more information.

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 allows you to share the same integrations or configured 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

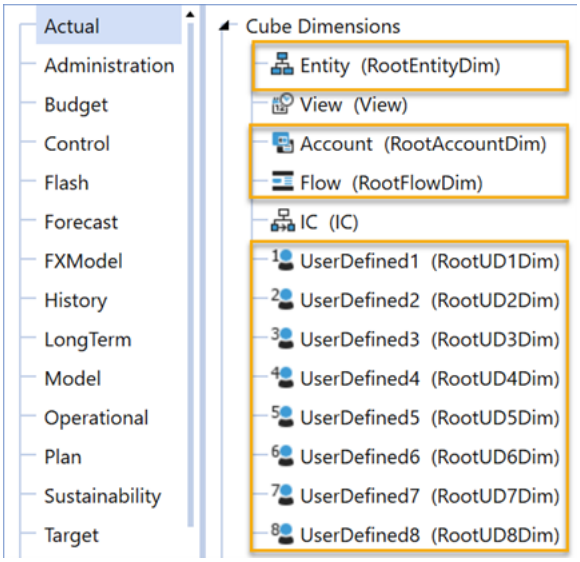
• Attribute Dimensions

• Attribute Value Dimensions

Cube Dimensions

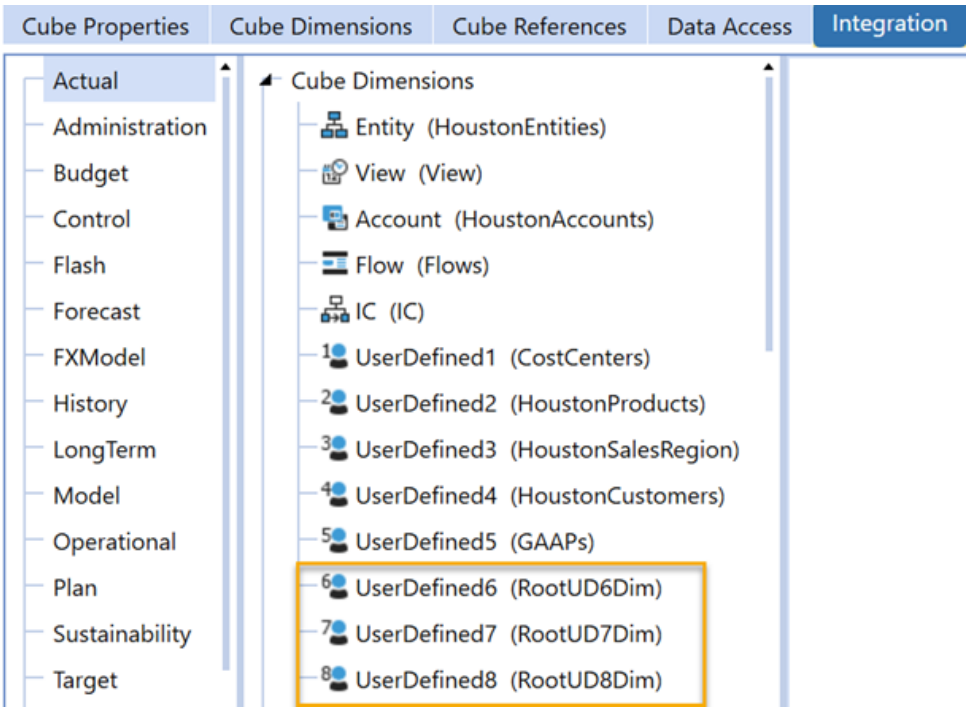
In this section, each Scenario Type are assigned Dimension Types defined in the Cube Dimensions tab.

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.

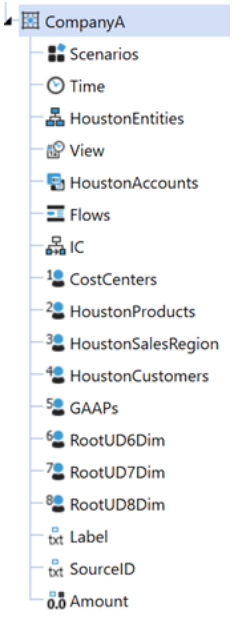
> **Example:** In the example below, this shows the Integration tab after Dimensions are assigned to Dimension Types by Scenario Type in the Cube Dimensions tab.



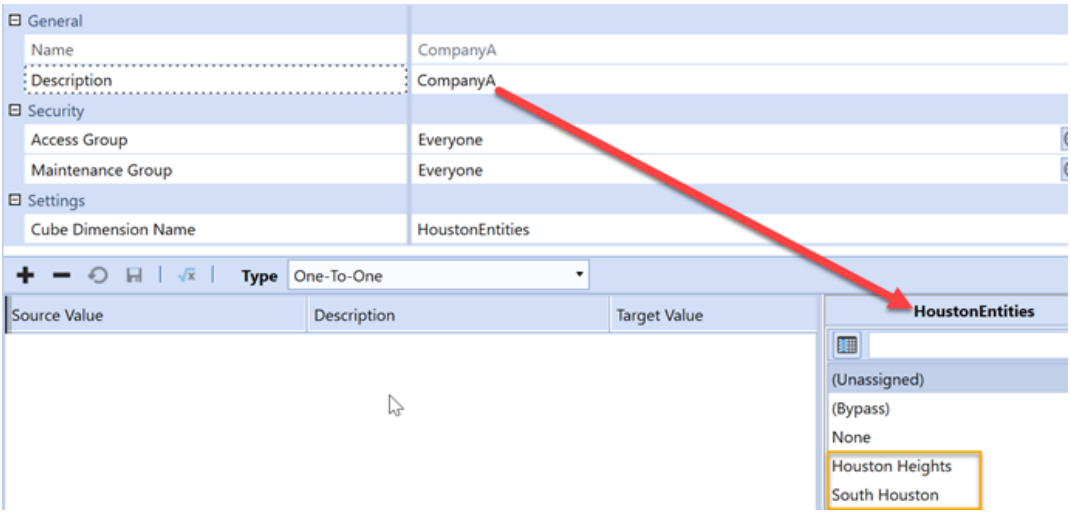
> **NOTE:** All the Dimension Types are enabled, by default, in the Actual Scenario Type. This is apparent because none of the Dimension Types are grayed out. When the Dimension Type is disabled, the Dimension Type displays gray.

> **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.

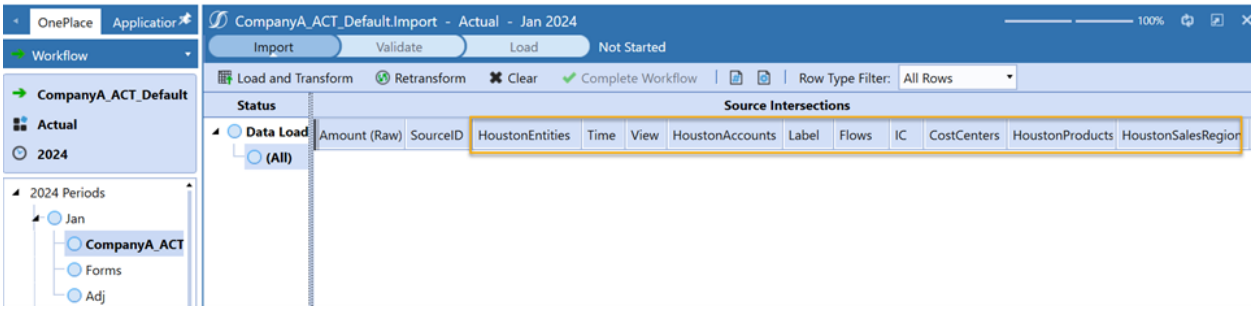
The configuration of the Integration tab is important as this drives everything data loading related. This impacts how the Data Sources are created, structured, and configured.



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, and Houston & Houston Heights = Dimension Members.

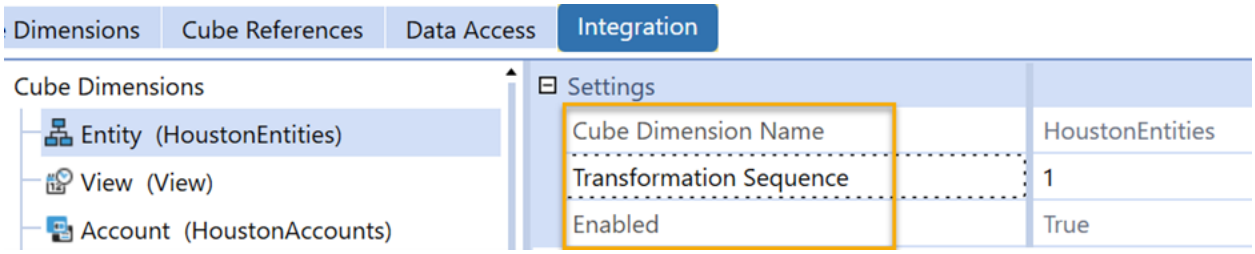


This controls how Stage tables display through the Import Workflow process. Workflow Profiles abide by the integration. The Integration tab integrates the Data Source, Transformation Rules, Workflow Profiles together.



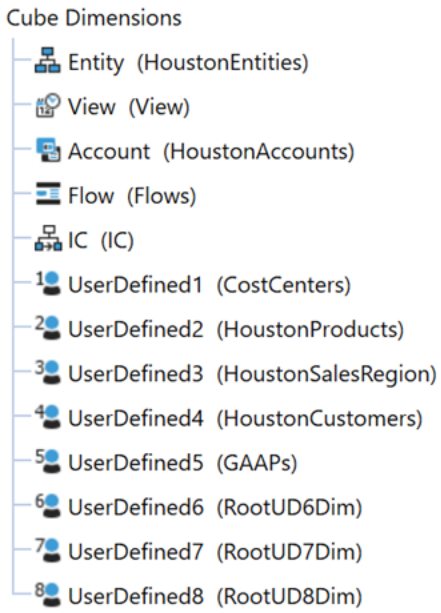
Cube Dimensions Settings

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. This property is Read-Only. 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 for more information.

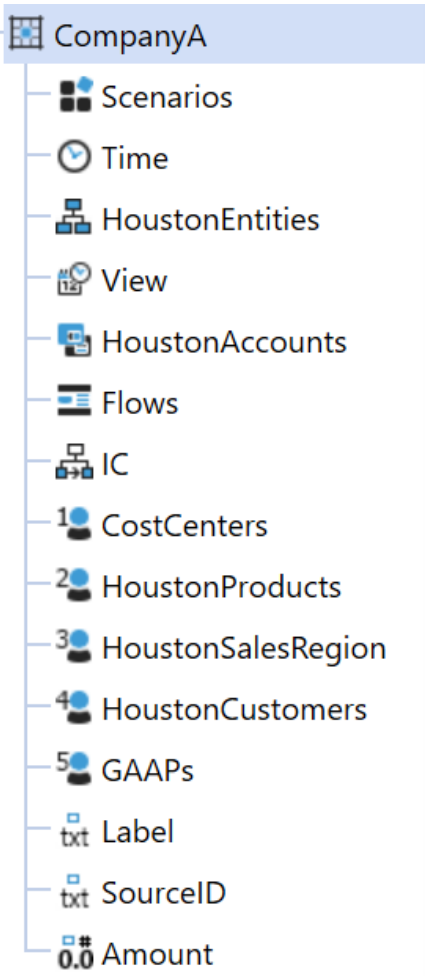
Transformation Sequence: By default, each Dimension Type is assigned a numeric value to identify the order in which the Dimension Type transformations should occur. By default, the order in which the transformations will occur is shown below Entity> UD8.



Enabled: By default, enabled is set to True for each Dimension Type for a Scenario Type. Entity, View, and Account Enabled property is 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. Also, 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. See Data Sources and Transformation Rules.

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.

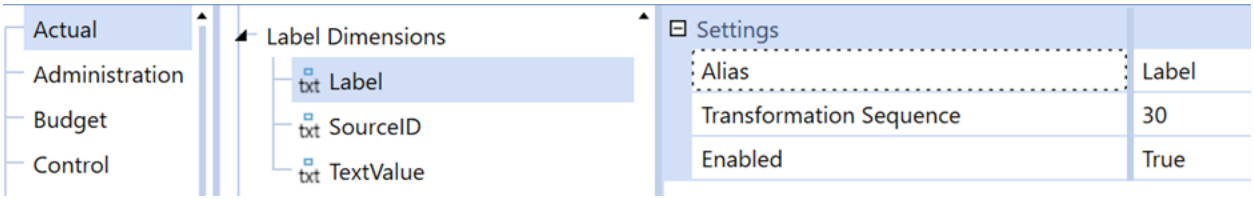
Label Dimensions

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 its 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 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.

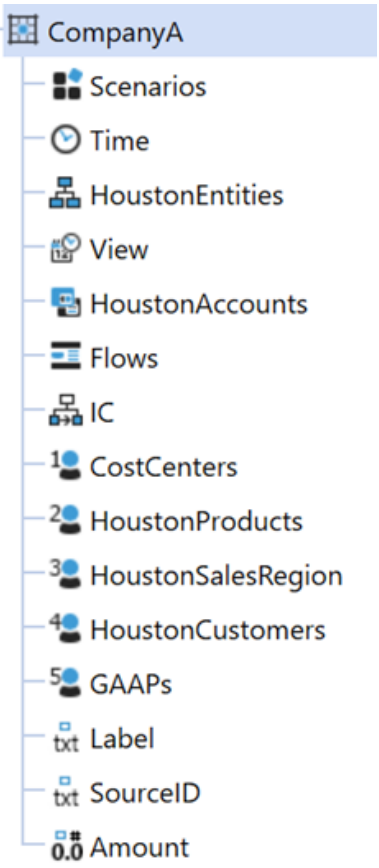
TextValue: This is to store large amounts of textual data. Max size is 2GB of text.

Attribute Dimensions

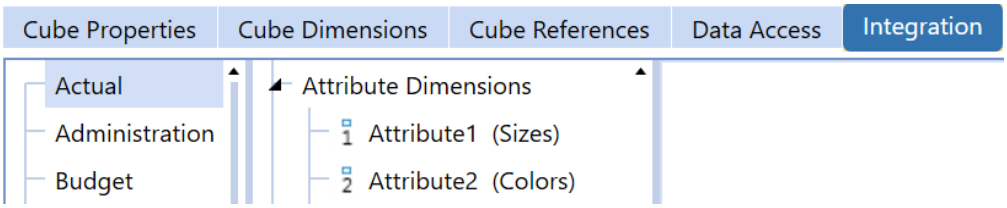
The Attribute Dimensions are used independently or in concert with the Cube Dimensions for reporting purposes. When working with the Cube Dimensions, they can be considered as a feature or characteristic of a Dimension and Dimension Member. As an alternative, they can also be created completely independent 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.

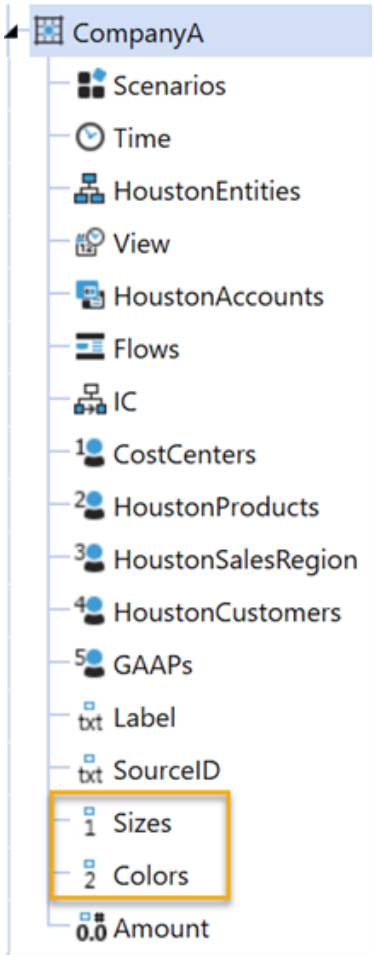
By default, Attribute Dimensions are disabled. Each Scenario Type and process will determine whether they are needed. Therefore, the default behavior is to disable all the Attributes. The same situation exists as with Cube Dimensions but in the opposite order of operation. By default, they are disabled 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:



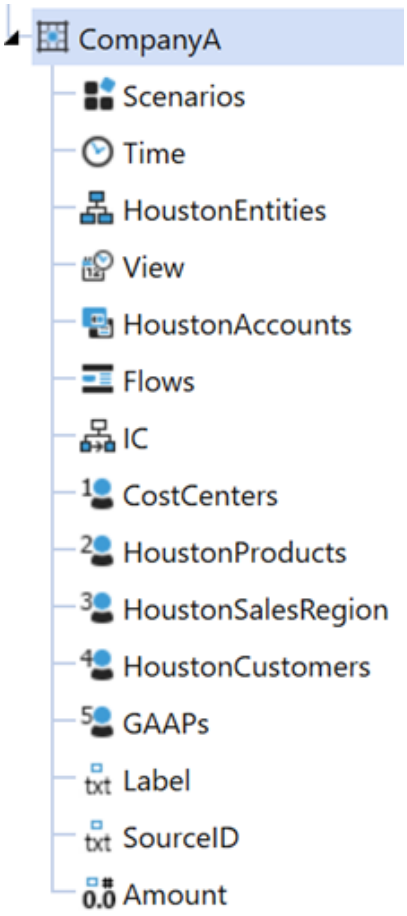
If the Data Source is created prior to setting the Enabled property from False to True for an Attribute Dimension, the Data Source will be constructed with the Dimension Types configured and no Attribute Dimension. You would then have to manually add the Attribute Dimensions to the Data Source.

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.

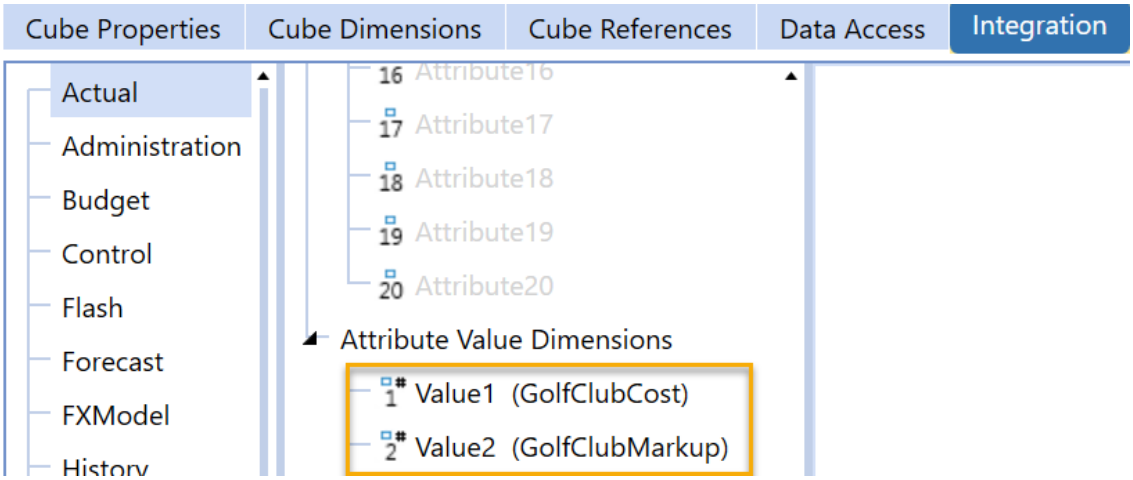
Attribute Value Dimensions

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.

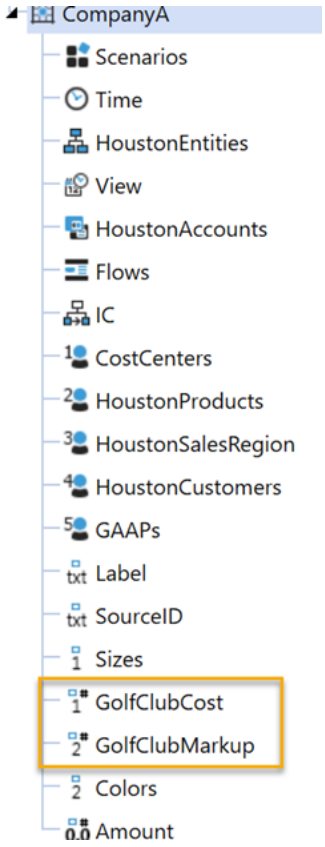
Attribute Value Dimensions should be defined and configured prior to creating a Data Source. If they are not enabled at the time of Data Source creation, this is how the Data Source would display: No Attributes Values available but all Cube Dimensions available.



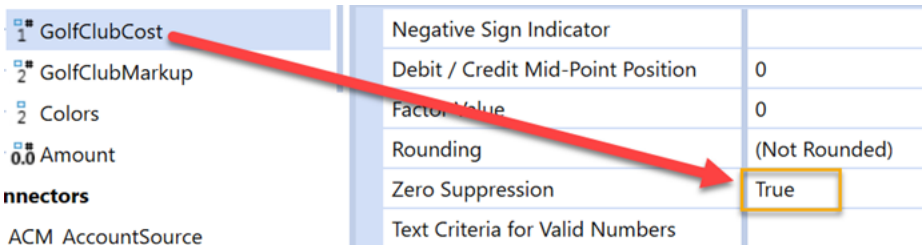
NOTE: By enabling the Attribute Value Dimensions for a Scenario Type prior to the creation of a Data Source, this is how they will display:



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. (e.g., 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 defines what to display in the rows

Column Axis

This drop down 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.

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 via BRAPI Rules will require Application Security Roles / LockFxRates.
- Users that will unlock rates manually or via BRAPI Rules will require Application Security Roles / UnLockFxRates.

Security Roles that Manage the Locking Features

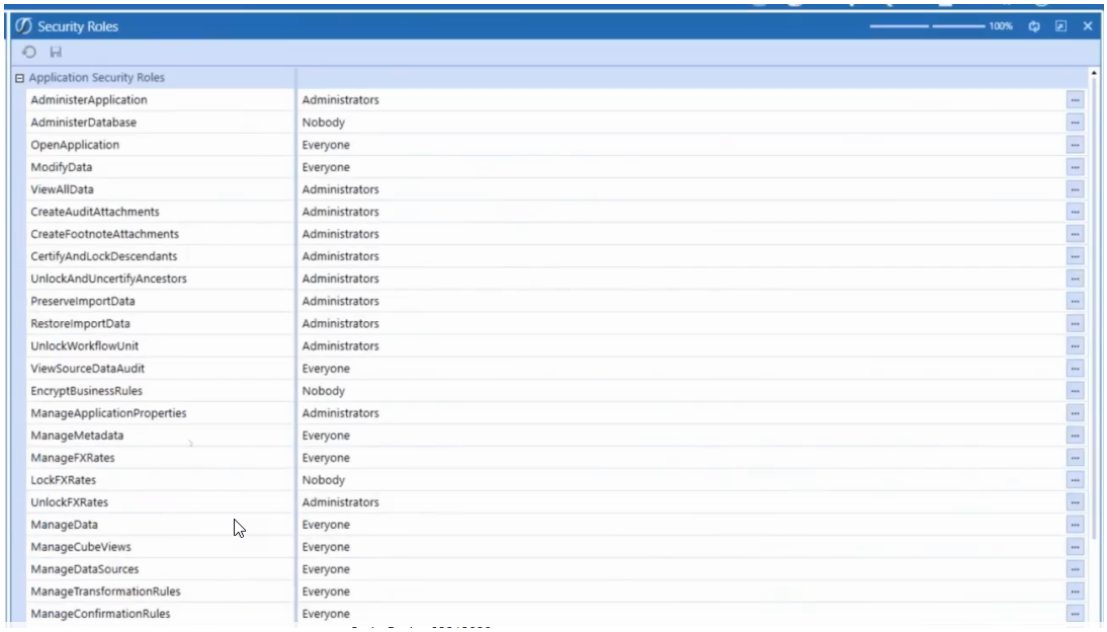
Depending on the settings for LockFXRates and UnlockFXRates, users will have access to the Manage FX Rate Locking window.

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: track the current status of the Rate Type as to who performed the unlock or lock.

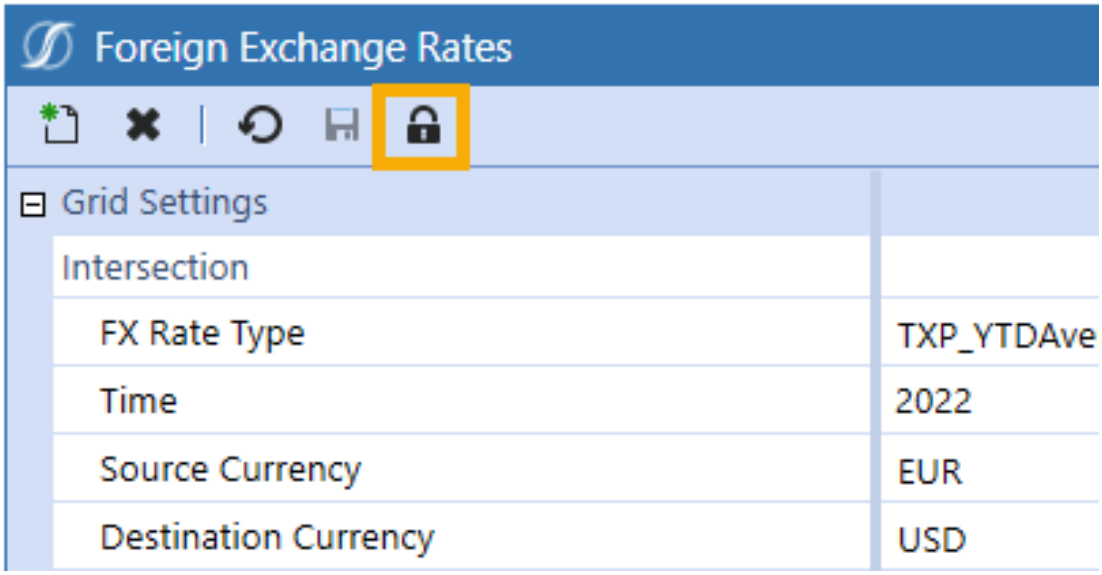
View Task Activity

There are two new Task Activity logs produced. Lock Fx Rate and Unlock Fx Rate will be generated whenever the Lock status is modified.

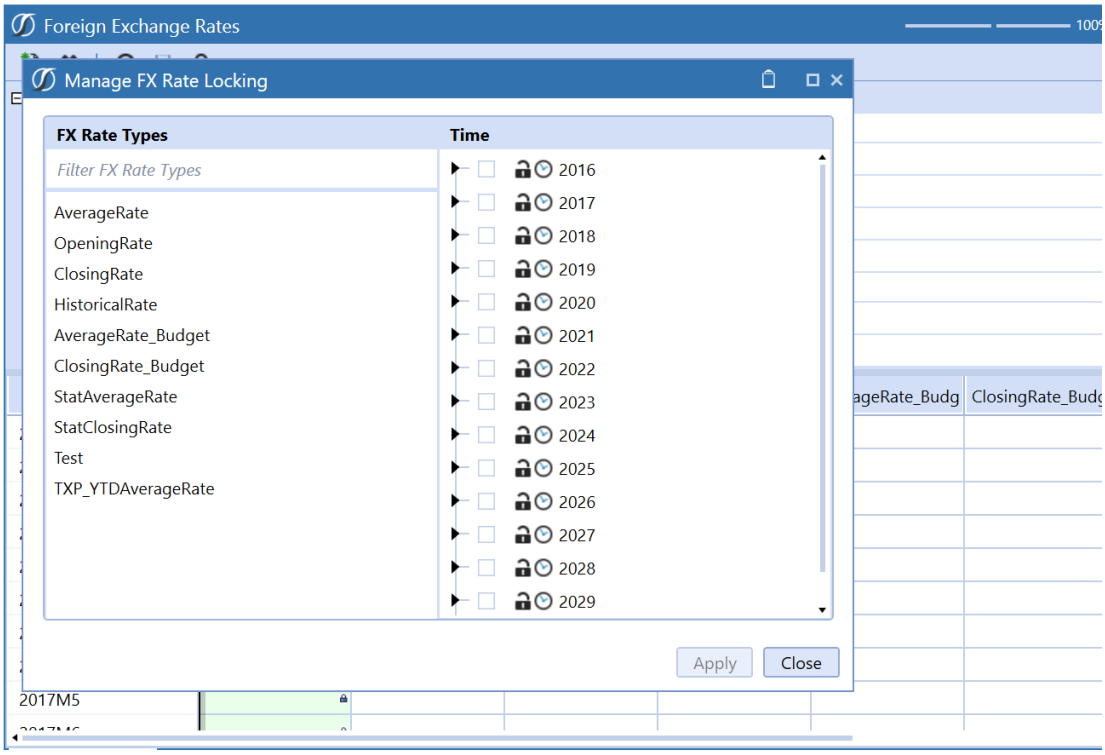
1. From the application, click **Task Activity**.
2. In the task list you will see which users have locked or unlocked rates and the time.

Lock and Unlock Rates

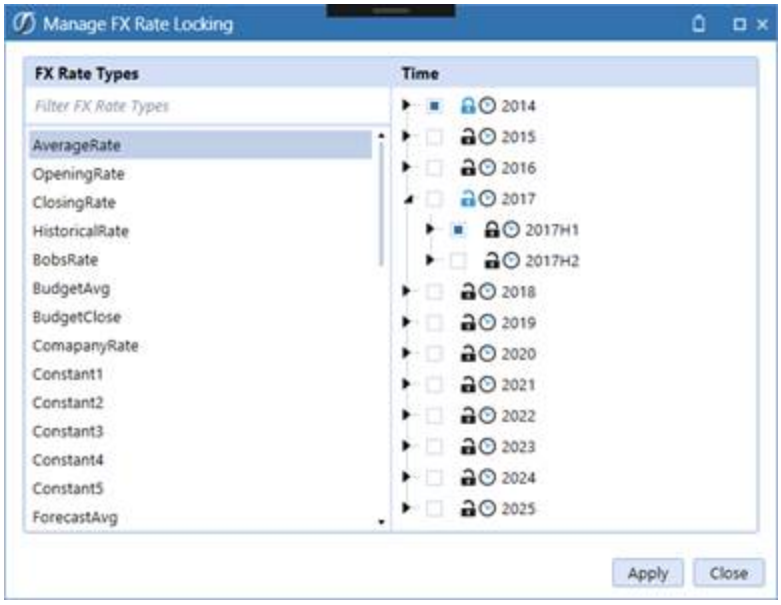
1. From the Application tab, go to **Cube > FX Rates**.
2. Click the lock icon at the top of the page.



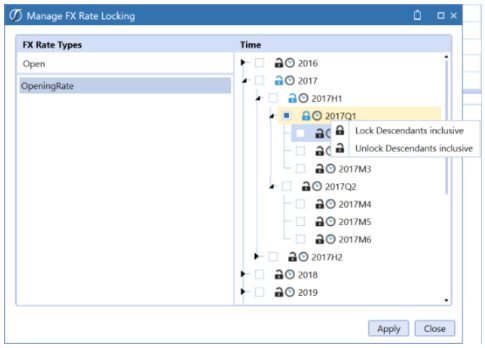
3. The Manage FX Rate Locking window opens.



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 selection “check box”, it will set the Lock status as a Lock or Unlock icon. 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. Click **Apply** and then click **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.

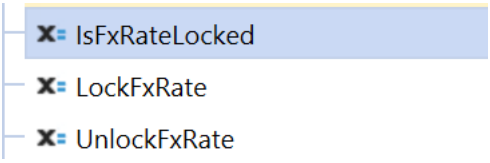
Foreign Exchange Rates

Grid Settings	
Intersection	
FX Rate Type	OpeningRate
Time	2017M1
Source Currency	CAD
Destination Currency	GBP
Rows and Columns	
Row Axis	Time
Column Axis	FX Rate Types

	AverageRate	OpeningRate	ClosingRate	HistoricalRate	AverageRate_Budg
2017					
2017H1					
2017Q1					
2017M1					
2017M2					
2017M3					
2017Q2					
2017M4					
2017M5					

BRAPI Rule Functions

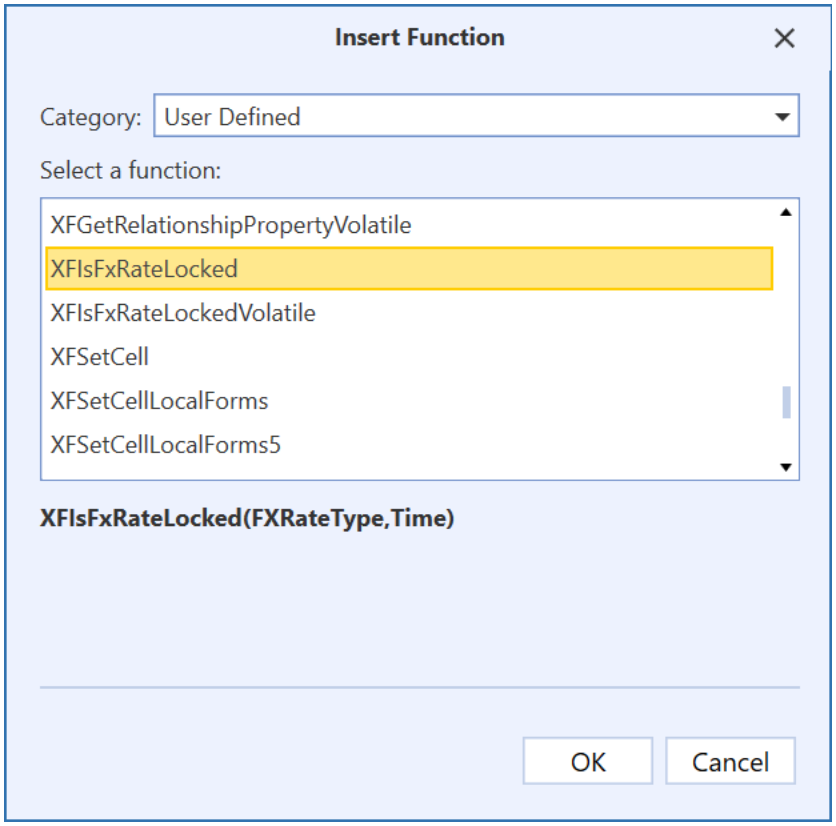
There are three BRApi's that allow for automation and dashboard development, ISFxRateLocked, LockFxRate, and UnlockFxRate.



Spreadsheet or Excel Functions

Retrieve the status in reports. n spreadsheet or Excel, you can go to functions and select XFisFXRateLocked for your rate sheets.

1. From the Application tab, go to **Tools > Spreadsheet**.
2. Click the **OneStream** tab.
3. Click the **Insert Function** button.
4. In Category, click **User Defined**.
5. In Select a function, click **XFIsFxRateLocked**.



6. Click **OK**. The Function Arguments window opens.
7. Enter a value in FxRateType.
8. Enter a value in Time.
9. Click **OK**. The spreadsheet will populate with the new data.

> **NOTE:** If you change the lock or unlock rate types in the spreadsheet, when you Refresh the application and return to the Rates grid, the data will be updated to reflect your changes in the spreadsheet.

Member Filters

A Member Filter can be used to filter the data down to what an administrator wants to see by creating a list of restricted Members. Member Filters contain multiple Member Scripts separated by commas.

A Member Script is a brief statement typed to query one defined set of Dimensional Members. Members can be specified for any or all Dimensions and the primary Dimension can also specify a Member Expansion Formula (e.g., .Descendants).

Here is an example of a simple Member Script that returns the year 2012:

T#2012

Here is an example of a Member Script with a Member expansion that returns all the Income Statement Accounts.

A#[Income Statement].Descendants

If one or just a few Dimensions in the Member Script are qualified, the remaining Dimensions are pulled either from the Cube View POV, the Global POV, or Workflow, Time, and Scenario. Separate each Dimension qualified in the Member Script with a colon. Here is an example of a fully qualified Member Script:

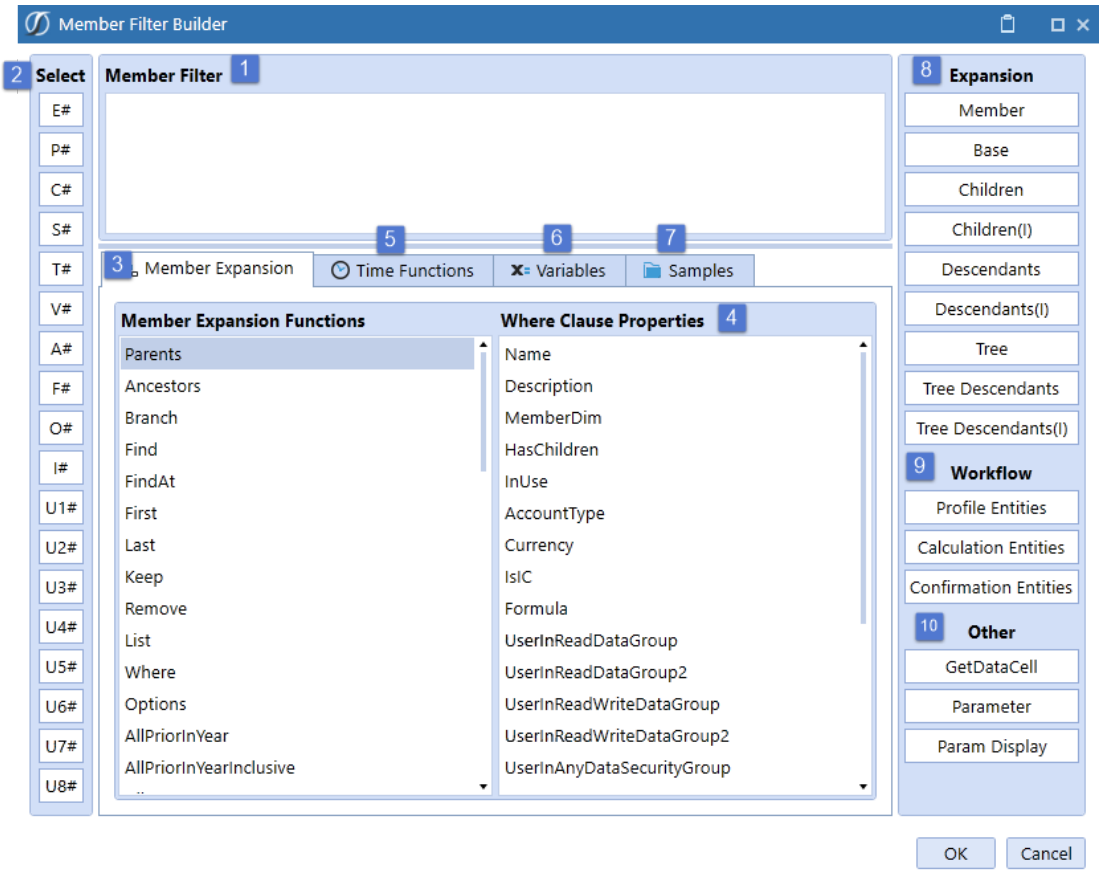
Cb#GolfStream:E#Houston:P#Texas:C#Local:S#Budget:T#2012M3:V#YTD:A#60000:F#None:O#Forms:
IC#None: U1#Sales: U2#HybridXL: U3#Northeast: U4#SportsCo: U5#None: U6#None: U7#None: U8#None

Here is an example of a Member Filter made up of three different Member Scripts that returns the Actual, Budget and Forecast Scenarios.

S#Actual, S#Budget, S#Forecast

Member Filter Builder

The dialog makes it simple to build complex Member Filters without having to remember or look up the proper syntax. This dialog is embedded in areas such as Cube Views and the Dimension Library.



1. Member Filter

   This area is where the Member Filter will be built. Type in this section or use the dialog to help fill it out.

2. Member Selection

   There is a button for each Dimension here that will launch the appropriate selection dialog.

3. Member Expansion Functions
   Double-click on a Member Expansion to add it.
4. Member Expansion Where Clause
   If the Member Expansion Where is selected, use the Where Clause Properties to complete the expression.
   Example: UD2#AllProducts.Children.Where(Name Contains Clubs)
5. Time Functions
   These only apply to the Time Dimension, such as T#POVPrior1. Double-click on a Time Function to add it.
6. Substitution Variables
   Double-click on a system wide Substitution Variable to add it.
7. Samples
   Refer to this tab for examples on how to build Member Expansions, Time Functions, Where Clause Expressions, GetDataCell Expressions, and Custom Member List Expressions.
8. Expansion
   These buttons are commonly used Member Expansions. Click the Expansion to add it.
9. Workflow
   These buttons are commonly used Workflow Member Expansions used in Cube Views that point to a Report, Form, or Dashboard and are affiliated with a specific Workflow Profile.
10. Other
    These buttons are commonly used Member Filter Functions which allow the user to create calculated columns and rows or use a custom Parameter to store Member lists.

Member Script Abbreviations and Examples

There are 18 Dimensions that can be used to filter data. This section briefly explains each Dimension and gives an example. See Dimension for a full description of each Dimension. Square brackets are optional around Member names, but are mandatory if the Member name contains a space, period, comma, or semi-colon.

Cube

Cb#GolfStream

The specific Cube being referenced.

Entity

E#Houston

The specific Entity being referenced.

Parent

P#Texas

The Parent of the Entity being referenced. This is important because Entities can be rolled up into multiple hierarchies.

Consolidation

C#Local

The specific Member of the Consolidation Dimension being referenced.

Scenario

S#Actual

The specific Scenario being referenced.

Time

T#2012Q1

Specific time period in focus. There are several selections available relative in nature to certain boundaries like the time period set in the POV, the selected Workflow time period, or the Global time period. More of these constants are explained in the Member Script Constants section later in this document.

T#POV

This returns the current period of the selected year in the Cube POV.

T#W1

This returns the first week of the selected year in the Cube POV (this only applies to weekly applications).

T#M1

This returns the first month of the selected year in the Cube POV.

T#Q1

This returns the first quarter of the selected year in the Cube POV.

T#H1

This returns the first half of the selected year in the Cube POV.

View

V#YTD

The specific view of the requested information.

Account

A#60000

The specific Account being referenced.

Flow

F#BegBal

Specific Flow Member being referenced. There is a dedicated Flow Dimension.

Origin

O#Import

The Origin of the data being referenced and then deciding if it was imported, entered as a Journal, or entered into a Form.

UD1 to UD8

U2#[Fairway Woods] or U3#TotalRegions

Specific Members from the eight User Defined Dimensions are being referenced. If all eight of these Dimensions are not being used, they still need to be qualified in the POV, in a Cube View, or in the Member Script to get a proper intersection. Each User Defined Dimension has a Member called None or the reserved word All that can be used (e.g. U5#All) in some areas of the product such as Intercompany Reporting settings.

Member Script Constants

Member Script Constants are used as a point of reference when querying data for Members and data. Below are the three constant types:

Point of View (POV)

Select Members based on their literal or relative position to the selected main Point of View (POV) or the Cube View POV. The main POV settings are available under the Application Tab| Tools | Application Properties. Call out a Dimension to pull its Member value from what is selected in the POV in this way: E#POV, C#POV, T#POV, V#POV, A#POV,... (all Dimension types).

Workflow (WF)

Refer to the time period or Scenario currently selected in the Workflow module. Examples are T#WF and S#WF.

Global

The Application has a Global Time and Global Scenario setting found under the Application Tab|Tools|Application Properties. These settings can be used optionally and can be referenced (e.g., T#Global and S#Global). An example of using this as a reference is to build a Cube View and have Global Time and everything before it appear in columns and the Actual Scenario and everything after it appear as the Forecast Scenario.

Member Expansion Functions

Additional commands can be added at the end of a Member Script to help expand the presented results. For example, E#US will return just the US Entity, while E#US.Descendants will return all of the child Entities that fall under the US Entity such as California, Texas, New York, etc.

Member Expansions

To illustrate these expansion concepts, the following examples will use a portion of the Entity structure from GolfStream Corporation, which is a golf supply company. This is the management rollup, but there is also a geographical rollup where Houston rolls up to Texas, Carlsbad rolls up to California, Frankfurt rolls up to Europe, etc.

- Total GolfStream
    - Clubs
        - NA Clubs
            - Canada Clubs
                - Montreal
                - Quebec City
            - US Clubs

        - Augusta
        - Carlsbad
        - Houston
    - Frankfurt
- Golf Balls
    - Europe Golf Balls
    - NA Golf Balls
- Accessories & Apparel
- Course Management
- Corporate

Add these expansions after a Member name to return the desired Members like this: E#Houston.Ancestors.

> **TIP:** Use square brackets, [ ], to reference any name with a space. For example, E#[NA Clubs].

Member

This lists the Member requested. E#Frankfurt.Member is the same as stating E#Frankfurt. This expansion can be combined with another member expansion to create an inclusive list. This is an example of using Member for a Children Inclusive list:

E#[Total GolfStream].Member.Children returns the same as E#[Total GolfStream].ChildrenInclusive:

- Total GolfStream
    - Clubs
    - Golf Balls
    - Accessories & Apparel
    - Course Management
    - Corporate

This is an example of using Member for a Parent Inclusive list:

E#Clubs.Member.Parents returns Clubs and its parent, Total GolfStream.

This is an example of using Member for an Ancestors Inclusive list:

E#[NA Clubs].Member.Ancestors returns NA Clubs and its ancestors, Clubs and Total GolfStream.

Base
E#[NA Clubs].Base queries only the base level Entities and excluding any other aggregate Members:

- Montreal
- Quebec City
- Augusta
- Carlsbad
- Houston

Children
E#Clubs.Children returns the first level children in a flat list:

- NA Clubs
- Frankfurt

ChildrenInclusive
E#[NA Clubs].ChildrenInclusive returns both NA Clubs and its children:

- NA Clubs
    - Canada Clubs
    - US Clubs

This can be chained together with another .Children statement to see the next level as well. This may also be referred to as Children.Children elsewhere in the documentation.

E#[NA Clubs].ChildrenInclusive.Children returns:

- NA Clubs
    - Canada Clubs
        - Montreal
        - Quebec City
    - US Clubs
        - Augusta
        - Carlsbad
        - Houston

Descendants
E#[NA Clubs].Descendants lists every Member under NA Club in a flat, non-hierarchical list excluding NA Clubs:

- Canada Clubs
- Montreal
- Quebec City
- US Clubs
- Augusta
- Carlsbad
- Houston

DescendantsInclusive

E#[US Clubs].DescendantsInclusive lists every Member under US Clubs in a flat, non-hierarchical list including US Clubs:

- Augusta
- Carlsbad
- Houston Heights
- South Houston
- Houston
- US Clubs

Tree

E#[US Clubs].Tree returns the specified Member and all Members below it in a hierarchy:
This expansion automatically includes expandable rows.

- US Clubs
    - Augusta
    - Carlsbad
    - Houston

TreeDescendants

E#Clubs.TreeDescendants lists every Member under Clubs in hierarchical tree excluding Clubs:

- NA Clubs
    - Canada Clubs
        - Montreal
        - Quebec City
    - US Clubs
        - Augusta
        - Carlsbad
        - Houston
- Frankfurt

TreeDescendantsInclusive

E#[US Clubs].TreeDescendantsInclusive lists every Member under US Clubs in a hierarchical tree including US Clubs:

- US Clubs
    - Augusta
    - Carlsbad
    - Houston

Parents

E#Houston.Parents returns the direct Parents of the given Member regardless of how many hierarchies to which the Member belongs:

- US Clubs (from the management rollup)
- Texas (from the geographical rollup)

The Parent is derived from the Cube View's POV setting by default. If a Member is used in multiple hierarchies, specify a specific Parent using the following syntax in order to override it: E#Houston.Base:P#USClubs or E#Houston.TreeDescendants:P#Texas.

> **NOTE:** P# works differently when using it with an expandable Tree filter because that filter processes the children as individual queries, so the P# will only apply to the top-level Members. Use E#Houston.TreeDescendants:P#Texas instead, or specify a Parent on the Cube View's POV settings.

Ancestors

E#[NA Clubs].Ancestors returns all Members up the chain from NA Clubs:

- Total GolfStream
- Clubs

Branch

Expand multiple Members of a given expansion by finding specific items and then performing additional expansions on those items. Refer to the Samples Tab in the Member Filter Builder for an example of the syntax.

E#[Total GolfStream].Children.Branch(Find([Clubs]).Children, Find(Golf Balls).Children) returns Total GolfStream's Children, it then finds Clubs' and Golf Balls' Children and returns them in the hierarchy as well.

- Total GolfStream
    - Clubs
        - NA Clubs
        - Frankfurt
    - Golf Balls
        - Europe Golf Balls
        - NA Golf Balls
    - Accessories & Apparel
    - Course Management
    - Corporate

This is also used with Quick Views in order to expand several hierarchies at one time.

Find

Find will apply Member expansions to a nested subset of results:

A#[Income Statement].Descendants.Find(64000).Children will return all of the descendants of the Income Statement as a flat list, but will find account 64000 (i.e. Earnings Before Taxes in this example) and then indent and present that account's children.

FindAt

This returns a specific Member of a given expansion using a zero-based position index and performing an additional expansion on the specific item. Refer to the Samples Tab in the Member Filter Builder for an example of the syntax.

E#[Total GolfStream].Children.FindAt(1).Children returns Total GolfStream's Children, locates the Member in the first position, and returns its children as well.

- Total GolfStream
    - Clubs
    - Golf Balls
        - Europe Golf Balls
        - NA Golf Balls
    - Accessories & Apparel
    - Course Management
    - Corporate

First

First will find the first items in the list of results and allow additional Member Filters to be applied:

E#[Total GolfStream].Children.First.Children will list all of the children of Total GolfStream, go to the first Entity (Clubs) and then show its children:

- Clubs
    - NA Clubs
    - Frankfurt
- Golf Balls
- Accessories & Apparel
- Course Management
- Corporate

Last

Similar to First, Last will find the last items in the list of results and allow additional Member Filters to be applied.

Keep

Keep will search the results from a Member Filter and only keep certain values:

E#[Total GolfStream].Children.Keep(Clubs, [Golf Balls]).Children will list all of the first level children of Total GolfStream, only keeping Clubs and Golf Balls and then show their children:

- Clubs
    - NA Clubs
    - Frankfurt
- Golf Balls
    - Europe Golf Balls
    - NA Golf Balls

Remove

This will remove some of the Members from the results:

E#[Total GolfStream].Children.Remove(Corporate).Find(Clubs).Children.Find([NA Clubs]).Children will list several Entities and then remove the Corporate Entity from the results:

- Clubs
    - NA Clubs
        - Canada Clubs
        - US Clubs
    - Frankfurt
- Golf Balls
- Accessories & Apparel
- Course Management

List

This will create a list of specific Members:

A#Root.List(63000, 64000) will return these two accounts.

An Indent Level can also be specified when defining a list of Members. Refer to the Samples Tab in the Member Filter Builder for an example of the syntax.

E#[Total GolfStream].List(Clubs.IndentLevel(0), [Golf Balls].IndentLevel(2), [Course Management].IndentLevel(1))

- Clubs
    - Golf Balls
        - Course Management

Where

The Where clause in a Member Filter allows further qualification of the results. It adds an additional level of filtering after the list of members has been retrieved. Use cases are provided below.

You can filter members where a property is a certain value:

- X#MyMember.Where(PropertyName = 'Value')
- E#[Total GolfStream].Descendants.Where(IsIC = True)

You can add other member expansions, such as .Children, before and after the Where clause:

- X#MyMember.MemberExpansion.Where(PropertyName = 'Value').MemberExpansion
- A#Sales.Descendants.Where(Name Contains '60000').Children

You can filter for multiple values using detailed conditions within .Where() using AND, OR with additional parentheses:

- X#MyMember.Where((PropertyName = 'Value') OR (PropertyName = 'Value'))
- A#60999.Descendants.Where((Name Contains '60000') OR (Name Contains '41000'))
- E#Root.Children.Find(US).Children.Find(Michigan, Texas)Children.Where((Name = Flint) OR (Name = Rochester) OR (Name = Dallas)). This returns Michigan and Texas and only specific child city Entities.

You can use the IN or NotIN qualifiers to return members in or not in a list:

- X#MyMember.Where(PropertyName In '60000')
- A#60999.Descendants.Where(Name In '60000', '41000') returns 60000 as it is a descendant of 60999. 41000 is not returned as it is not a descendant of 60999.
- E#Root.Children.Find(US).Children.Find(Michigan, Texas). Children.Where(Name In Flint, Rochester, Dallas) returns Michigan and Texas and only a few child Entities.

You can also add a filter to another member expansion, such as .Descendants, with or without .Where:

- A#MyMember.MemberExpansion(MyPropertyName='MyValue')
- A#Sales.Descendants.Where(Name = '60000')
- A#Sales.Descendants(Name = '60000')

You can combine criteria within the Where clause, such as filtering for different properties:
A#Sales.Descendants.Where((HasChildren = True) AND (Name Contains '60000'))
You can use operators other than = within the Where Clause. Below is the list of supported operators:

- **>=**
- <=
- <>
- =

- >
- <
- StartsWith
- Contains
- DoesNotContain
- EndsWith

Operators with greater than or less than are not applicable to the text-based fields:

- >=
- <=
- >
- <

See Member Expansion Where Clause for examples using common member properties.

Options

Use this to reference sub-cubes and specify how dimensions should be processed. Options must immediately follow the expansion function it is being used for.

For example, A#MyAccount.Base might return a different list when looking at an extended dimension associated with a different cube. Therefore, the cube and Scenario Type can be specified using Options:

A#MyAccount.Base.Options(Cube=CubeName, ScenarioType=ScenarioTypeName, MergeMembersFromReferencedCubes=False)

The following example returns account 19999's children from the Total GolfStream cube and Actual Scenario Type:

A#19999.Children.Options(Cube=[Total GolfStream], ScenarioType=Actual, MergeMembersFromReferencedCubes=False)

If you drill to base members in a Cube View or Quick View, you can use Options to ensure you are returning the desired members. For example, A#60000.Base could return different members due to extended dimensions. Using Options ensures you are returning the desired members:

- A#60000.Base.Options(Cube=Houston, ScenarioType=Actual, MergeMembersFromReferencedCubes=False)
- A#60000.Base.Options(Cube=NewYork, ScenarioType=Actual, MergeMembersFromReferencedCubes=False)

The following example returns tree descendants inclusive of the account based on the workflow cube and Scenario Type property of the workflow:
A#MyAccount.TreeDescendantsInclusive.Options(Cube=|WFCube|, ScenarioType=XFMemberProperty(DimType=Scenario,Member=|WFScenario|, Property=ScenarioType),MergeMembersFromReferencedCubes=False)

Combined Expressions

You can stack multiple Member Expansions to display additional members, add conditions, remove members, and more.

This example shows Children of Children which display the children members and their children members.

E#[Total GolfStream].Children.Children returns:

- Clubs
    - NA Clubs
    - Frankfurt
- Golf Balls
    - Europe Golf Balls
    - NA Golf Balls
- Accessories & Apparel
- Course Management
- Corporate

This example uses the List and Find expressions to display base and parent members:

E#[Total GolfStream].List(Clubs,Corporate,Frankfurt).Find(Clubs).Base.Find(Carlsbad).Parents returns:

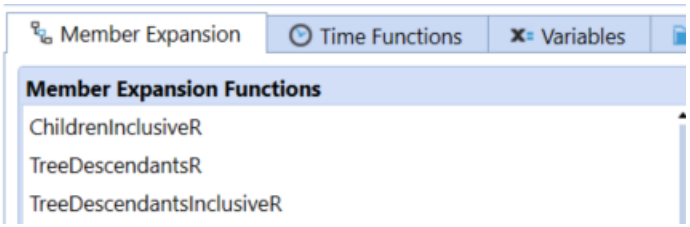
- Clubs
- Montreal
- Quebec City
- Augusta
- Carlsbad
- California
- US Clubs
- Total Product Rollup
- Houston Heights
- South Houston
- Frankfurt
- Corporate
- Frankfurt

For more Member Expansion examples, refer to the Samples Tab in the Member Filter Builder.

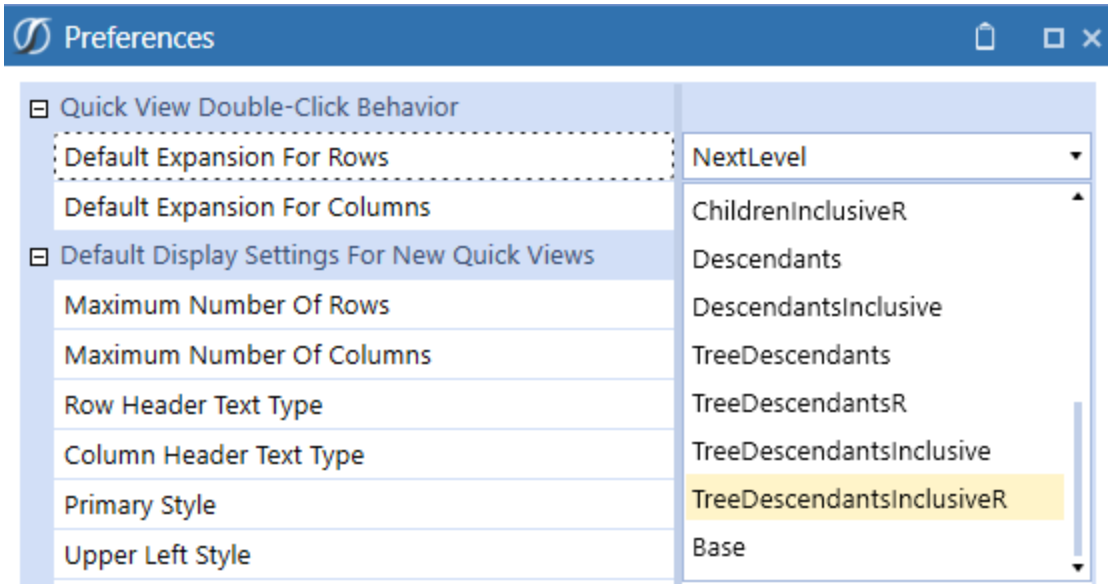
Reverse Order Member Expansions

Functionality to support reversing the direction of the order of results from certain member filters are supported in Cube Views and Quick Views. These reverse order selections include ChildrenInclusiveR, TreeDescendantsR, and TreeDescendantsInclusiveR.

In Cube Views, this functionality can be set in the Member Filter Builder using the Member Expansion Functions tab for Rows (and/or) Columns.



In Quick Views, this can be set in Preferences under the Quick View Double-Click Behavior section in the Default Expansion for Rows (and/or) Columns.



This example will demonstrate the TreeDescendantsR being used in a Quick View. This Gross Income view below has been created using Next Level expanison (available in a Quick View):

	Jan 2018	Feb 2018
61000 - Gross Income	27,462,816	28,236,051
60999 - Net Sales	56,155,448	56,119,871
60000 - Operating Sales	57,982,342	57,944,272
2000_100 - Third Party Sales	50,945,236	50,985,236
2000_200 - OEM Sales	4,606,157	4,536,127
2000_300 - Subassembly Sales	382,840	382,800
2000_400 - Parts Sales	2,048,108	2,040,108
60100 - IC Sales	15,000	15,990
60200 - Returns & Allowances	1,841,894	1,840,391
2200_100 - Third Party Returns & Allowances	728,117	727,974
2200_200 - OEM Returns & Allowances	37,372	37,312
2200_300 - Subassembly Returns & Allowances	95,952	95,052
2200_400 - Parts Returns & Allowances	980,454	980,054
60250 - Other Outside Sales		
43000 - Cost of Goods Sold	28,692,632	27,883,820
41000 - Operating Cost of Goods Sold	28,677,632	27,868,820
42000 - IC Cost of Goods Sold	15,000	15,000

To change the double-click behavior, select Preferences under the Administration menu. In the Quick View Double-Click Behavior section, select the drop-down list for Default Expansion For Rows and select TreeDescendantsR. The result is the reverse direction of the originating account on expansion for rows when using the double click:

	Jan 2018	Feb 2018
2000_100 - Third Party Sales	50,945,236	50,985,236
2000_200 - OEM Sales	4,606,157	4,536,127
2000_300 - Subassembly Sales	382,840	382,800
2000_400 - Parts Sales	2,048,108	2,040,108
60000 - Operating Sales	57,982,342	57,944,272
60100 - IC Sales	15,000	15,990
2200_100 - Third Party Returns & Allowances	728,117	727,974
2200_200 - OEM Returns & Allowances	37,372	37,312
2200_300 - Subassembly Returns & Allowances	95,952	95,052
2200_400 - Parts Returns & Allowances	980,454	980,054
60200 - Returns & Allowances	1,841,894	1,840,391
60250 - Other Outside Sales		
60999 - Net Sales	56,155,448	56,119,871
41000 - Operating Cost of Goods Sold	28,677,632	27,868,820
42000 - IC Cost of Goods Sold	15,000	15,000
43000 - Cost of Goods Sold	28,692,632	27,883,820
61000 - Gross Income	27,462,816	28,236,051

Time Member Expansions

These Member expansions can be added onto a regular time-based Member Script. They can be used in a Cube View or in the Form Template's Time Filter for 'Complete Form' option, so only the time periods needed will be displayed.

AllPriorInYear

This returns all the time periods before the specified time period excluding the specified time for its frequency. T#2012M6.AllPriorInYear, T#WF.AllPriorInYear return the previous periods in that year, but not the specified period.

AllPriorInYearInclusive

This returns all the time periods before the specified time period including the specified time for its frequency. T#2012M4.AllPriorInYearInclusive returns periods 2012M1 through 2012M4.

AllNextInYear

This returns all the time periods after the specified time period not including the specified time for its frequency. T#2012M8.AllNextInYear returns periods 2012M9 through 2012M12.

AllNextInYearInclusive

This returns all the time periods before the specified time period including the specified time for its frequency. T#2012M8.AllNextInYearInclusive returns periods 2012M8 through 2012M12.

Weeks

This returns all the weeks associated with the specified time filter. T#2017M2.Weeks returns all the weeks in M2. T#2017.Weeks returns all the weeks in 2017. If the POV Time is 2017M7, T#POV.Weeks returns all the weeks in M7.

Months

This returns all the months associated with the specified time filter. T#2017.Months returns all the months in 2017. If the POV Time is set to 2017Q2, T#POV.Months returns all the months in Q2.

MonthsInQuarter

This returns the months in the specified quarter. If the Workflow period is 2012M2, T#WF.MonthsInQuarter will return 2012M1, 2012M2, and 2012M3.

MonthsInHalfYear

This returns the months in the half year of the specified period. If the Global period is 2012M2, T#Global.MonthsInHalfYear will return 2012M1 through 2012 M6.

MonthsInYear

This returns all of the months in the year of the specified period. If the POV period is 2012M7, T#POV.MonthsInYear will return 2012M1 through 2012M12.

Quarters

This returns the quarters associated with the specified year. T#2017.Quarters will return 2017Q1, 2017Q2, 2017Q3, 2017Q4.

QuartersInHalfYear
This returns the quarters in the half year of the specified period. If the Workflow period is 2012M3, T#WF.QuartersInHalfYear will return 2012Q1 and 2012Q2.

QuartersInYear
This returns all of the quarters in the year of the specified period. T#2012M7.QuartersInYear will return 2012Q1, 2012Q2, 2012Q3 and 2012Q4.

HalfYears
This returns the half years associated with the specified year. T#2017.HalfYears returns 2017H1 and 2017H2.

HalfYearsInYear
This returns the half years in the year of the specified period. If the Global period is 2012M5, T#Global.HalfYearsInYear will return 2012H1 and 2012H2.

Prior 1-Prior 12
This returns the prior period(s) in relation to the specified period. T#2010M12.Prior12 will return the 12 months prior to 2010M12 not including 2010M12. T#2017W40.Prior12 will return the 12 weeks prior to 2017W40 not including 2017W40.

Next 1-Next 12
This returns the next period(s) in relation to the specified period. T#2010M12.Next12 will return the 12 months after 2010M12 not including 2010M12. T#2017W40.Next12 will return the 12 weeks after 2017W40 not including 2017W40.

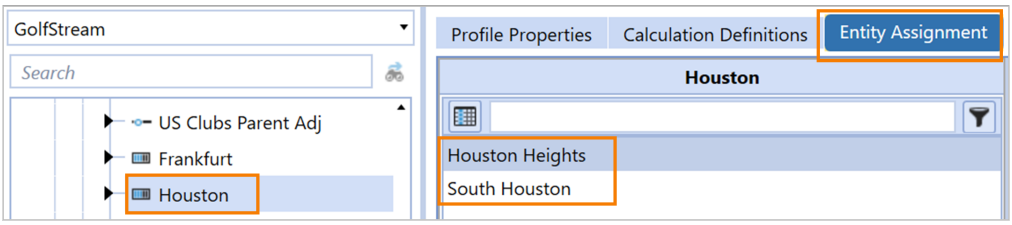
Workflow Member Expansions

These are used in Cube Views that are used in Reports, Forms or Dashboards and presented during the Workflow process. These Method Queries can also be used when setting pop-up Parameters when Dashboards and Reports are run:

WFProfileEntities (Entity Dimensions Only)

This returns all Entities associated with this Workflow Profile.

In the below example, the Houston Workflow Profile is assigned the Houston Heights and South Houston Entity members. If the Workflow POV is set to Houston, then E#Root.WFProfileEntities returns Houston Heights and South Houston.



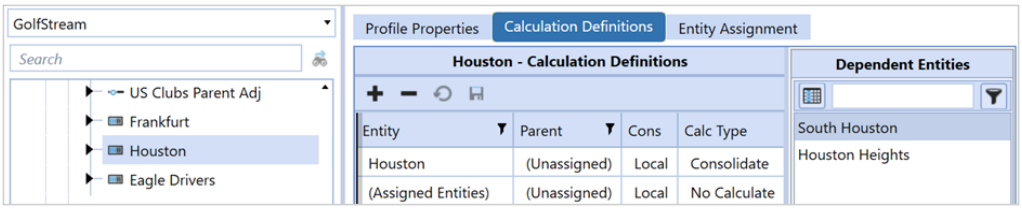
WFCalculationEntities (Entity Dimension Only)

This filters to the Entities encompassed within the Workflow Profile Calculation Filter, which are calculated, translated, and consolidated when the user clicks Process during Workflow.

When used as a member expansion, this expression returns all Entities defined as part of the Calculation Definitions for the selected Workflow Unit.

In the below example, the Houston Workflow Profile defines Houston and the Assigned Entities, South Houston and Houston Heights, within Calculation Definitions.

E#Root.WFCalculationEntities returns Houston, South Houston, and Houston Heights.

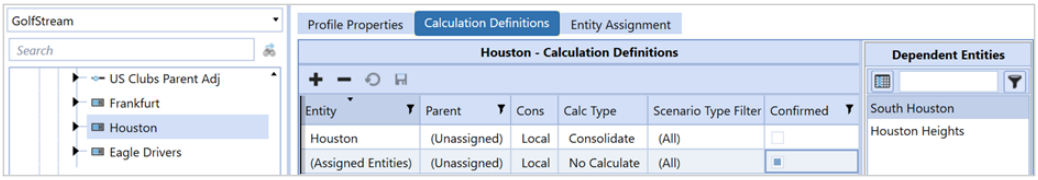


WFConfirmationEntities (Entity Dimensions Only)

This filters to the Entities encompassed within the Workflow Profile Confirmation Filter, which are checked when the user clicks Confirm during Workflow.

In the below example, the Houston Workflow Profile has the Confirmed Filter selected for the Assigned Entities, South Houston and Houston Heights, within the Calculation Definitions.

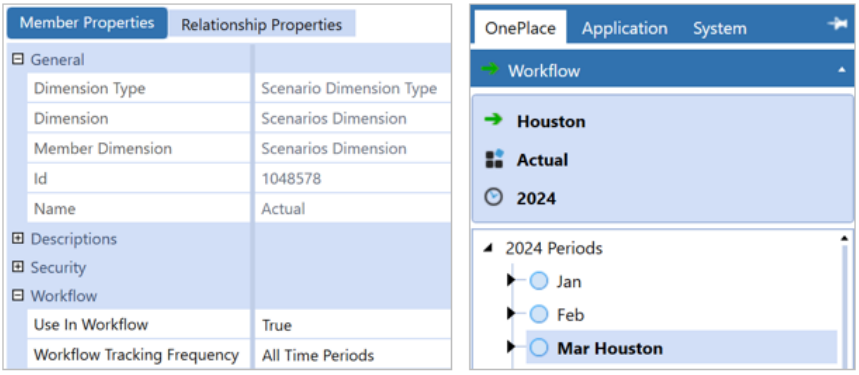
E#Root.WFConfirmationEntities returns South Houston and Houston Heights.



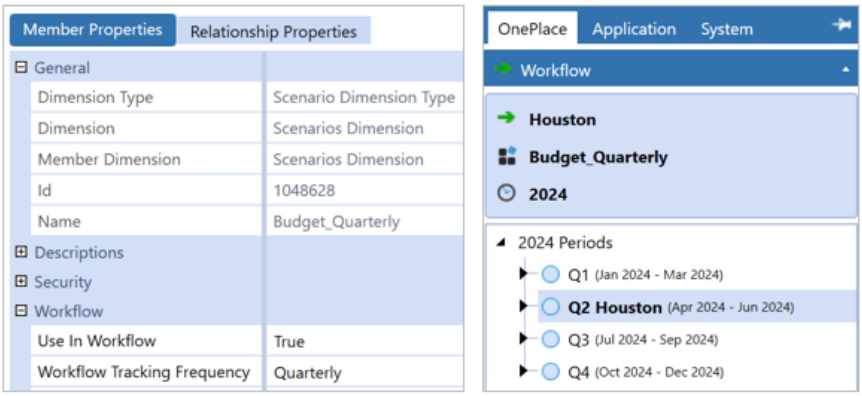
WFTimePeriods

This returns the time period(s) associated with the Workflow Profile. When using the Standard Workflow Tracking Frequency, it returns the single time period associated with the selected Workflow Unit. When using other Workflow Tracking Frequencies, it returns the range of time periods for the selected Workflow Unit between WFStartTime and WFEndTime.

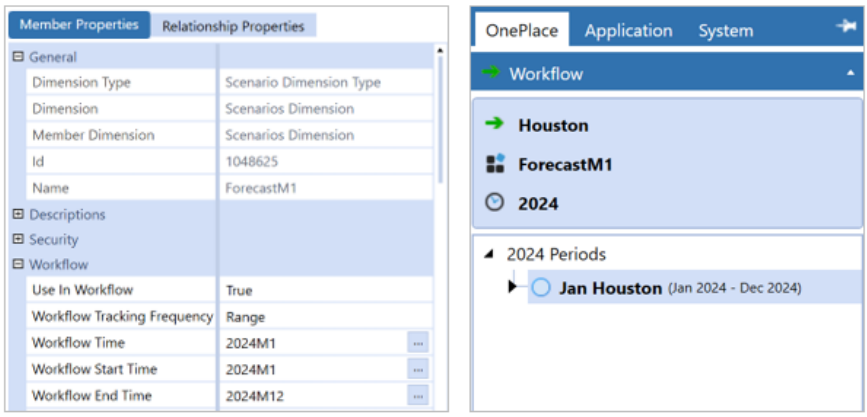
In the below example, the Actual scenario member has a Workflow Tracking Frequency of All Time Periods in a monthly application. If the Workflow POV is Actual and 2024M3 then T#Root.WFTimePeriods returns 2024M3.



In the below example, the Budget_Quarterly scenario member has a Workflow Tracking Frequency of Quarterly. If the Workflow POV is Actual and 2024Q2 then T#Root.WFTimePeriods returns 2024M4, 2024M5, and 2024M6.



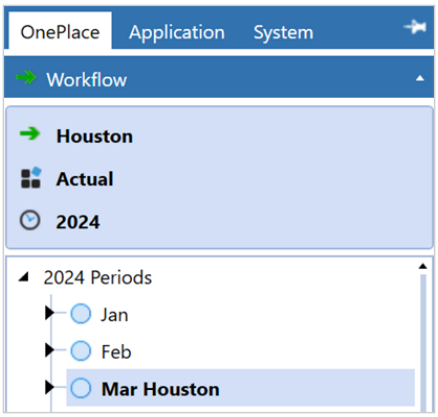
In the below example, the ForecastM1 scenario member has a Workflow Tracking Frequency of Range, WFStartTime of 2024M1, and WFEndTime of 2024M12. If the Workflow POV is ForecastM1 and 2024M1, then T#Root.WFTimePeriods returns 12 time members, 2024M1 through 2024M12.



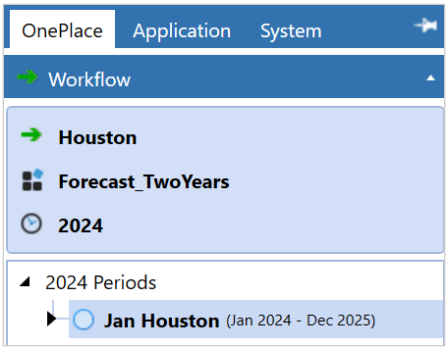
WFCalculationTimePeriods

This returns the time periods needed to be calculated for a Workflow Unit. For example, if a range-based Workflow Unit spans two years, it would return the last period of the first year and the last period of the second year. Executing a calculation using those two time periods would cause all 24 time periods to be calculated because the calculation engine automatically calculates all prior periods in a year. WFCalculationTimePeriods is intended to be used in Data Management Steps and Business Rules.

In the below example, the Actual scenario member has calculations monthly. If the Workflow POV is Actual and 2024M3, then T#Root.WFCalculationTimePeriods returns 2024M3.



In the below example, the Forecast_TwoYears scenario member has a range Workflow Unit that spans two years from 2024M1 to 2025M12. If the Workflow POV is Forecast_TwoYears, then T#Root.WFTimePeriods returns 2024M12 and 2025M12 as calculations are needed for each year.



WFChannelMembers (User Defined Dimensions Only)

This returns the UD Dimension Members associated with the Workflow Channel that is associated with this Workflow Profile. This can be a one-to-many relationship. For example, UD1 represents Cost Centers for this application and three UD1 Members are associated with the Workflow Channel called Engineering. If the Workflow Profile in the Workflow View is associated with the Workflow Channel of Engineering, UD1#Root.WFChannelMembers will return those three UD1 Members.

Member Expansion Where Clause

The Where Clause adds an additional level of filtering after the list of Members has been retrieved. See **Member Expansions > Where** for more information.

Below are examples of Where clauses filtering for common member properties, however, you can filter on additional member properties that are not shown below:

Name (e.g., (Name StartsWith ‘Sales’))
Filter for members based on their Name property.
A#Sales.Descendants.Where(Name Contains '6000') filters the Member list to include only Member names containing ‘6000.’ This could also be written as:

A#Sales.Descendants(Name StartsWith 6000)

The following examples are based on if an entity hierarchy was the following:

- Total GolfStream
    - Clubs
        - NA Clubs
        - Frankfurt
    - Golf Balls
        - Europe Golf Balls
        - NA Golf Balls
    - Accessories & Apparel
    - Course Management
    - Corporate

E#[Total GolfStream].Children.Where((Name Contains 'Balls') OR (Name = 'Clubs'))

This returns the children of Total GolfStream if the member name contains Balls or equals Clubs, which is:

- Clubs
- Golf Balls

Other Member Expansions can also be added after Where Clause expansions.

E#[Total GolfStream].Member.Children.Where((Name Contains Balls) OR (Name = Clubs)).Children

This returns Total GolfStream due to E#[Total GolfStream].Member. It also returns its children member where Name contains Balls or equals Clubs, and those member's children, which is:

- Total GolfStream
    - Clubs
        - NA Clubs
        - Frankfurt
    - Golf Balls
        - Europe Golf Balls
        - NA Golf Balls

Description (e.g., (Description Contains 'Cost'))
Filter for members based on their Description property.
A#Root.Descendants.Where(Description Contains 'Cost') filters the Member list to include only accounts with Descriptions containing 'Cost.'

MemberDim
Filter for members within certain dimensions.
A#MyAccount.Base.Where(MemberDim = DimensionName)
A#[Income Statement].Where(MemberDim = |WFAccountDim|) filters the member list to the accounts within the workflow's Account Dimension.
U2#Top.Descendants.Where(MemberDim = 'HoustonProducts') filters the Member list to the products in the HoustonProducts UD2 Dimension.

HasChildren (e.g., (HasChildren = True))
Filter for members with or without children.
A#Root.Children.Where(HasChildren = False) filters the Member list to include only the members without children.

A#Root.Children.Where(HasChildren = True) filters the Member list to include only the members with children.

InUse
Filter for Entity, Account, Flow and UD1-UD8 members in use or not in use.
A#Root.Children(InUse = True) filters the results to only include the members with the In Use member property set to True.

AccountType
Filter Account members based on their Account Type.
A#Root.Base(AccountType = Revenue) filters the results to only the Members with the Account Type of Revenue.

Currency

Filter for entity members of a specific currency.

E#[Total GolfStream].Descendants.Where(Currency = EUR) filters the results to only include descendants of Total GolfStream with the currency of EUR.

IsIC

Filter for intercompany members based on the Is IC Entity member property and the Is IC Account member property.

E#[Total GolfStream].Descendants.Where(IsIC = True) filters the results to only include descendants of the Total GolfStream with the Is IC Entity member property set to True.

A#Sales.Descendants.Where(IsIC = True) filters the results to only include descendants of Sales with the Is IC Account member property set to True.

Formula

Filter for Scenario, Account, Flow, and UD1-UD8 members with Member Formulas. F#Root.TreeDescendants(Formula <> '') filters out Members with an empty value for the Formula property so that only members with Member Formulas show.

User In Security Group Where Clauses

These **Where Clauses** will restrict the list to only Members the user is allowed to read and write to based on the user security settings. OR expressions can be used to refer to multiple properties:

E#[Total GolfStream].Base.where((UserInReadDataGroup = True) or (UserInReadWriteDataGroup = True))

Choices:

UserInReadDataGroup

UserInReadDataGroup2

UserInReadWriteDataGroup

UserInReadWriteDataGroup2

UserInAnyDataSecurityGroup

TextN (Text1 through Text8)

This uses the TextN property in the Scenario, Entity, Account, Flow, and UD1 to UD8 Dimensions. There are eight TextN properties in these types of Members that can be used for any need. These Where Clauses will query those fields for their values:

A#Root.Children(Text1 = 'Blue')

A#Root.Children(Text2 StartsWith 'Strategic')

A#Root.Children(Text3 Contains 'Tax')

A#Root.Children(Text4 EndsWith 'Old')

A#Root.Children(Text5 <> '') filters out the Members that have an empty value for the Text5 property.

For more Where Clause examples, refer to the Samples Tab in the Member Filter Builder.

Time Functions

Time Functions and Pov Member References

When using the POV as a point of reference, use the following constants in the Member Scripts. In the constants below, N refers to the number of periods desired.

T#POV (All Dimension Types)
Select Members based on their literal or relative position to the selected main Point of View (POV) or the Cube View POV. The main POV settings are available under the OnePlace module. Call out a Dimension in order to pull its Member value from what is selected in the POV in this way: E#POV, C#POV, T#POV, V#POV, A#POV, and all other dimension types.

If a user's Cube POV is 2024M1, then a Cube View with T#POV returns 2024M1.

If the POV is 2024, T#POV returns 2024.

T#PovYear (Uses Time Member Suffix, e.g., T#PovYearM1)
This returns a specific time period within a current POV year defined by the currently selected POV period. If POV is 2024M1:

- T#PovYear returns 2024
- T#PovYearM3 returns 2024M3

You can combine T#PovYear with expansions. If the POV is 2024M3:

- T#PovYear.HalfYears returns the 2024H1 and 2024H2
- T#PovYear.Quarters returns all quarters for 2024
- T#PovYear.Months returns all months for 2024
- T#PovYear.Weeks returns all weeks for 2024 in weekly applications

T#PovPriorYear (Uses Time Member Suffix, e.g., T#PovPriorYearM12)

This returns a specific time period within the prior year defined by the currently selected POV period. If POV is 2024M1:

- T#PovPriorYear returns 2023
- T#PovPriorYearm12 returns 2023M12

T#PovPriorinYear1 (Uses Integer Suffix, e.g., T#PovPriorInYear1)

This returns the relative prior period from the currently selected POV period, but within the same year. If the current period is 2024M8:

- T#PovPriorInYear4 returns 2024M4
- T#PovPriorInYear10 returns the first period in the year, 2024M1, as there are not 10 periods prior to 2024M8 within the same year

T#PovPrior1 (Uses Integer Suffix, e.g. T#PovPrior1)

This returns the relative prior period from the currently selected POV period. If the POV is 2024M1, T#PovPrior3 returns 2023M10.

T#PovNextInYear1 (Uses Integer Suffix, e.g., T#PovNextInYear2)

This returns the relative next period in the future from the currently selected POV period, but within the same year. If the current period is 2024M4:

- T#PovNextInYear5 returns 2024M9
- T#PovNextInYear10 returns the last period in the year 2024M12, as there are not 10 periods after 2024M4 in within the same year.

T#PovNext1 (Uses Integer Suffix, e.g., T#PovNext1)

This returns the relative next period from the currently selected POV period. If the POV is 2024M8, T#PovNext5 returns 2025M1.

T#PovFirstInYear

This returns the first period in the year portion of the POV period. It is the same as T#POVYearM1, T#POVYearQ1, or, T#POVYearH1 based on the Input Frequency setting of the given Scenario. T#M1, T#Q1, or, T#H1 work the same way as well.

- If the POV is 2024M2, T#PovFirstInYear returns 2024M1.
- If the POV is 2024Q2, T#PovFirstInYear returns 2024Q1.

T#PovLastInYear

This returns the last period in the year portion of the POV period. If the current period is 2024M1, T#PovLastInYear returns 2024M12.

T#POVFirstInMonthPrior1

POV Month Example

This refers to the POV month and returns N months prior. If the POV is 2024M10, and T#POVFirstInMonthPrior5 is requested, 2024M5 is returned.

POV Week Example

This refers to the POV week and goes back N weeks from the first week in that month. This varies by calendar (445, 454, 544).

If the POV is 2024W38 (M9) in a 52 week 445 calendar, T#PovFirstInMonthPrior5 returns 2024W30. This example's result is driven by the first week in M9, which is 2024W35, and then returns five periods prior to 2024W30.

T#PovFirstInQuarterPrior1 (Uses Integer Suffix, e.g., T#PovFirstInQuarterPrior1)
This refers to the POV sub-period's quarter and goes back N periods from the first period in that quarter. If the period is 2024M5, T#PovFirstInQuarterPrior2 returns 2024M2. This example's result is driven by the first period in the requested quarter, which is 2024M4, and then returns two periods prior to 2024M2.

T#PovFirstInHalfYearPrior1 (Uses Integer Suffix, e.g., T#Pov FirstInHalfYearPrior2)
This refers to the POV sub-period's half year and goes back N periods from the first period in that half year.

If the period is 2024M5, T#PovFirstInHalfYearPrior2 returns 2023M11. This example's result is driven by the first period in the half year requested, which is 2024M1, and then returns two periods prior to 2023M11.

T#PovFirstInYearPrior1 (Uses Integer Suffix, e.g., T#Pov FirstInYearPrior3)
This refers to the POV year and goes back N periods from the first period.

If the period is 2024M5, T#PovFirstInYearPrior1 returns 2023M12. This example's result is driven by the first period in the year requested, which is 2024M1, and then returns one period prior to 2023M12.

T#PovLastInYearPrior1 (Uses Integer Suffix, e.g., T#PovLastInYearPrior4)
This refers to the POV year and goes back N periods from the last period.

If the period is 2024M5, T#PovLastInYearPrior1 returns 2024M11. This examples result is driven by the last period in the year requested, which is 2024M12 and then returns one period prior from this period to 2024M11.

Time Functions and Workflow References

T#WF (e.g., T#WF or S#WF)
This refers to the time period or Scenario currently selected in the Workflow module. Examples are T#WF and S#WF.

- If the Workflow POV is 2024M1, T#WF returns 2024M1.
- If the Workflow POV is Actual, S#WF returns Actual.

T#WFYear (Uses Time Member Suffix, e.g., T#WFYearM1)

This returns a specific time period within the current Workflow year. If the Workflow period is 2024M1:

- T#WFYear returns 2024
- T#WFYearM3 returns 2024M3

T#WFPriorInYear1 (Uses Integer Suffix, e.g., T#WFPriorInYear1)

This returns the relative prior period from the currently selected Workflow period, but within the same year. If the current Workflow period is 2024M8:

- T#WFPriorInYear4 returns 2024M4
- T#WFPriorInYear10 returns the first period in the year, 2024M1, as there are not 10 periods prior to 2024M8 within the same year

T#WFPrior1 (Uses Integer Suffix, e.g., T#WFPrior2)

This returns the relative prior period to the current Workflow period. If the current Workflow period is 2024M1, T#WFPrior3 returns 2023M10.

T#WFNextInYear1 (Uses Integer Suffix, e.g., T#WFNextInYear1)

This returns the relative next period in the future from the currently selected Workflow period, but within the same year. If the current Workflow period is 2024M4:

- T#WFNextInYear5 returns 2024M9
- T#WFNextInYear10 returns the last period in the year, 2024M12, as there are not 10 periods after 2024M4 within the same year.

T#WFNext1 (Uses Integer Suffix, e.g., T#WFNext2)
This returns the relative next period to the current Workflow period. If the current Workflow period is 2024M8, T#WFNext5 returns 2025M1.

T#WFFirstInYear
This returns the first period in the year portion of the Workflow period. This is the same as T#WFYearM1, T#WFYearQ1, or T#WFYearH1 based on the Input Frequency settings of the given Scenario.

- If the current Workflow period is 2024M2, T#WFFirstInYear returns 2024M1.
- If the current Workflow period is 2024Q2, T#WFFirstInYear returns 2024Q1.

T#WFLastInYear
This returns the last period in the year portion of the Workflow period. If the current Workflow period is 2024M1, T#WFLastInYear returns 2024M12.

Time Functions and Global References

T#Global (e.g., T#Global or S#Global)
The Application has a Global Time and Global Scenario setting found under the Application Tab > Tools > Application Properties. These settings can be referenced, for example T#Global and S#Global. For example, T#Global can be referenced in a Cube View's columns where everything before T#Global appears as the Actual scenario, and everything after T#Global appears as the Forecast scenario.

If the Global POV is 2024M1, T#Global returns 2024M1.

If the Global POV is Actual, S#Global returns Actual.

T#GlobalYear (Uses Time Member Suffix, e.g., T#GlobalYearM1)

This returns a specific time period within the current Global year. If the Global period is equal to 2024M1:

- T#GlobalYear returns 2024
- T#GlobalYearM3 returns 2024M3

T#GlobalPriorInYear1 (Uses Integer Suffix, e.g., T#GlobalPriorInYear1)

This returns the relative prior period from the currently selected Global period, but within the same year. If the current Global POV period is 2024M8:

- T#GlobalPriorInYear4 returns 2024M4
- T#GlobalPriorInYear10 returns the first period in the year, 2024M1, as there are not 10 periods prior to 2024M8 within the same year.

T#GlobalPrior1 (Uses Integer Suffix, e.g., T#GlobalPrior3)

This returns the relative prior period to the current Global period. If the current Global period is 2024M1, T#GlobalPrior3 returns 2023M10.

T#GlobalNextInYear1 (Uses Integer Suffix, e.g., T#GlobalNextInYear5)

This returns the relative next period in the future from the currently selected Global POV period, but within the same year. If the current Global POV period is 2024M4:

- T#GlobalNextInYear5 returns 2024M9
- T#GlobalNextInYear10 returns the last period in the year, 2024M12, as there are not 10 periods after 2024M4 within the same year.

T#GlobalNext1 (Uses Integer Suffix, e.g., T#GlobalNext5)
This returns the relative next period to the current Global period. If the current Global period is 2024M8, T#GlobalNext5 returns 2025M1.

T#GlobalFirstInYear
This returns the first period in the year portion of the Global period. This is the same as T#GlobalYearM1, T#GlobalYearQ1, or T#GlobalYearH1 based on the Input Frequency setting of the given Scenario.

- If the current Global period is 2024M2, T#GlobalFirstInYear returns 2024M1.
- If the current Global period is 2024Q1, T#GlobalFirstInYear returns 2024Q1.

T#GlobalLastInYear
This returns the last period in the year portion of the Global period. If the current Global period is 2024M1, T#GlobalLastInYear returns 2024M12.

For more Time Function examples, refer to the Samples Tab in the Member Filter Builder.

Time Functions and General References

Time functions, such as T#Year(|!MyTimeParam!|), return values based on what is within the parentheses, such as a parameters, substitution variables, and time members. For example, a user can select a time member from a Parameter prompt, such as 2024M6.

With these time functions, reference the year separate from the sub-period wherever T# is used. The following examples use parameters and substitution variables.

This example uses a Parameter named MyTimeParam.

The Cube View can use the selection like this:

- T#Year(|!MyTimeParam!|)Period(|!MyTimeParam!|)

The Year function returns the specified year, and the Period function returns the period without the year. The result is a valid time member because they are combined.

If the user selected 2024M6, it looks like this after the Parameter substitution:

- T#Year(2024M6)Period(2024M6)

The result used in the Cube View is T#2024M6.

This example shows last year's values for the same month. If the user selects 2024M6:

- T#YearPrior1(|!MyTimeParam!|)Period(|!MyTimeParam!|) returns T#2023M6.

This example shows last month's values for the same year. If the user selects 2024M6:

- T#Year(|!MyTimeParam!|)PeriodPrior1(|!MyTimeParam!|) returns T#2024M5.

This example shows all base members for the year using a Member Expansion:

- T#Year(|!MyTimeParam!|).Base

These examples look at the user's POV using a substitution variable. If the user's POV is 2024M1:

- T#Year(|POVTime|)PeriodNext1(|POVTime|) returns 2024M2.
- T#YearPrior2(|POVTime|)Period(|POVTime|) returns 2022M1.
- T#YearNext2(|POVTime|)Period(|POVTime|) returns 2026M1.
- T#Year(|POVTime|)Quarter(|POVTime|).MonthsInQuarter returns 2024M1, 2024M2, and 2024M3.

This example looks at the Workflow POV Time member and then counts back two years prior, such as when viewing prior year's YTD Net Income Values. If the Workflow time is 2024M1, T#YearPrior2(|WFTime|) returns 2022.

This example looks at the Global POV Time member and then counts forward 3 years. If the Global Time is 2024M3, T#YearNext3(|GlobalTime|) returns 2027.

Here are additional examples for each General Time Function, which return values based on the parameters, substitution variables, and time members within the parentheses.

T#Year():

This returns the year. If a period is specified after, it returns the year and then the specified period. If the Year section is skipped, the year comes from the Period section. Below are examples:

T#Year(|!TimeParam!|)Period(|!TimeParam!|)

- T#Year(2024M6) returns 2024.
- T#Year(2024M6)Period(2024M6) returns 2024M6.
- T#Year(2024M6)M2 returns 2024M2.
- T#Year(2024M6)PeriodPrior3(2024M6) returns 2024M3.
- T#Year(2024M6)PeriodPrior8(2024M6) returns 2023M10. PeriodPrior8 caused the year to change.
- T#Year(2024M6)PeriodPriorInYear8(2024M6) returns 2024M1. PeriodPriorInYear8 does not change the year, and returns M1 as there are not 8 periods prior to M6 in the year.

T#YearPrior1():

This returns a prior year based on the number in the function. If a period is specified after, it returns the prior year and then the specified period.

- T#YearPrior2(2024M6) returns 2022.
- T#YearPrior1(2024M6)Period(2024M6) returns 2023M6.
- T#YearPrior2(2024M6)Period(2024M6) returns 2022M6.

T#YearNext1():

This returns a future year based on the number in the function. If a period is specified after, it returns the future year and then the specified period.

- T#YearNext2(2024M6) returns 2026.
- T#YearNext1(2024M6)Period(2024M6) returns 2025M6.

T#Period():

This returns the period and year. The year comes from the period section if the Year() section is skipped.

- T#Period(2024M6) returns 2024M6.

T#PeriodPrior1():

This returns the relative prior period based on the number in the function. The year comes from the relative prior period.

T#PeriodPrior3(|WFTime|)

- T#PeriodPrior3(2024M6) returns 2024M3.
- T#PeriodPrior8(2024M6) returns 2023M10.

T#PeriodNext1():

This returns the relative future period based on the number in the function. The year comes from the relative future period.

T#PeriodNext2(|GlobalTime|)

- T#PeriodNext2(2024M6) returns 2024M8.
- T#PeriodNext8(2024M6) returns 2025M2.

T#PeriodPriorInYear1():

This returns the relative prior period within the same year based on the number in the function. The year comes from the period section.

- T#PeriodPriorInYear2(2024M6) returns 2024M4.
- T#PeriodPriorInYear8(2024M6) returns 2024M1. This returns the first period in the year, M1, as there are not 8 periods prior to M6 in the same year.

T#PeriodNextInYear1():

This returns the relative future period within the same year based on the number in the function. The year comes from the period section.

T#PeriodNextInYear1(|GlobalTime|)

- T#PeriodNextInYear1(2024M6) returns 2024M7.
- T#PeriodNextInYear8(2024M6) returns 2024M12. This returns the last period in the year, M12, as there are not 8 periods after M6 in the same year.

T#FirstPeriodInQuarter():

This returns the first period in the quarter. The year comes from the period section.

T#FirstPeriodInQuarter(|WFTime|)

- T#FirstPeriodInQuarter(2024M6) returns 2024M4.
- T#FirstPeriodInQuarter(2024M2) returns 2024M1.

T#FirstPeriodInQuarterPrior1():

This looks at the first period in the quarter, and then it returns the relative prior period based on the number in the function. The year comes from the relative prior period.

- T#FirstPeriodInQuarterPrior1(2024M6) returns 2024M3. The first period in the quarter is 2024M4, and then 1 period prior is M3.
- T#FirstPeriodInQuarterPrior4(2024M6) returns 2023M12. The first period in the quarter is 2024M4, and then 4 periods prior is 2023M12.

T#FirstPeriodInQuarterNext1():

This looks at the first period in the quarter, and then it returns the relative future period based on the number in the function. The year comes from the relative future period.

T#FirstPeriodInQuarterNext2(|POVTime|)

- T#FirstPeriodInQuarterNext2(2024M6) returns 2024M6. The first period in the quarter is 2024M4, and then 2 periods in the future is M6.
- T#FirstPeriodInQuarterNext9(2024M6) returns 2025M1. The first period in the quarter is 2024M4, and then 9 periods in the future is 2025M1.

T#FirstPeriodInHalfYear():

This returns the first period in the half year. The year comes from the period section.

T#FirstPeriodInHalfYear(|WFTime|)

- T#FirstPeriodInHalfYear(2024M6) returns 2024M1.

T#FirstPeriodInHalfYearPrior1():

This looks at the first period in the half year. Then it returns the relative prior period based on the number in the function. The year comes from the relative prior period.

- T#FirstPeriodInHalfYearPrior1(2024M6) returns 2023M12. The first period in the half year is 2024M1, and then 1 period prior is 2023M12.

T#FirstPeriodInHalfYearNext1():

This looks at the first period in the half year, and then it returns the relative future period based on the number in the function. The year comes from the relative future period.

T#FirstPeriodInHalfYearNext2(|WFTime|)

- T#FirstPeriodInHalfYearNext2(2024M6) returns 2024M3. The first period in the half year is 2024M1, and then 2 periods in the future is M3.

T#FirstPeriodInYear():

This returns the first period in the year. The year comes from the period section.

T#FirstPeriodInYear(|WFTime|)

- T#FirstPeriodInYear(2024M6) returns 2024M1.

T#FirstPeriodInYearPrior1():

This looks at the first period in the year and then it returns the relative prior period based on the number in the function. The year comes from the relative prior period.

T#FirstPeriodInYearPrior1(|!TimeParam!|)

- T#FirstPeriodInYearPrior1(2024M6) returns 2023M12. The first period in the year is 2024M1, and then 1 period prior is 2023M12.

T#FirstPeriodInYearNext1():

This looks at the first period in the year, and then it returns the relative future period based on the number in the function. The year comes from the relative future period.

T#FirstPeriodInYearNext2(|PovTime|)

- T#FirstPeriodInYearNext2(2024M6) returns 2024M3. The first period in the half year is 2024M1, and then 2 periods in the future is M3.

T#Quarter():

This returns the quarter. The year comes from the period section.

T#Quarter(|WFTime|).

- T#Quarter(2024M6) returns 2024Q2.
- T#Quarter(2024M1) returns 2024Q1.

T#QuarterPrior1():

This returns the relative prior quarter based on the number in the function. The year comes from the relative prior quarter.

- T#QuarterPrior1(2024M6) returns 2024Q1.
- T#QuarterPrior3(2024M6) returns 2023Q3.

T#QuarterNext1():

This returns the relative future quarter based on the number in the function. The year comes from the relative future quarter.

T#QuarterNext2(|POVTime|).

- T#QuarterNext2(2024M6) returns 2024Q4.
- T#QuarterNext3(2024M6) returns 2025Q1.

Weekly Functions

The following Time Functions apply to weekly applications. The following examples are based on a 52 Week 445 calendar.

T#FirstPeriodInMonth

If a weekly period is used in the function, this returns the first week in the month. The year comes from the period section.

- T#FirstPeriodInMonth(2024W43) returns 2024W40.

T#FirstPeriodInMonthPrior1()

If a weekly period is used in the function, this looks at the first weekly period in the month. It then returns the relative prior weekly period based on the number in the function. The year comes from the relative prior period.

T#FirstPeriodInMonthPrior3(|WFTime|).

- T#FirstPeriodInMonthPrior3(2024W5) returns 2024W2.
- T#FirstPeriodInMonthPrior3(2024W1) returns 2023W50.

T#FirstPeriodInMonthNext1()

If a weekly period is used in the function, this looks at the first weekly period in the month. It then returns the relative future weekly period based on the number in the function. The year comes from the relative future period.

- T#FirstPeriodInMonthNext2(2024W49) returns 2024W50. The first weekly period in the month is 2024W48, and then 2 periods after is 2024W50.
- T#FirstPeriodInMonthNext2(2024W52) returns 2024W50. The first weekly period in the month is 2024W48, and then 2 periods after is 2024W50.

T#Month()

This returns the month. The year comes from the period section.

- T#Month(2024W40) returns 2024M10.

T#MonthPrior1()

This returns the relative prior month based on the number in the function. The year comes from the relative prior month.

- T#MonthPrior1(2024W5) returns 2024M1.
- T#MonthPrior2(2024W5) returns 2023M12.

T#MonthNext1()

This returns the relative future month based on the number in the function. The year comes from the relative future month.

T#MonthNext5(|POVTime|).

- T#MonthNext5(2024W9) returns 2024M8.
- T#MonthNext2(2024W52) returns 2025M2.

Changing the Time Label in Headers When Using These Time Functions
When using this Time function to change a header label in a Cube View, use the :Name("") function and substitute what is seen.

The following example shows what a Time function returns in the header of a Cube View without and with a :Name("") function. The Time function below returns the last period of the prior year using the Global Time period. For this example, the Global Time is 2024M6 and the |MFYear| substitution variable is the Member Filter's year:

T#YearPrior1(|GlobalTime|)M12 returns 2023M12 - Dec 2023 (dependent on the Cube View Header Text properties).

T#YearPrior1(|GlobalTime|)M12:Name("Year End |MFYear|") returns Year End 2023.

Below are examples of what similar substitution variables return if Global Time is 2024M6:

- T#YearPrior1(|GlobalTime|)M12:Name("Year End |MFTime|") returns Year End 2023M12.
- T#YearPrior1(|GlobalTime|)M12:Name("Year End |MFTimeDesc|") returns Year End Dec 2023.
- T#YearPrior1(|GlobalTime|)M12:Name("Year End |MFTimeShortDesc|") returns Year End Dec.
- T#YearPrior1(|GlobalTime|)M12:Name("Year End |MFSubPeriod|") returns Year End M12.

Substitution Variables

Substitution Variables can be used to exchange values at run time for Parameters presented to the user before a Dashboard runs, for Members in the POV, for Global variables for Scenario or Time, or for Workflow variables for Time, Scenario or other Workflow Profile attributes.

Substitution Variables from Current POV

These settings will return the values from the user's current POV for each Dimension.

By adding Desc to the end of many of these substitution variables, the Description for the Member will be returned instead of the Member name:

|PovEntityDesc| will return Akron instead of the Entity name 123.

POVCube and POVCubeDesc

POVEntity and POVEntityDesc

POVParent and POVParentDesc

POVConsolidation and POVConsolidationDesc

POVScenario and POVScenarioDesc

POVTime, POVTimeDesc and POVTimeShortDesc

POVView and POVViewDesc

POVAccount and POVAccountDesc

POVFlow and POVFlowDesc

POVOrigin and POVOriginDesc

POVIC and POVICDesc

POVUD1-UD8 and POVUD1-8Desc

> **NOTE:** If there is an index out-of-range, the Cube View will display an error showing the data POV as invalid. When the Cube View is passed to the Pivot Grid control, it attempts to resolve the error column. It cannot, so in the case of the Time dimension it will instead display the default time member as "Name - Description". To resolve the issue in the Pivot Grid, you must first correct the issue in the source Cube View.

Substitution Periods are defined within a given Year. The number of sub periods are limited to the number of months or weeks within the given application year. For example, if you go prior or next within a given year, it will stop at the year boundary. If PovSubPeriodPrior14 in a given monthly app for a given year will not go past M1. PovSubPeriodNext will not go past M12. In a weekly app, PovSubPeriodPrior will not go past W1 and PovSubPeriodNext will not go past W53.

POVTimeDimProfile
This returns the Time Dimension Profile name associated with the user's current Cube POV.

POVCurrency
This returns the actual currency for an Entity based on the Consolidation Dimension setting in the POV. For example, if Consolidation is set to Local for the Manchester Entity, GBP will return, which is its currency. If Consolidation is set to Translated and the Parent is set to US, USD will return.

POVTimePriorInYearN
This returns the prior (or more) time period if it is still in the current year.

POVTimePriorN
This returns the prior (or more) time period.

POVTimeNextInYearN
This returns the next (or more) time period if it is still in the current year.

POVTimeNextN
This returns the next (or more) time period.

POVYear

This returns the Year portion of a POV Time period.

POVYearPriorN

This returns the Year portion of a Time period for the previous year where N is the number of years.

POVYearNextN

This returns the Year portion of a Time period for the upcoming year where N is the number of years.

POVSubPeriod

This returns the Week, Month, Quarter, or Half Year portion of a Time period. If the POV is 2012M5, M5 is the SubPeriod.

POVSubPeriodPriorN

This returns the previous Week, Month, Quarter, or Half Year portion of a Time period.

POVSubPeriodNextN

This returns the next Week, Month, Quarter, or Half Year portion of a Time period.

POVSubPeriodNum

This returns the sub period's number for the current POV, so if the POV is set to M3, the number 3 is returned.

POVTimeFirstInYear

This is the same as T#POVFirstInYear, T# W1, T#M1, T#Q1, or T#H1 based on the Input Frequency of the Scenario.

POVTimeLastInYear

This returns the last time period in the year based on the Input Frequency of the Scenario.

Workflow Substitution Variables

WF

This is used within the Point of View area for Scenario and Time: T#WF.

WFProfile

This provides the current Workflow Profile name.

WFProfileIndex

This provides a numeric value of the Workflow Profile in the hierarchy.

WFProfileLastDescendantIndex

This provides an index of bottom descendants in the Workflow Profile tree.

Workflow Profile Keys

These return a numeric internal ID for Workflow Profiles.

WFProfileKey

WFReviewProfileKeys

WFInputParentProfileKeys

WFImportProfileKeys

WFFormProfileKeys

WFJournalProfileKeys

WFScenario

This returns the Scenario in the Workflow View.

WFScenarioDesc

This returns a Description of the Scenario in the Workflow View.

WFScenarioID

This returns the Numeric ID of the Scenario in the Workflow View.

WFTime

This returns the Time period associated with current Workflow Unit.

WFTimeDesc and WFTimeShortDesc

This returns the Description (e.g. Feb 2011) or Short Description (e.g. Feb) for Workflow Time as defined in Time Profile.

WFTimeID

This returns the Numeric ID of the Workflow Unit Time period.

WFTimePriorInYearN

This returns the prior (or more) Workflow time period if it is still in the current year.

WFTimePriorN

This returns the prior (or more) time period.

WFTimeNextInYearN

This returns the next (or more) time period if it is still in the current year.

WFTimeNextN

This returns the next (or more) time period.

WFYear

This returns the year portion of the Workflow Unit.

WFYearPriorN

This returns the Year portion of a Time period for the previous year where N is the number of years.

WFYearNextN

This returns the Year portion of a Time period for the upcoming year where N is the number of years.

WFSubPeriod

This returns the Week, Month, Quarter, or Half Year portion of a Time period

WFSubPeriodPriorN

This returns the previous Week, Month, Quarter, or Half Year portion of a Time period.

WFSubPeriodNextN

This returns the next Week, Month, Quarter, or Half Year portion of a Time period.

WFSubPeriodNum

This returns the period's number for the current Workflow Period, so if the Workflow is set to M3, the number 3 is returned.

WFTimeFirstInYear

This is the same as T#WFFirstInYear, T#W1, T#M1, T#Q1, or T#H1 based on the Input Frequency of the Scenario.

WFTimeLastInYear

This returns the last time period in the year based on the Input Frequency of the Scenario.

WFStartTime

This is the Workflow Start Time entry for this Scenario.

WFStartTimeDesc

This provides a Description of the Workflow Start Time entry for this Scenario.

WFStartTimeShortDesc

This provides a Short Description of the Workflow Start Time entry for this Scenario.

WFEndTime

This is the Workflow End Time entry for this Scenario.

WFEndTimeDesc

This provides the Description of the Workflow Start Time entry for this Scenario.

WFEndTimeShortDesc

This provides the Short description of the Workflow Start Time entry for this Scenario.

WFCubeRoot

This returns the very top level of the Workflow Profile hierarchy.

WFCube

This returns the Cube related to this Workflow Profile.

WFTimeDimProfile

This returns the Time Dimension Profile name associated with the user's current Workflow POV.

WFEntityDim

The returns the Dimension in play for the Workflow Entities assigned to this Workflow Profile.

WFScenarioDim

This returns the Scenario Dimension for the Cube assigned to this Workflow Profile.

WFAccountDim

This returns the Account Dimension associated with the Cube for this Workflow Profile.

WFFlowDim

This returns the Flow Dimension for the Cube assigned to this Workflow Profile.

WFUD1Dim-WFUD8Dim

This returns the UD1 Dimension-UD8 Dimension for the Cube assigned to this Workflow Profile.

WFText1, WFText2, WFText3 and WFText4

This provides four optional text fields that can be populated and referenced from a Workflow Profile.

Global Substitution Variables

GlobalScenario

This is the Global Point of View Scenario from Application Properties.

GlobalTime

This is the Global Point of View Time period from Application Properties.

GlobalScenarioDesc

This provides the Descriptions for the Global Scenario.

GlobalTimeDesc and GlobalTimeShortDesc
This provides the Description (e.g. Feb 2011) or Short Description (e.g. Feb) for Global Time, as defined in the Time Profile.

Cube View Substitution Variables

These settings will return the values from the user's current Cube View in the Point of View Slider for each Dimension. For the Dimensions not specified in the Cube View's Point of View Slider, the Member will be represented by the user's POV settings.

By adding Desc to the end of many of these substitution variables, the Description for the Member will be returned instead of the Member name:

|CVAccountDesc| will return Net Sales instead of 61000

|CVTimeDesc| will return Dec 2011 instead of 2011M12

|CVTimeShortDesc| will return Dec instead of 2011M12

The following Substitution Variables will return the Cube View Name, Cube Name, or Dimension Name associated with the specific Cube View as well as a Description.

CVName and CVDesc

CVCube and CVCubeDesc

CVTimeDimProfile

CVEntity and CVEntityDesc

CVParent and CVParentDesc

CVConsolidation and CVConsolidationDesc

CVScenario and CVScenarioDesc

CVTime, CVTimeDesc and CVTimeShortDesc

CVView and CVViewDesc

CVAccount and CVAccountDesc

CVFlow and CVFlowDesc

CVOrigin and CVOriginDesc

CVIC and CVICDesc

CVUD1-8 and CVUD1-8Desc

CVCurrency

This returns the actual currency for an Entity based on setting in the specific Cube View.

Additional Cube View Time Functions

See Substitution Variables from Current POV to see examples of these Time Functions.

CVTimePriorInYear
CVTimePrior
CVTimeNextInYear
CVTimeNext
CVYear
CVYearPrior
CVYearNext
CVSubPeriod
CVSubPeriodPrior
CVSubPeriodNext
CVSubPeriodNum
CVTimeFirstInYear
CVTimeLastInYear

General Substitution Variables

The script or variable will use pipe characters to include a pre-defined substitution variable, e.g. |UserName|. These can be used in Business Rules, Cube Views, and Dashboard headers.

Null
|Null| is empty text. |Null| is mostly used in Cube-level security and the Stage parser. I t can also be used within a comma-separated list of Parameter values when you need to set a value to be an empty string. However, |Null| is not a commonly used substitution variable.

Space
This is used to replace a string with nothing or use a space along with the Name function in a Member Filter. For example, in order to make a Column Header or Row Header blank, use T#POV:Name(“ ”) or T#POV:Name(|space|). Both will produce the same result.

UserName
This provides the current user name

UserText1-4

This allows the reference of the Text1 through Text4 properties related to a User account: |UserText3|.

AppName

This provides the application name

DateTimeForFileName

This returns the current date and time: 20131208_102540.

DateForFileName

This returns the current date: 20131208.

DateLong

This returns the current date: Sunday, December 08, 2013.

DateMMDDYYYY

This returns the current Date as Month, Day, Year: 12/08/2013.

DateDDMMYYYY

This returns the current Date as Day, Month, Year: 08/12/2013.

DateYYYYMMDD

This returns the current Date as Year, Month, Day: 2013/12/08.

DateTimeHHMMSS

This returns the current Date/Time as Hour, Minutes, Seconds: 11:00:19.

DateTimeForFileNameUTC
DateForFileNameUTC
DateLongUTC
DateMMDDYYYYUTC
DateDDMMYYYYUTC
DateYYYYMMDDUTC
DateTimeHHMMSSUTC
The version of these functions with the UTC suffix returns the same result, but in Coordinated Universal Time.

Member Filter Substitution Variables

In order to change how a Member appears in results, use the MF functions below. See the section on Changing the Time Label in Headers When Using These Time Functions under Time Functions for several examples of how these are used. These are used in Name and GetDataCell functions only.

MFTime

MFTimeDesc

MFTimeShortDesc

MFYear

MFSubPeriod
MFSubPeriodNum

The following Substitution Variables work with the XFMemberProperty function in order to retrieve any Dimension Member Name being used within a Member Filter. Add Desc to any of these Substitution Variables in order to display the Member Description instead of the Member Name. See XFMemberProperty under Commonly Used Member Filter Functions for more details on using these Substitution Variables.

|MFCube|
|MFEntity|
|MFParent|
|MFConsolidation|
|MFScenario|
|MFTime|
|MFView|
|MFAccount|
|MFView|
|MFFlow|
|MFOrigin|
|MFIC|
|MFUD1|-|MFUD8|

Loop1-4Variable
Loop1-4DisplayVariable
Loop1-4Index
Variable1-10

See Presenting Data With Books, Cube Views and Dashboards.

Custom Substitution Variables

For user-defined (i.e., custom) Substitution Variables, the user will enclose the variable name using both pipes and exclamation points: |!myVariable!|

This excludes the Member Filter Substitution Variables listed above.

A good use of this is for Parameters added to a Dashboard. A Parameter called SalesDashboardEntity might be created and prompt the user with a list of all Entities to choose from before filtering a Report. In the Cube View that drives this Report, limit the Cube View's POV to the selected Parameter Entity by entering |!SalesDashboardEntity!|. When designing a Dashboard, |!SalesDashboardEntity!| can also be entered in the title of the Dashboard Component in order to be displayed at run time.

Samples of Member Filter Functions

The Samples tab in the Member Filter Builder contains examples of Member Filter Functions. Within the Member Filter Builder, hover over the samples with your cursor for a description and an example. Samples are explained below, or the relevant section is provided where you can find more information.

Samples of Member Expansions

See Member Expansions and Reverse Order Member Expansionsand the Samples tab in the Member Filter Builder for examples.

Samples of Where Clause Expressions

See Member Expansions and Member Expansion Where Clause and the Samples tab in the Member Filter Builder for examples.

Samples of Time Expressions

See Time Functions and the Samples tab in the Member Filter Builder for examples.

Samples of GetaDataCell Expressions

Use the GetDataCell function to retrieve specific cells, perform math, or Business Rule operations. The results are returned in a Cube View column or row. The column or row header comes from what is defined within :Name(), which can be changed.

Variance Using GetDataCell

This calculates the difference as a ratio between two members using: ((A-B)/Abs(B))

GetDataCell(Variance(S#Actual,S#BudgetV1)):Name(Variance)

This example returns the value of ((S#Actual - S#BudgetV1) / Abs(S#BudgetV1)) in a Cube View column or row named Variance.

Variance Percent Using GetDataCell

This calculates the difference as a percentage between two members using: ((A-B)/Abs(B)) * 100

GetDataCell(VariancePercent(S#Actual,S#BudgetV1)):Name(Var %)

This example returns the value of ((S#Actual - S#BudgetV1) / Abs(S#BudgetV1)) * 100 in a Cube View column or row named Var %.

Better or Worse Using GetDataCell

This calculates the difference between two members while considering the Account Type. For example:

- Revenue accounts are calculated using (A-B)
- Expense accounts are calculated using ((-1 * A) - (-1 * B))

GetDataCell(BWDiff(S#Actual, S#BudgetV1)):Name(BetterWorse Difference)

This example returns the value of (S#Actual - S#BudgetV1) for Revenue accounts and ((-1 *S#Actual) - (-1 * S#BudgetV1)) for Expense accounts in a Cube View column or row named BetterWorse Difference.

Better or Worse Percent Using GetDataCell

This calculates a Variance Percent while considering the Account Type. For example:

- Revenue Accounts are calculated using ((A-B) / Abs(B)) * 100
- Expense Accounts are calculated using (((-1 * A) – (-1 * B))/Abs(B)) * 100

GetDataCell(BWPercent(S#Actual, S#BudgetV1)):Name("BetterWorse %")

Ratio Using GetDataCell

This calculates the ratio between two members using the Divide function. Use the Divide function instead of the division operator (/) to avoid dividing by zero or NoData.

GetDataCell(Divide(A#Cash, A#AcctsRec)):Name(Ratio)

This example returns A#Cash divided by A#AcctsRec, if there is data and the denominator is not zero, in a Cube View column or row named Ratio.

Difference Using GetDataCell

This calculates the difference of two members using the Subtraction operator.

GetDataCell(S#Scenario1-S#Scenario2):Name(Difference)

Sum Using GetDataCell

This calculates the sum of two members using the Addition operator.

GetDataCell(S#Scenario1+S#Scenario2):Name(Total)

Custom Functions to Call Business Rules using GetDataCell

There are Custom Functions found by navigating to **Samples > GetDataCell Expressions** which call a Finance Business Rule using the BR# token within a GetDataCell.

Custom Function:

GetDataCell(BR#[BRName = MyRuleName, FunctionName = MyFunctionName]):Name(Custom Function)

Custom Function (Condensed Syntax):

GetDataCell(BR#[MyBusinessRuleName, MyFunctionName]):Name(Custom Function)

Here is an example of a business rule that dynamically calculates Gross Profit percentage and a Cube View that calls it. The business rule and Cube View Member Filter syntax are below:

- **Business Rule Function**

  The Function name in the example is GrossProfitPercentage, which needs to be referenced in the Member Filter.

```
Case Is = FinanceFunctionType.DataCell
    If args.DataCellArgs.FunctionName.XFEqualsIgnoreCase("GrossProfitPercentage")
        Return api.Data.GetDataCell("Divide(A#GrossProfit,A#Sales)")
    End If
```

- **Cube View Member Filter Using Custom Function(Condensed Syntax)**

  In the GetDataCell string, define the business rule name, function name, and Cube View column or row header name. This example returns the values from the GrossProfitPercentage function in the CorporateCalcs business rule in a Cube View column or row named Gross Profit %.

  GetDataCell(BR#[CorporateCalcs, GrossProfitPercentage]):Name(Gross Profit %)

Custom Function With Parameters

GetDataCell(BR#[BRName=MyBusinessRuleName, FunctionName=MyFunctionName, Name1=Value1, AnotherName=[Another Value]]):Name(Custom Function)

This passes a business rules in a Cube View Member Filter to perform calculations on the members referenced in the Cube View. Configure the Finance business rule to indicate the name of the function and any name-value pairs to reference in the Member Filter. A different action can be performed based on the definition of the name-value pair. The function can be reused with different values passed in through the name-value pairs, making it versatile.

In the example below, the business rule calculates an account as a percentage of sales using accounts in the Cube View member script. The business rule and Cube View Member Filter syntax are below.

- Business Rule Function and Name-Value Pair

  The Function name in the below example is PctOfSales, which needs to be referenced in the Member Filter. The name-value pair is Account, which needs to be referenced and defined in the Member Filter.

```
        Case Is = FinanceFunctionType.DataCell
                If
args.DataCellArgs.FunctionName.XFEqualsIgnoreCase("PctOfSales") Then
                        Dim cvAccount As String_
                        =
args.DataCellArgs.NameValuePairs.XFGetValue("Account")
                        'Calculate account percentage of sales
                        Return api.Data.GetDataCell("Divide(A#" & cvAccount
& " ,A#60999)")
                End If
```

- Cube View Member Filter Using Custom Function With Parameters

  In the GetDataCell string define the business rule name, the function name, the name-value pair from the business rule, and the Cube View row or column header name. This example returns A#GrossProfit divided by A#Sales, based on the business rule and name-value pair, in a row or column named Gross Profit %.

GetDataCell(BR#[BRName = CorporateCalcs, FunctionName = PctOfSales, Account = 61000]):Name(Gross Profit %)

In the example below, the business rule reads the current time period from the rule and gets the value for the prior year based on the Member script. The business rule and Cube View Member Filter syntax are below.

- **Business Rule Function and Name-Value Pair**

  The Function name in the below example is PYMonthForCol, which needs to be referenced in the Member Filter. The Name-Value Pair is Field1, which needs to be referenced and defined in the Member Filter. Based on the name-value pair, the rule can run different actions.

```
Case Is = FinanceFunctionType.DataCell
If args.DataCellArgs.FunctionName.XFEqualsIgnoreCase("PYMonthForCol") Then
            Dim ms1 As String = args.DataCellArgs.NameValuePairs("Field1")
            Dim priorYearTimeName As String = _
api.Time.GetNameFromId(api.Time.GetPriorYearPeriodId())
            Dim memberScript As New System.Text.StringBuilder
            memberScript.Append(ms1)
            memberScript.Append(":T#")
            memberScript.Append(priorYearTimeName)
            'Get the Data Cell For the Prior Year
            Return api.Data.GetDataCell(memberScript.ToString)
End If
```

- **Cube View Member Filter Using Custom Function With Parameters**

  In the GetDataCell string, define the business rule name, the function name, and the name-value pair from the business rule. The Cube View returns the sales, A#60999, from the prior year.

  GetDataCell("BR#[BRName = XFR_CVDataCellHelperNew, FunctionName = PYMonthForCol, Field1 =A#60999]"):Name("Sales Last Year")

Samples of Column/Row Expressions

These samples contain GetDataCell calculations that reference the Cube View column and row names to perform column and row math.

See Column and Row Expressions and the Samples tab in the Member Filter Builder for examples.

Samples of Custom Member List Expressions

There are Custom Member List expressions that call a member list from a Finance business rule. You can build a custom list in a business rule and pass that business rule in a Cube View. This is used to return a list of members or member information when running the rule in a Cube View. MemberInfo allows users to specify additional information such as a Parent Entity ID and an Indent Level. An unlimited number of name-value pairs can be passed to the Member List Business Rule. Some use cases are ranking members by the top ten values in a Cube View or listing members alphabetically.

Custom Member List:

E#Root.CustomMemberList(BRName=MyBusinessRuleName, MemberListName=MyMemberListName)

Custom Member List (Condensed Syntax):

E#Root.[MyBusinessRuleName, MyMemberListName]

Custom Member List With Parameters:

E#Root.CustomMemberList(BRName=MyBusinessRuleName, MemberListName=MyMemberListName, Name1=Value1, AnotherName=[Another Value])

This example returns a member list in a Cube View. The business rule and Cube View Member Filter syntax are below.

- **Business Rule with Name Value Pair**

  In the Business rule, define the List Name and the name-value pairs.

```
Select Case api.FunctionType
Case Is = FinanceFunctionType.MemberList
        If args.MemberListArgs.MemberListName.XFEqualsIgnoreCase("EntityParentList") Then
            Dim entityList As String = args.MemberListArgs.NameValuePairs("EntityList")
            Dim objMemberListHeader As New MemberListHeader
(args.MemberListArgs.MemberListName)
            Dim objMemberInfos As List(Of MemberInfo) = _
api.Members.GetMembersUsingFilter(args.MemberListArgs.DimPk, entitylist, Nothing)
            Dim objMemberList As New MemberList(objMemberListHeader, objMemberInfos)
            Return objMemberList
        End If
```

- **Cube View Member Filter Using Custom Member List**

  In the Cube View Member Filter, define the business rule name, Member List name, the name-value pairs from the business rule, and the members to which it applies. This example returns the base members of E#Houston, which are South Houston and Houston Heights. The name-value pairs can be changed to create additional custom member lists.

  E#Root.CustomMemberList(BRName = XFR_MemberListBasicNew, MemberListName = EntityParentList, EntityList = E#Houston.Base)

This example returns a member list in alphabetical order in a Cube View. The business rule and Cube View Member Filter syntax are below.

- **Business Rule to Sort Member List in Alphabetical Order**

  This business rule sorts the balance sheet member list alphabetically when it is called from a Cube View. It returns A#BalanceSheet.base members for the workflow profile cube and Actual Scenario Type.

```
'This will put a member list of a dimension in Alphabetical order.
'Use the following on the cube view A#Member.[Name of Business Rule, Name of List in
Business Rule]
'A#Root.[XFR_MemberListAlphabetical, Acct_Sort]
```

```
Dim Memberlistname As String = "BSAcct_Sort"
Dim wfProfileName As String = api.Workflow.GetWorkflowUnitInfo.ProfileName
Dim wfProfileCubeName As String = BRApi.Workflow.Metadata.GetProfile(si, _
wfProfileName).CubeName
Dim CurCube As String = api.Pov.Cube.Name

Select Case api.FunctionType
        Case Is = FinanceFunctionType.MemberList
            If args.MemberListArgs.MemberListName = "BSAcct_Sort" Then
                    Dim objMemberListHeader = New
MemberListHeader(args.MemberListArgs.MemberListName)
                    Dim MemberListstart As String =
"A#BalanceSheet.base.Options(Cube = " & _
                    wfProfileCubeName & _
                    ",ScenarioType=
Actual,MergeMembersFromReferencedCubes=False)"
                    'brapi.ErrorLog.LogMessage (si,"Cube Name: " &
wfProfileCubeName)
                    'Read the members
                    Dim objMemberInfos As List(Of MemberInfo) = _
api.Members.GetMembersUsingFilter(args.MemberListArgs.DimPk, _
                    MemberListstart, Nothing)

                    'Sort the members
                    Dim objMembers As List(Of Member) = Nothing
                    If Not objMemberInfos Is Nothing Then
                        objMembers = (From memberInfo In objMemberInfos
Order By _
                        memberInfo.Member.Name Ascending Select
memberInfo.Member _
                        Distinct).ToList()
                    End If

                    'Return
                    Return New MemberList(objMemberListHeader, objMembers)
            End If
End Select
    Return Nothing
```

- **Cube View Member Filter Syntax**

  In the Cube View Member Filter, define the business rule name and the Member List name. The A#BalanceSheet.base members for the workflow profile cube and Actual Scenario Type will show in the Cube View alphabetically by the member names.

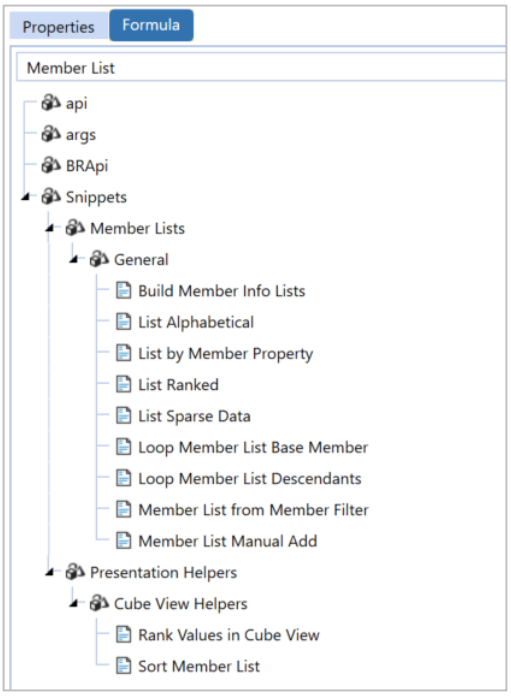
  A#Root.[XFR_MemberListAlphabetical, BSAcct_Sort]

This example shows how custom display names can be applied and displayed in a Cube View header when creating custom Member Lists.

Business Rule for Custom Display Names in Member Lists

```
Select Case api.FunctionType
Case Is = FinanceFunctionType.MemberList
Dim listName As String = args.MemberListArgs.MemberListName
        'Get the Passed in parameters
        Dim entityList As String = args.MemberListArgs.NameValuePairs("EntityList")
        If listName.Equals("AlphaSortList", _
StringComparison.InvariantCultureIgnoreCase) Then
Dim objMemberListHeader = New _
MemberListHeader(args.MemberListArgs.MemberListName)
            'Read the members
            Dim objMemberInfos As List(Of MemberInfo) = _
api.Members.GetMembersUsingFilter(args.MemberListArgs.DimPk, _
entityList, Nothing)
            'Sort the members
            Dim objMembers As List(Of MemberInfo) = Nothing
            If Not objMemberInfos Is Nothing Then
                objMembers = (From memberInfo In objMemberInfos Order By _
memberInfo.Member.Name Ascending Select memberInfo).ToList()
            End If
            'Apply a custom display name to the first item.
            If Not objMembers Is Nothing Then
                If objMembers.Count > 0 Then
                    objMembers(0).RowOrColNameForCalcScript = _
objMembers(0).NameAndDescription & "(Custom Name)"
                End If
            End If
            'Return list
            Return New MemberList(objMemberListHeader, objMembers)
        End If
End Select
```

Search Snippets for Member List in a Finance business rule for examples of business rules for Custom Member Lists, such as the List Alphabetical and List Ranked snippets. The Cube View Member Filter syntax is shown within a comment in the business rule sample.



Samples of XFBR, XFCell, and XFMember Property

XFBR

You can use XFBR to call a business rule to create a custom string substitution.

XFBR(businessRuleName, brFunctionName, Name1=[Value 1], Name2=Value2)

This example uses XFBR to call a business rule to return an alternative View member based on the user's POV. The business rule and Cube View Member Filter syntax are below.

- **Business Rule for Custom Description Syntax**

```
If (args.FunctionName.XFEqualsIgnoreCase("GetAltViewDescription")) Then
            Dim viewName As String = args.NameValuePairs.XFGetValue("ViewName")
            If viewName.XFEqualsIgnoreCase("Periodic") Then
                Return "MTD"
            Else
                Return viewName
            End If
```

```
        End If
```

- **Cube View Member Filter Syntax**

  The Cube View Member Filter contains V# and then defines the business rule's name, the function name, and a name-value pair within XFBR(). If the POV contains V#Periodic, V#MTD is show in the Cube View.

  V#XFBR(XFR_DescriptionHelper, GetAltViewDescription, ViewName=[|PovView|])

This example uses XFBR to call a business rule to return a certain time period based on the scenario selected when the Cube View is run. The business rule and Cube View Member Filter syntax are below.

- **Business Rule for Custom Description Syntax**

  If the scenario is Actual, then the twelve months prior is returned. Otherwise, the Cube View time is returned.

```
If (args.FunctionName.XFEqualsIgnoreCase("timeHelper1")) Then
            Dim currentScenario As String = args.NameValuePairs.XFGetValue
("currentScenario")
            If currentScenario.XFEqualsIgnoreCase("Actual") Then
                Return "MonthPrior12(|CVTime|)"
            Else
                Return "|CVTime|"
            End If
        End If
```

- **Cube View Member Filter Syntax**

  The Cube View Member Filter contains T# and then defines the business rule's name, the function name, and a name-value pair within XFBR(). The Scenario Member in the Cube View POV is |!SelectScenario!|, which prompts the user to select a Scenario member when the Cube View is run. If the user selects S#Actual, the time period 12 months prior is

returned.

T#XFBR(XFR_CubeViewHelper, timeHelper1, currentScenario = |!SelectScenario!|)

XFCell

See XFCell and Using Retrieve Functions in Extensible Documents and the Samples tab in the Member Filter Builder for examples.

XFMemberProperty

This function allows users to specify a Dimension property and display the Member Property selection as a row or column header on a Cube View. This can be used if the header name for a row or column must be different than the member name or description. Use the XFMemberProperty within the Name() or GetDataCell() portion of a Member Filter.

This example returns the base members of A#60999 and shows the Account Type in the row or column instead of the member name or description.

A#60999.Base:Name(XFMemberProperty(DimType=Account, Member=|MFAccount|,Property=AccountType))

The following name-value pair settings are needed in this function:

DimType: Define the dimension type such as Entity, Account, and so on.

Member: Define the dimension member name. In the above example, the member name is defined with a Member Filter Substitution Variable.

> **NOTE:** To reference the dimension members specified in the Member Filter, use the dimension-specific Member Filter Substitution Variables. For example, if the Member Filter is A#NetSales.Children, use =|MFAccount| to dynamically point to each Child Member of Net Sales while running this function. See Member Filter Substitution Variables for more details.

Property: Define the exact dimension member property name, such as Property=AccountType.

Add additional name-value pair settings in this function if the property varies, such as:

- **VaryByCubeType**: Use this if the property varies by Cube Type.

  A#NetSales.Children:Name(XFMemberProperty(DimType=Account, Member=|MFAccount|, Property=Text1, VaryByCubeType=Standard)

- **VaryByScenario**: Use this if the property varies by Scenario Type, such as Actual or Budget.

  A#NetSales.Children:Name(XFMemberProperty(DimType=Account, Member=|MFAccount|, Property=Text1, VaryByScenarioType=Actual))

- **VaryByTime**: Use this if the property varies by a specific Time Period such as 2024M5.

  A#NetSales.Children:Name(XFMemberProperty(DimType=Account, Member=|MFAccount|, Property=Text1, VaryByScenarioType=Actual, VaryByTime=2024M5))

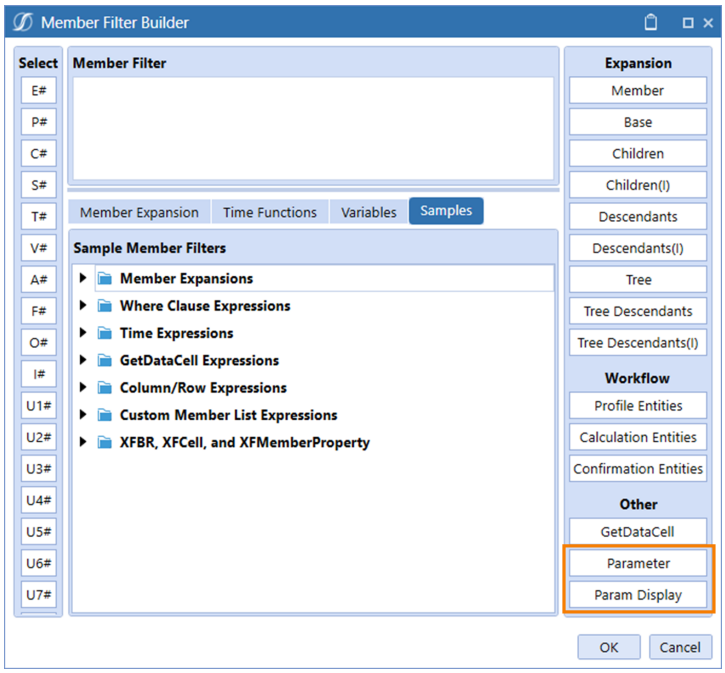
- **TimeDimProfile**: Use this to specify the Time Dimension Profile. This can be set to CV, WF, any Time Dimension Profile name, or a Cube name using the CB# qualifier.

  T#2024.Base:Name(XFMemberProperty(DimType=Time, Member=|MFTime|, Property=Description, TimeDimProfile=|CVTimeDimProfile|))

  The example above uses the |MFTime| Substitution Variable to reference the members in the T#2024.Base Member Filter. It also uses the |CVTimDimProfile| Substitution Variable to point to the Time Dimension Profile for the Cube View. See Member Filter Substitution Variables for more details on these Substitution Variables.

Parameter and ParamDisplay

There are also buttons in the Member Filter Builder for parameters which open the Object Lookup dialog box. Use these to enter a custom Parameter reference that comes from either a Form or Dashboard in a Member Filter. Parameter Display is only used when working with a Delimited List Parameter where the Display Item is displayed instead of the Member name.



Workflow

A primary feature of the OneStream platform is the ability to tailor and optimize all aspects of an application to best fit the requirements of the model design, such as Consolidation, Planning, Forecasting or Operational models. In this section you will learn about the various import methods, workflow analysis, and the blend workflow.

Workflow

The Workflow Engine is the coordinator of all activity in the system. It protects you from having to deal with the complexities of an advanced Multi-Dimension Analytic Model. Profiles and Channels are the basic building blocks of a solid workflow management structure. Data Units represent units of work for loading, clearing, calculating, storing, and locking data within the Multi-Dimensional engine.

Workflow

Workflow is the overall system manager coordinating all end user activities while guaranteeing the quality and transparency of source data used to feed the Analytic Models contained in OneStream applications.

The Workflow Engine's primary responsibilities are to:

- Protect the end user from Analytic Model complexities.
- Manage and audit data collection.
- Manage and enforce the quality process along with the data certification process.
- Manage and intelligently coordinate the data consolidation process.
- Deliver right-time information and guided analysis.
- Create a standardized end-user experience with the capability of becoming personalized through guided Workspaces.

The primary reason Workflow exists is to care and feed Analytic Cubes. Therefore, before a Workflow hierarchy can be created, at least one Cube marked as Top Level for Workflow must exist in the application.

A Cube is an Analytic Model that consists of eighteen Dimensions and provides system designers the ability to quickly and reliably create multiple Cubes within an application. They also create a Dimension Library that enables reusability and Dimension calculation logic. This capability enables system designers to create optimal Analytic Models tailored to the specific business process exhibited without having to move data in and out of the application.

The Workflow Engine's job is to provide a common and seamless user experience no matter how an application designer chooses to implement the underlying Analytic Models (Cubes). This capability enables application designers to create organized applications because they are free from worrying about how to train users on the specific Analytic Models to access during the business process.

When a Cube is defined, it can be marked as a Top-Level Cube for Workflow. This setting tells the Workflow Engine the Cube is eligible for Workflow management. Each Cube can only participate in one Workflow Management Structure, which means if a Cube is referenced by another higher-level Cube, it cannot be set as a Top-Level Cube for Workflow.

After a Cube is tagged as a Top-Level Cube for Workflow, the Workflow Engine will recognize the Cube and allow a Workflow Management Structure to be created for the Cube based on the Suffix Values for varying Workflow by Scenario Type.

Building a Workflow Management Structure for a Cube starts with defining how many variations there will be in the processes used to feed the Cube (suffix for varying Workflow by Scenario Type). Workflow hierarchy variation is aligned with the ten Scenario Types that enable Cube extensibility. By default, no suffix values are created which means the Cube can only have one hierarchy in its Workflow Management Structure.

Setting up a Workflow Management Structure that enables multiple Workflow hierarchies is accomplished by assigning a defining set of suffix values reflecting the different business processes that will be used to populate the target Cube.

Creating Workflow Suffix Groups

For example, a Cube has been created to collect actual reporting data, financial planning data, and some other miscellaneous data collections (Cube Name = FinancialReporting). The first step to defining a Workflow Management Structure for this Cube is to determine how data will be provided to the Cube and the business processes it was designed to analyze.

The table below details the three business processes for the Cube and the proposed suffix values that will identify the business process in the Workflow Management Structure.

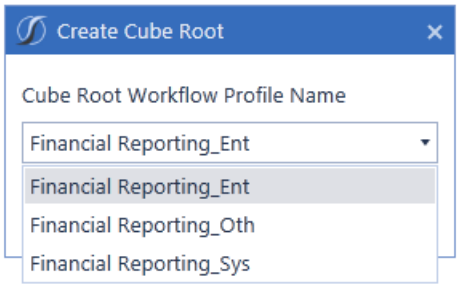
Business Process	Data Source	Suffix
Actual:	System Interfaces	Sys
Plan Data:	Keyed/Excel Uploads by Entity	Ent
Other Data:	Unknown at this point	Oth

Next, the suffix values need to be assigned to the Scenario Type that will be used to capture data for each process. The assignment process will vary from application to application and will be dependent on the requirements of each business process.

Workflow	
Is Top Level Cube For Workflow	True
Suffix for varying Workflow by Scenario Type	
Actual	Sys
Administration	
Budget	Ent
Control	
Flash	Ent
Forecast	
FXModel	Oth
History	Oth
LongTerm	
Model	Oth
Operational	
Plan	Ent
Sustainability	
Target	
Tax	Sys
Variance	Oth

Creating a Workflow Hierarchy Using a Workflow Suffix Group

Three Workflow hierarchies can be created for the FinancialReporting Cube. This is accomplished by opening the Workflow Profiles screen and clicking the Create a CubeRoot Workflow Profile button from the tool bar. The Workflow Engine will automatically present the three Workflow hierarchy options for the FinancialReporting Cube.



The Workflow Engine can now manage three different collection hierarchies for the FinancialReporting Cube. After a Workflow processing has begun and data is being actively collected for the Cube, the Workflow Management Structure (suffix definitions) cannot be changed.

Workflow can be thought of as an outline (Workflow Profile Hierarchy) of the business process that is being used to model and analyze. This section describes the Components of a Workflow Management Structure and the relationship between Workflow Profiles, Entity Members and Origin Members.

Workflow Profiles

Workflow Profiles are the foundation of the data loading process, which is where Data Sources and Transformation Rules are assigned, and the anchors for the review and certifications. This section will go through more detail about all the configuration settings available. See Workflow in Workflow for more details on how the Workflow function works.

> **TIP:** It is best to lay out the organization first in Excel to view and review how the structure looks. Once confirmed, create the same structure in OneStream.

> **TIP:** There are a variety of combinations for building out the Workflow Profiles. Below are a couple from some best practices.

Fully Integrated

This is the complete flow from data loading to certification in the same structure.

Shared Services

Data loading and data certification are integrated; however shared services are separated from the certification.

Separation of Duties

Data load and certification each have their own structures.

Navigation tips for this section

Each profile can have a separate configuration based on the Scenario. Out of the box, the standard configuration is set to (Default).

Workflow Profile Toolbar and Right Click Options

Create a Cube Root Workflow Profile

Use this to begin the hierarchy of a new Workflow Profile.

Create Child Under Current Workflow Profile

There are three types of profiles that can be created:

Review

Reviewer or Certification

Base Input

Data Load

 Parent Input

Parent Adjustments

Create Sibling of Current Workflow Profile

This has the same option as above.

Delete Selected Workflow Profile and its Children

Use this to delete the selected Workflow Profile and any children it may have

Rename Profile

Use this to rename a Workflow Profile or Workflow Profile Input Type

Cancel All Changes Since Last Save

Use this to cancel any unsaved changes

Save

Use this to save any changes made to selected Workflow Profile

Move Current Workflow Profile as Child of Another Profile

Use this to move a selected child of a selected Workflow Profile to another Workflow Profile

Move Current Workflow Profile as Sibling of Another Profile

Use this to move a selected sibling of a selected Workflow Profile to another Workflow Profile

Move Up

Use this to move sibling profiles up in the hierarchy

Move Down

Use this to move sibling profiles down in the hierarchy

Work with Profiles

Use this icon when working in the Workflow Template screen to navigate to the Workflow Profile screen.

Work with Templates

Use this icon when working in the Workflow Profile screen to navigate to the Workflow Template screen. See Workflow Templates for more details on this feature.

Update Input Children Using Template

Select a specific Workflow Profile Template and assign it to a Workflow Profile's input child. See Workflow Templates for more details on this feature.

Unassign Selected Entity

Use this to assign a selected Entity from a Workflow Profile

Show Items that Reference Selected Item

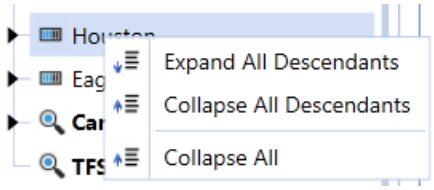
Use this to see the other areas where the selected item is being used.

Navigate

This icon appears in various fields and when clicked it navigates to a section that coincides with the Workflow Profile property. For example, if this icon is clicked in the Cube Name setting, the Cube screen will open allowing the user to make any changes needed before assigning a Cube to a Workflow Profile.

Use the Workflow search tool to filter down to the specific Workflow Profile or Workflow Input Type desired. The Cube Root defaults to the last one selected and displays the associated Workflow Profiles. In order to see another Workflow structure hierarchy, select a different Cube Root.

Right-click on any Workflow Profile name in order to expand or collapse all the selected Workflow Profile's descendants.



Using Workflow Profiles

A Workflow Profile is the basic building block of a Workflow Management Structure. Another way to think about a Workflow Profile is a task list that should be performed by a group of users in relation to a group of Entities. Eight different types of Workflow Profiles are available for use within a Workflow hierarchy. Each Workflow Profile type and its role within a Workflow hierarchy is described below.

Cube Root Profile Type

A Cube Root Profile should be thought of as the definition of a Workflow hierarchy for an entire Cube or a suffix group defined by the Cube. (See the preceding section on Cubes and Workflow). As a result, to create a new Cube Root Profile there must be a Cube defined in the Analytic Model marked as Is Top Level Cube For Workflow = True.

Default Profile Relationship

When a new Cube Root Profile is created, the Workflow Engine will automatically create a matching Default Input profile that is prefixed with the Cube Root Profile name (e.g. Corporate_ Default). A Default Input profile is required for each Workflow hierarchy and it cannot be deleted. The job of the Default Input is to establish the default relationship between the Workflow structure and the Entity Members within the Cube the Cube Root Profile was created to control.

Controlling Workflow States

Cube Root Profiles have a predefined Workflow used to control the state of the Workflow Management Structure (Open / Closed). Cube Root Profiles are used as a mechanism to control the state of the entire Workflow hierarchy for a given Scenario and time because they represent the top element of a Workflow hierarchy.

Open State

The Workflow is available for usage and locking is controlled at the individual Workflow Profile level. In addition, all Workflow hierarchy structure information is read from the current Workflow hierarchy as it reads the Workflow Profiles management screen. This also means the Workflow hierarchy is accessed from memory (cache) rather than being read from the database which provides very fast read performance.

Closed State

The act of closing a Workflow hierarchy triggers the Workflow Engine to place a high-level lock on the Workflow. This means individual Workflow Profile lock status values do not matter, and the Workflow level will display a to indicate a closed Workflow. In addition, the Workflow Engine will take a snapshot of the current Workflow hierarchy structure being managed from the Workflow Profiles management screen. It will store it in a historical audit table for the Scenario and time being closed. This also means the Workflow hierarchy is not accessed from memory (Cache) as would be the case with a Workflow in an open state. A closed Workflow must be read from the database rather than memory because it is considered a point in time snapshot stored in a historical table. This is a performance penalty noticed when reading the entire closed Workflow hierarchy for a Scenario and time. Workflow hierarchies should only be closed if major changes are being made to the Workflow hierarchy and the structure of a Cube and historical hierarchy relationships need to be preserved.

Review Profile Type

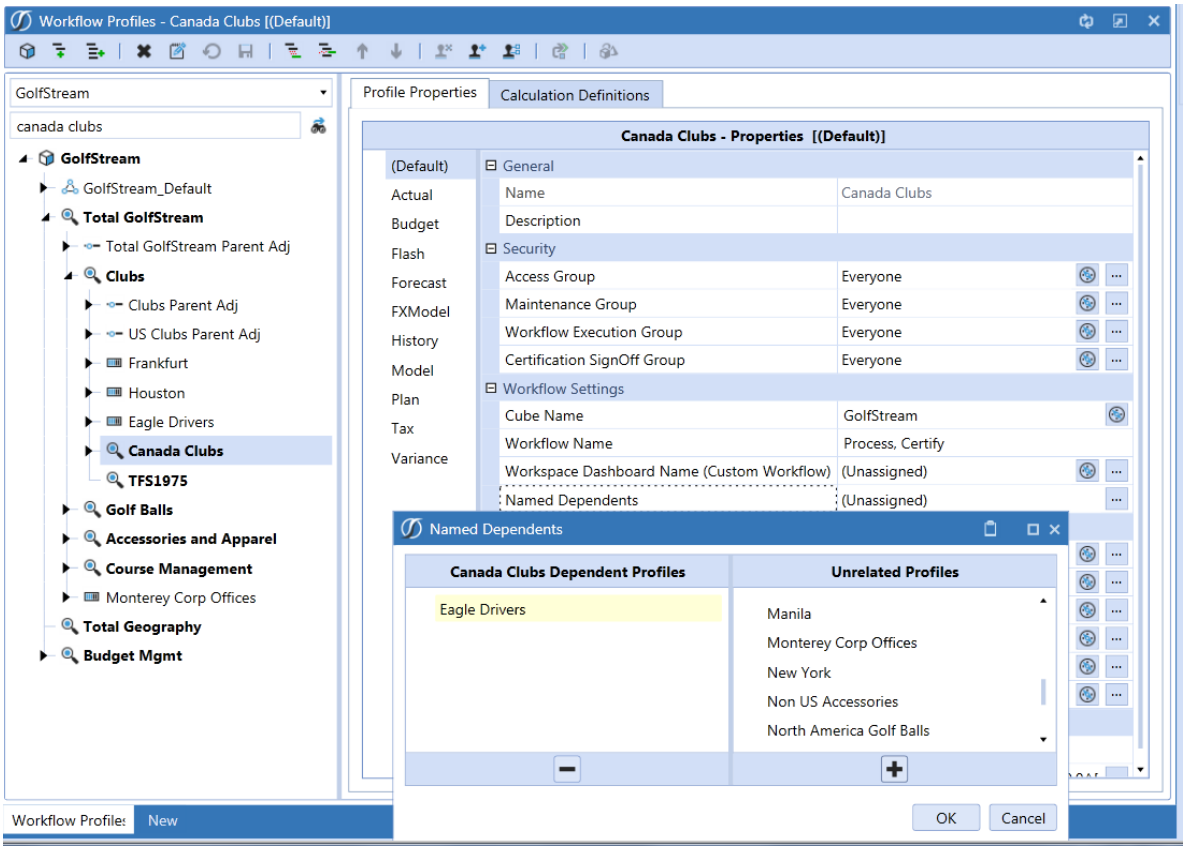
Review Profiles are best thought of as checkpoints in the Workflow hierarchy structure. This type of profile does not have a direct relationship to an Entity Member or Origin Member. However, Review Profiles can have Calculation Definitions assigned to them, so that Cube calculations can be ran in a controlled manner during the Workflow sign-off process.

Named Dependents

Review Workflow Profiles have a unique ability to establish a dependency on the status of Input Parent Profiles that are not their direct descendant in the Workflow hierarchy structure. This concept is referred to as a Named Dependent relationship and was developed to accommodate situations where a single Input Parent Profile loads data for many legal Entities that have very different responsibility structures from a sign-off perspective. This situation is very common when an organization utilizes a Shared Services infrastructure strategy.

Creating a Named Dependent

The screenshot below demonstrates an example of the Canada Clubs Review Profile establishing a Workflow dependency on the Eagle Drivers Input Parent Profile, even though the Eagle Drivers Input Parent is not a descendent of Canada Clubs.



Input Parent Profile Types

Input Parent Profile types are special profile types because they are the point where a relationship is formed between the Workflow Management Structure and an Entity Member. All Input Parent Profiles types share the common purpose of organizing different types of data input used to feed an Entity. This is accomplished through the use and control of dependent profiles called Input Children.

All Input Parent Profiles must have at least one Input Child of each type (Import, Form and Adjustment). This requirement exists because of the relationship between Input Child Profiles and the Origin Dimension Members. Input Child Profiles can be thought of as a specialized extension of the Input Parent with added intelligence and control features particular to data updating.

Default Input

Default Input Parent Profiles are special because they cannot be created directly. They are automatically created whenever a Cube Root Profile is created.

Unassigned Entities

The primary purpose of the Default Input Profile type is to serve as the initial relationship between the Entities belonging to a Cube Root Profile and the Workflow hierarchy. Entities cannot be explicitly assigned to a Default Input Profile. Any Entity Member under the Cube with which the Default profile is associated and is not explicitly assigned to a Parent Input Profile or Base Input Profile, is implicitly assigned to the Default Profile.

Parent Input

Parent Input Profiles are used to allow adjustments to Parent Entities in the Cube. Adjustments to a Parent Entity are only allowed via Forms or AdjInput Members of the Origin Dimension. Consequently, Import Child Profiles are not allowed to be used with a Parent Input Profile. The Workflow Engine will automatically create an Import Child for each Parent Input Profile, but the Import Child will be forced to be inactive (Profile Active = False).

Assigned Entities

The primary purpose of the Parent Input Profile type is to establish a relationship between Parent Entities that require the ability to accept adjustments and the Workflow hierarchy. Parent Entities do not need to be explicitly assigned to a Parent Input Workflow Profile unless the Parent Entity requires the ability to be adjusted. Most Parent Entities exist as unassigned Entities and therefore are controlled by the Default Input Profile.

Base Input

Base Input Profiles are used to control all methods of data entry for Base Entities in the Cube. This is the most common Workflow Profile type and can be thought of as the workhorse of data update management. Base Input Parents define the Entities that can be updated, the Cube being targeted, and all the Import Child types that will participate in the input scheme.

Assigned Entities

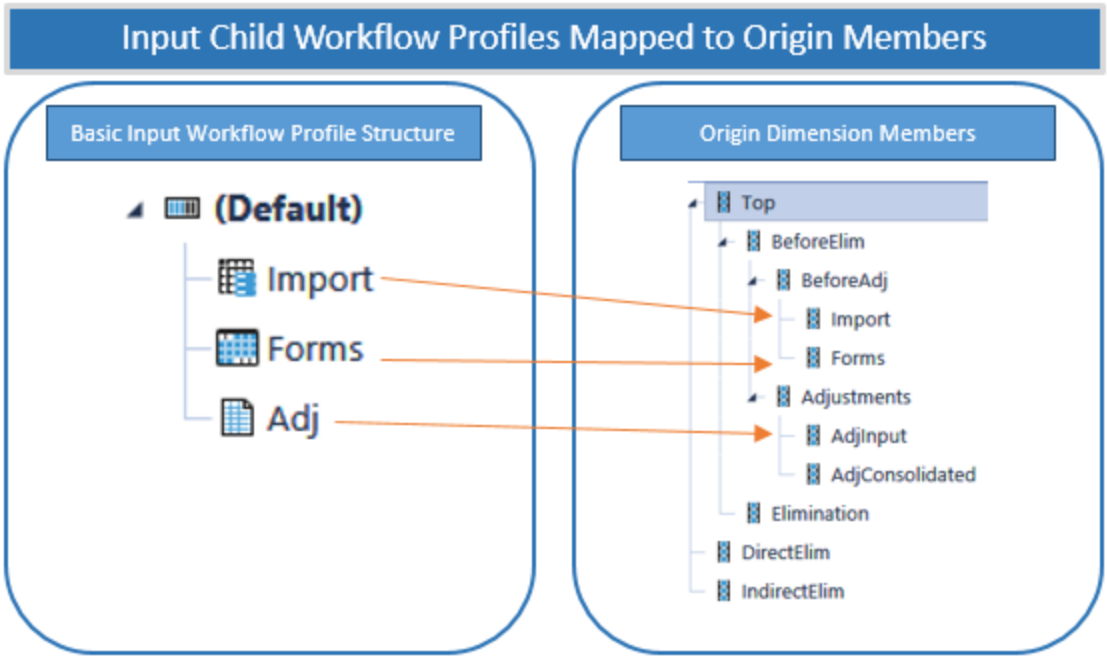
The primary purpose of the Parent Input Profile type is to establish a relationship between Base Entities that need to receive data from an external source and the Workflow hierarchy.

Input Child Profile Types

Input Child Profile types will always be base Members of a Workflow hierarchy and can only be children of one of the three types of Input Profiles: Default, Parent or Base.

Origin Dimension Relationship

Input Children are mapped directly to Origin Members in the Analytic Cube to create a control mechanism between the Workflow hierarchy and the Cube. This linkage enables the Workflow Engine to control the importing, form data entry, adjustment (journal) data entry and data locking processes for one or more Entities.



Controlling Input Channels

The relationship between Origin Members and the Input Child Profiles that map to them enables the Workflow hierarchy to be configured in a way that allows/disallows certain input channels for the Entities assigned to the Input Parent Profile. By setting the Profile Active switch, an entire channel of input can be enabled/disabled.

For example, if a Forms Input Child has the Profile Active set to False, and there are no other active Input Child siblings of the type Form, data entry forms and Excel (SetCells Function or Cube Views) cannot be used to set data cell values for the Entities assigned to the Input Parent Profile of the Forms Input Child. The same technique can be used to enable/disable Import and Adjustment input types.

Import Child

An Import Child defines and controls how data is imported into the Cube (See Data Loading for more details). Each Import Child is bound to the Data Source and Transformation Rule Profile which will define its Workflow behavior during the Import Workflow Step.

Import Child Origin Mapping

The Workflow Engine enforces a relationship between Import Child Profile types and the Import Member of the Origin Dimension. This means, when loading data, the Origin Member will be forced to the value Import.

Forms Child

A Form Child defines and controls how data is manually entered in the Cube. Each Form Child is bound to an Input Forms Profile that will define its Workflow behavior during the Input Forms Step.

Form Child Origin Mapping

The Workflow Engine enforces a relationship between Form Child Profile types and the Forms Member of the Origin Dimension. This means when creating data entry forms, the Origin Member must be set to Forms or the data cell will appear as read only.

It is possible to use a data entry form to update the AdjInput Member of the Origin Dimension, but this requires the account being updated to have its Adjustment Type set to Data Entry rather than the default value of Journal.

Adjustment Child

A Journal Child defines and controls how data is entered via journal into the Cube. Each Form Child is bound to a Journal Template Profile that will define its Workflow behavior during the Input Journals Step.

AdjInput Child Origin Mapping

The Workflow Engine enforces a relationship between Adjustment Child Profile types and the AdjInput Member of the Origin Dimension. This means when creating journal entries, the Origin Member will be forced to the value AdjInput.

Workflow Entity Relationship

The relationship between Workflow Profiles and Cube Entities creates a powerful asset that can act as leverage across many areas of the Workflow experience. By performing the one-time setup process of binding Entities to specific Workflow Profiles, the OneStream Workflow Engine can make this relationship information available to the application designer in many areas of the product. This means Workflow control structures and data entry mechanisms can be designed using abstract methodologies that refer to the current Workflow Unit value, which in turn can be resolved to a list of associated Entities.

Workflow Entity Relationship Member Filter Examples

The primary way the Workflow Entity relationship is used is through analytic Member Filters.

E#Root.WFProfileEntities

When used in a Member Filter, this expression returns all Entities associated with the selected Workflow Unit.

E#Root.WFCalcuationEntities

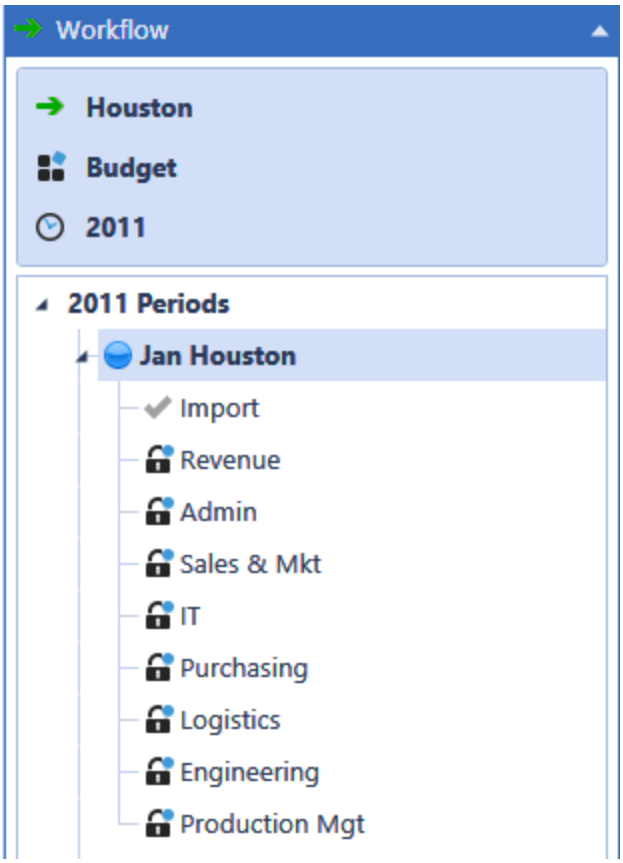
When used in a Member Filter, this expression returns all Entities defined as part of the Calculation Definitions for the selected Workflow Unit.

E#Root.WFConfirmationEntities

When used in a Member Filter, this expression returns all Entities defined as part of the Calculation Definitions when the Confirmed Switch is set to True for the selected Workflow Unit.

Multiple Input Workflow Profiles per Type

An application can have multiple input Workflow Profiles of the same type (Import, Forms, Journals) within each period. This is useful when multiple source systems are feeding the same Entities that need separate Data Sources in XF. It is also convenient when multiple form groups need to be completed by different groups of people. People only see and work on what they have access to because different access groups are created for each. The example below shows a budget where there is one Import Workflow Profile and eight different Form data entry channels that can be assigned to different groups of people. Each of these input channels can have multiple Forms.

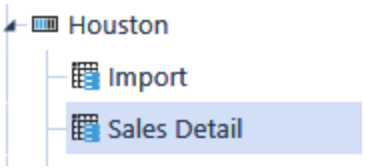


If there are multiple Import Workflow Profiles, it automatically handles clearing and merging data. For example, if there are two Import Workflow Profiles and import has already performed on one of them, when the second import is performed and the user clicks Load, all the target Entities are cleared. The two import data sets are merged, and a replace-style load is made to the financial model.

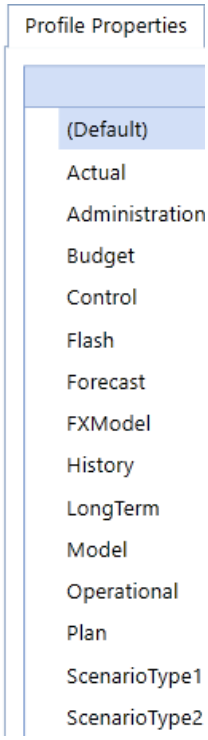
Load YTD and MTD in the Same Workflow Profile

During the configuration of a Workflow Profile for a Scenario, the Workflow may require the submission of YTD data in one Origin and MTD/Periodic data in another Origin within the same Workflow Profile. If the Default View for the Scenario is configured as YTD, the Origin submitting MTD/Periodic data needs to be configured appropriately for it to behave as MTD/Periodic.

Select the Workflow Profile to configure, expand the Workflow Profile, and then select the Origin to be configured. In this example, Houston.Import is YTD and Houston.Sales Detail is MTD.



Next, choose the Scenario type where this behavior is needed. Select (Default) if this behavior is required for all Scenario Types.



Select Integration Settings to change the following properties:

Integration Settings	
Data Source Name	(Unassigned)
Transformation Profile Name	(Unassigned)
Import Dashboard Profile Name	(Unassigned)
Validate Dashboard Profile Name	(Unassigned)
Is Optional Data Load	False
Can Load Unrelated Entites	False
Flow Type No Data Zero View Override	Periodic
Balance Type No Data Zero View Override	YTD
Force Balance Accounts To YTD View	True

As a result, the Flow Type Accounts will be processed as Periodic rather than YTD upon data submission or when no data is loaded for Flow Accounts. The Balance Accounts are forced to be processed as YTD upon data submission.

Cube Root Profile

The Cube Root Profile provides the hierarchy build and organizational structure of the Workflow Profiles for the different Cubes used within the application. All top-level Cubes can have one or more Workflow hierarchies defined depending on whether a suffix was added for a given Scenario Type in Cube settings.

Workflow Profile Types

Base Input

This is where most of the data updates take place and are broken out by the input channels whereby the Origin Dimension Members are filled out: Import (O#Import), Forms (O#Forms) and Journals (O#AdjInput).

Import
Import is typically used to load a GL data file or a OneStream configured Excel Template. The Import Origin can be configured by one of the following:

(Import, Validate, Load), (Import, Validate, Load, Certify), (Import, Validate, Process, & Certify), (Import, Validate, Process, Confirm, Certify) (Central Import), (Workspace), (Workspace, Certify), (Import Stage Only), (Import, Verify Stage Only), (Import, Verify, Certify Stage Only)

Form
Form is used to load data either through a Form template or a pre-configured Excel XFSetCell file. The Form Origin can be configured by one of the following:

(Form Input), (Form Input, Certify), (Form Input, Process, Certify), (Form Input, Process, Confirm, Certify), (Pre-Process, Form Input), (Pre-Process, Form Input, Certify), (Pre-Process, Form Input, Process, Certify), (Pre-Process, Form Input, Process, Confirm, Certify), (Central Form Input), (Workspace), (Workspace, Certify)

Journal
Journal is used to load journal adjustment data through a journal template. The Journal Origin can be configured by one of the following:

(Journal Input), (Journal Input, Certify), (Journal Input, Process, & Certify), (Journal Input, Process, Confirm, Certify), (Central Journal Input), (Workspace), (Workspace, Certify)

Parent Adjustment

If a top side adjustment is needed, do so with a Parent Input Workflow Profile either through a journal or form. Both will update the AdjInput Member in the Origin Dimension.

Review

These Workflow Profiles do not take input, but are meant for reviewing, confirming and certifying results from the lower input Workflow Profiles. Since the Review Workflow Profiles cannot load data, the only tasks available for a Review Workflow Profile are (Process, Certify), (Process, Confirm, Certify)

See Workflow Tasks in Using OnePlace Workflow of the Reference Guide for more details on each of these task types.

Profile Properties

The first tab is the primary configuration tab and is available for all types of profiles. It is where security and other objects are attached.

> **TIP:** This can be configured per Scenario Type.

General

Name

Name of profile.

Description

Brief description of profile.

> **NOTE:** If a description is added to a Workflow Profile in the Default Scenario, the description will display in the Workflow Profile POV dialog in OnePlace.

Security

Access Group

Controls the user or users that will have access to the Workflow Profile at run time to view results.

Maintenance Group

Controls the user or users that will have access to maintain and administer the rule group.

Workflow Execution Group

This group is configured for data loaders and allows users to execute Workflow.

Certification SignOff Group

This group is configured for certifiers and allows users to sign off on the Workflow. This group can be used to separate duties between a data loader and certifier.

Journal Process Group (Journals Only)

Access to this group allows users to process a journal.

Journal Approval Group (Journals Only)

Access to this group allows users to approve a journal.

Journal Post Group (Journals Only)

Access to this group allows users to post a journal.

> **NOTE:** Click 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 appropriate Group. Once the Group is selected, press CTRL and double-click to enter the correct name into the appropriate field.

Prevent Self-Post (Journals Only)

Allows you to prevent users from posting their own journals when users have security rights to post, process, or approve them. The Post button is disabled in journals where the status is Created User, Submitted User, or Approved User. Additionally, the Quick Post button is replaced with Submit, Approve, and Post action buttons.

- If set to True, members of the Journal Post security group cannot post their own journals within this workflow profile. This configuration replaces the Quick Post action with the Submit, Approve, and Post actions if you are a member of all of the following groups: Journal Process Group, Journal Approval Group, and Journal Post Group. Administrators still have access to the Quick Post action.
- If set to True (includes Admins), administrators and members of the Journal Post security group cannot post their own journals in this workflow profile. This configuration replaces the Quick Post action with the Submit, Approve, and Post actions if you are an administrator or a member of all of the following groups: Journal Process Group, Journal Approval Group, and Journal Post Group.
- If set to False, administrators and members of the Journal Post Group are allowed to post their own journals. This is the default.

> **TIP:** Administrators are members of the security group Administrators. This is defined at the environment level. They are also members of the security group set in the Application Security Role called AdministerApplication. This security group defines administrators for specific applications. Administrators are automatically granted rights to process, approve, and post journals without being members of Journal Security groups.

Prevent Self-Approval (Journals Only)

Allows you to prevent users from approving their own journals when users have security rights to approve and process journals. The Approve button is disabled in journals where they are the Created User or Submitted User. Additionally, the Quick Post button is replaced with Submit, Approve, and Post action buttons.

Auto Approved and Auto Reversing journals are excluded from the self-approval restriction. A Journal created from an Auto Approved journal template type will be auto-approved when it is submitted. A journal that is automatically created when an Auto Reversed journal is posted will also be auto-approved.

- If set to True, members of the Journal Approval security group cannot approve their own journals within this workflow profile. This configuration replaces the Quick Post action with the Submit, Approve, and Post actions if you are a member of all of the following groups: Journal Process Group, Journal Approval Group, and Journal Post Group. Administrators still have access to the Quick Post action.
- If set to True (includes Admins), administrators and members of the Journal Approval security group cannot approve their own journals within this workflow profile. This configuration replaces the Quick Post action with the Submit, Approve, and Post actions if you are an administrator or a member of all of the following groups: Journal Process Group, Journal Approval Group, and Journal Post Group.

- If set to False, administrators and members of the Journal Approval Group can approve their own journals. This is the default.

> **TIP:** Administrators are members of the security group Administrators. This is defined at the environment level. They are also members of the security group set in the Application Security Role called AdministerApplication. This security group defines administrators for specific applications. Administrators are automatically granted rights to process, approve, and post journals without being members of Journal Security groups.

Require Journal Template (Journals Only)
Restricts users from creating new free-form journals by disabling the Create Journals button. It also prevents user from loading journal files that do not contain a journal template, enforcing new journal creation from existing journal templates only. System and Application administrators can still create free-form journals. This property works in combination with the Journal Process Group property, which defines who can create journals in the workflow profile. Set the value to True to enable this feature.

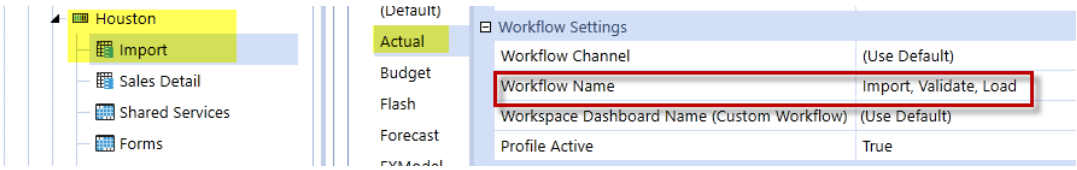
Workflow Settings

In this section the administrator can assign a Workflow Channel and a Workflow name. It is important to understand the tasks of each input type in order to properly assign a Workflow name.

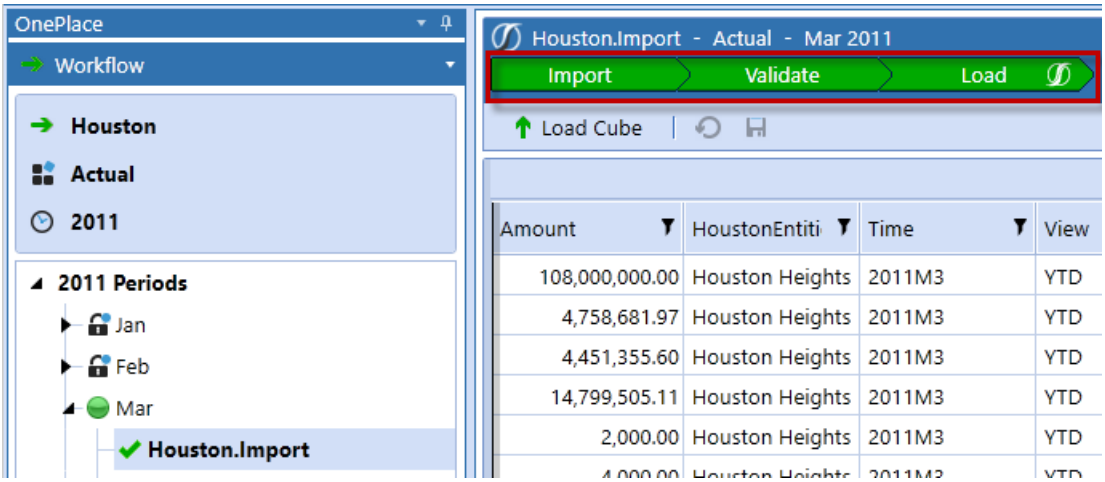
Workflow Channel
This option ties to the Workflow Channels found under the Application Tab/Workflow/Workflow Channels. This allows for an additional layer of security that defined through the Dimension Library. (e.g., Accounts, or UD1 – UD8). Click on the drop down to display all configured Workflow Channels. By default, this is set to Standard. This is an option on each load channel (such as Import, Forms, and Adjustments).

Workflow Name

The Workflow name controls the tasks the users need to complete in the Workflow. There are a variety of combinations based on the type of Workflow being designed. These tasks can vary by Scenario and Input Type. For example, the Workflow Name Import, Validate, Load is set for the Houston Workflow Profile Import Input Type for the Actual Scenario.



When working in the Houston Workflow and loading data in the Actual Scenario, complete the following tasks:



See Workflow Tasks in OnePlace Workflow for more details on these tasks.

Workspace Dashboard Name (Custom Workflow)

This property only works with a workflow name of Workspace or any selected containing workspace (for example, Workspace, Certify). Use this property to define any dashboard from the Application Dashboards page when an appropriate workflow name is chosen.

Integration Settings (Import Only Except Where Indicated)

Data Source Name
The way Fixed Width, Delimited, Data Management Export Sequences, or SQL-based based Data Sources are all managed here. These are originally configured under the Application Tab|Data Collection|Data Sources.

Transformation Profile Name
The type of maps used for the Workflow Profile are managed here. These are configured under the Application Tab|Data Collection|Transformation Rules. This option will display the choices of profiles available.

Import Dashboard Profile Name
The type of Dashboards wanted for the Workflow Profile Import phase are managed here.

Validate Dashboard Profile Name
The type of Dashboards wanted for the Workflow Profile Validate phase are managed here.

Is Optional Data Load
If a Workflow needs to load data to some periods, but not others, this option provides a quick way to complete the Workflow Channel if no data is loaded that month. When set to True, this Workflow Channel will receive an additional icon called Complete Workflow. You can then click on this and complete the process if no data is loaded to this channel.

Can Load Unrelated Entities (Import and Adjustment)
Typically set to False unless there is historical data to be loaded.

- If set to True, a Workflow Profile can load data to entities that are not assigned to its Input Parent Workflow Profile.
- If set to False, a Workflow Profile can only load data to assigned entities. A journal can be saved with unassigned entities, but cannot be submitted, approved, or posted.

When creating a new blank journal, the Entity Member Filter property is set to the value E#Root.WfProfileEntities. This ensures that entity properties show only assigned entities in its list. This filter value can be changed, but when creating journals from a template, the Entity Member Filter property uses the value from the template instead.

> **TIP:** The next three configurations are for configuring Workflow Profiles that may load as MTD. The entire Workflow for the year must be loaded consistently the same way for Zero No Data.

Flow Type No Data Zero View Override

The Workflow will override the Scenario settings of load data from Stage and force Flow Accounts to the Zero No Data Selected. Settings are YTD and Periodic.

Balance Type No Data Zero View Override

The Workflow will override the Scenario settings of load data from Stage and force Balance Accounts to Zero No Data Deselection are YTD and Periodic.

Force Balance Accounts to YTD View

If set to True, Balance Accounts are forced to the YTD View for loading no matter what View Member is assigned to the account in the data load file, if set to False, Balance Accounts are loaded with the View Member assigned to the account in the data load file.

Cache Page Size

Integer must be greater than 0. Setting will default to 20000 records upon save if an invalid value is entered.

Cache Pages In-Memory Limit

Integer must be greater than 0. Setting will default to 200 upon save if an invalid value is entered.

Cache Page Rule Breakout Interval

Integer must be greater than 0. Setting will default to 0 upon save if an invalid value is entered. The value entered reflects the count of Transformation Rules which define a point to pause processing to check if the current Cache Page's data records are fully mapped. If the current Cache Page is determined to be fully mapped, the remaining Transformation rules will not be processed.

Form Settings (Forms Only)

Input Forms Profile Name

The form templates displayed in the forms channel for this Workflow Profile are managed here. These are configured under the Application Tab|Data Collection|Form Templates.

Journal Settings (Journals Only)

Journal Template Profile Name

The journal templates displayed in the adjustments channel for this Workflow Profile are managed here. These are configured under the Application Tab|Data Collection|Journal Templates.

Data Quality Settings

Cube View Profile Name

The Cube Views displayed in the analysis pane are managed here. This section is displayed when clicking on the period level during the Process, Confirm, and Certify steps. These are configured under the Application Tab|Presentation|Cube Views.

Process Cube Dashboard Profile Name

The Dashboard’s rules that run during the Process step are managed here.

Confirmation Profile Name

The Confirmation Rules that run during the Confirm step are managed here. These are configured under the Application tab|Workflow|Confirmation Rules.

Confirmation Dashboard Profile Name
The Dashboards displayed during the Confirmation step are managed here.

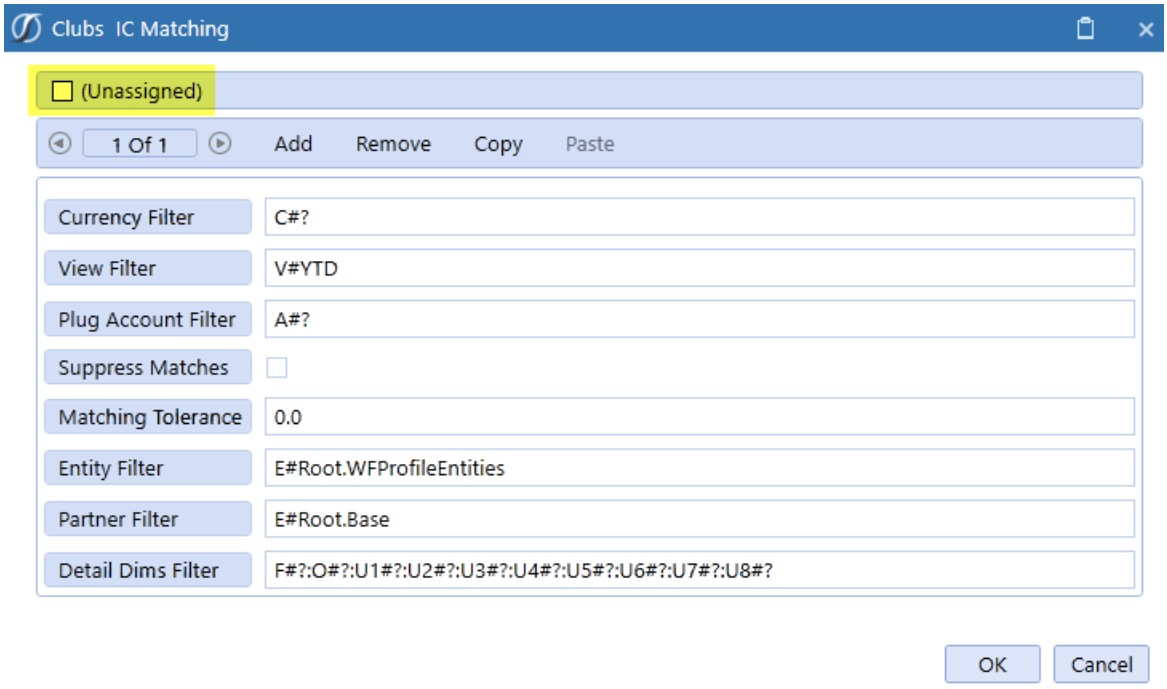
Certification Profile Name
The certification questions that prompt the user during the Certify step are managed here. These are configured under the Application Tab/Workflow/Certification Questions.

Certification Dashboard Profile Name
The Dashboards displayed in the analysis pane are managed here. This section is displayed when clicking on the period level on the Certify step. These are configured under the Application Tab/Presentation/Dashboards.

Intercompany Matching Settings

Matching Enabled
When set to True, the Matching Parameters section needs to be configured. This can be configured at all channels of the Workflow Profile. Settings are True or False.

Matching Parameters
Click on the ellipsis button in this field to configure the matching settings.



> **TIP:** Make sure the (Unassigned) box is unchecked, otherwise the fields will not be available to update.

Currency Filter

Enter the reporting currency.

View Filter

Enter how the Intercompany data should display in the Workflow (such as V#YTD or V#Periodic).

Plug Account Filter

Enter the Plug Account for this Intercompany match.

Suppress Matches

Check this to apply suppression.

Matching Tolerance
If the matching tolerance must be on the penny, leave this at 0.0, otherwise add a tolerance threshold for the offset amounts.

Entity Filter
The Workflow Entities associated with the Intercompany matching.E#Root.WFProfileEntities automatically points to the Entities assigned to the Workflow.

Partner Filter
Enter a Member Filter specifying the Partner Entities.

Detail Dims Filter
Enter the Account-Level Dimension Members (Flow, Origin, UD1…UD8) to ensure users are seeing the correct data in the Workflow.

Entity Assignment

The third tab is only available when clicking on the Cube Root Workflow Profile. This is where the actual Entity gets tied to the Workflow Profile. There are two sections to this tab:

Entity Assignment [Cube Name]
This section repaints the Workflow as it was created in the middle pane. This is where an administrator can click on a Workflow Profile, and then use the Unassigned Entities tab to search for Entities.

Unassigned Entities
This is the search window to find Entities to attach to a profile. This only becomes enabled for the data loading profiles. The search engine uses that contains technology to find Entities. In other words, type the whole word or part of the word and it will search through the complete list looking for that combination of characters. More than one profile may be chosen. Lastly, once an Entity has been attached, it will no longer appear in the search window.

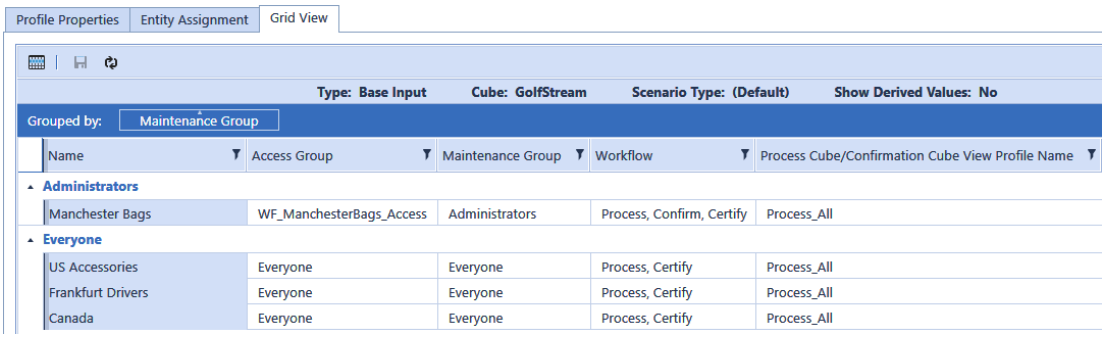
- Click this after the search criteria has been typed

- Click this to attach the item to the profile

- Click this to turn off the search criteria

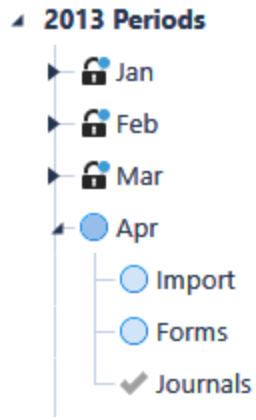
Workflow Profile Grid View

The Workflow Profile Grid View allows an administrator to make several changes to numerous Workflow Profiles at once. Select the Grid View icon and choose data to view. There is also a drag and drop option where the user can select a column label and drag it to the top in order to specifically group the data. This is only available when the Cube Root Profile is selected.



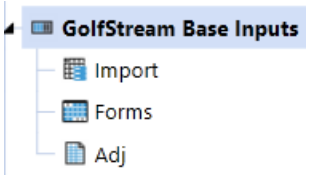
Central Input

For corporate configurations there is an option called Central Input. This can be used for situations where corporate does the final adjustment after the data has been loaded by the sites. The Workflow Channel can configure with Central Input which will then display a grey check mark for that Workflow Channel. This allows corporate to make updates and adjustments at the top level because the Workflow owns the Entity. All activity is tracked in the audit history.



Workflow Templates

Workflow Templates are useful when building a series of Base Input Workflow Profiles with similar settings. Design a template as generic or customized as desired and then apply the template to the new Base Input Workflow. On the Workflow Templates screen, there is a (Default) Template that can be used or click to create a new one. After the template is created, it will look similar to a Base Input Workflow Profile and include the default input types.



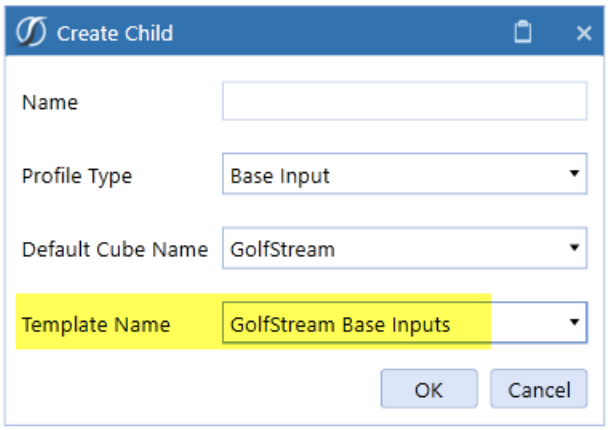
Customize the template to fit the needs of the Workflow design:

- Rename the inputs (for example, rename Adj to Journals)
- Add additional input types
- Disable input types by Scenario (for example, disable Journals for the Budget Scenario if that input type is not used)

- Configure Intercompany Parameters
- Assign common Cube View or Dashboard Profiles

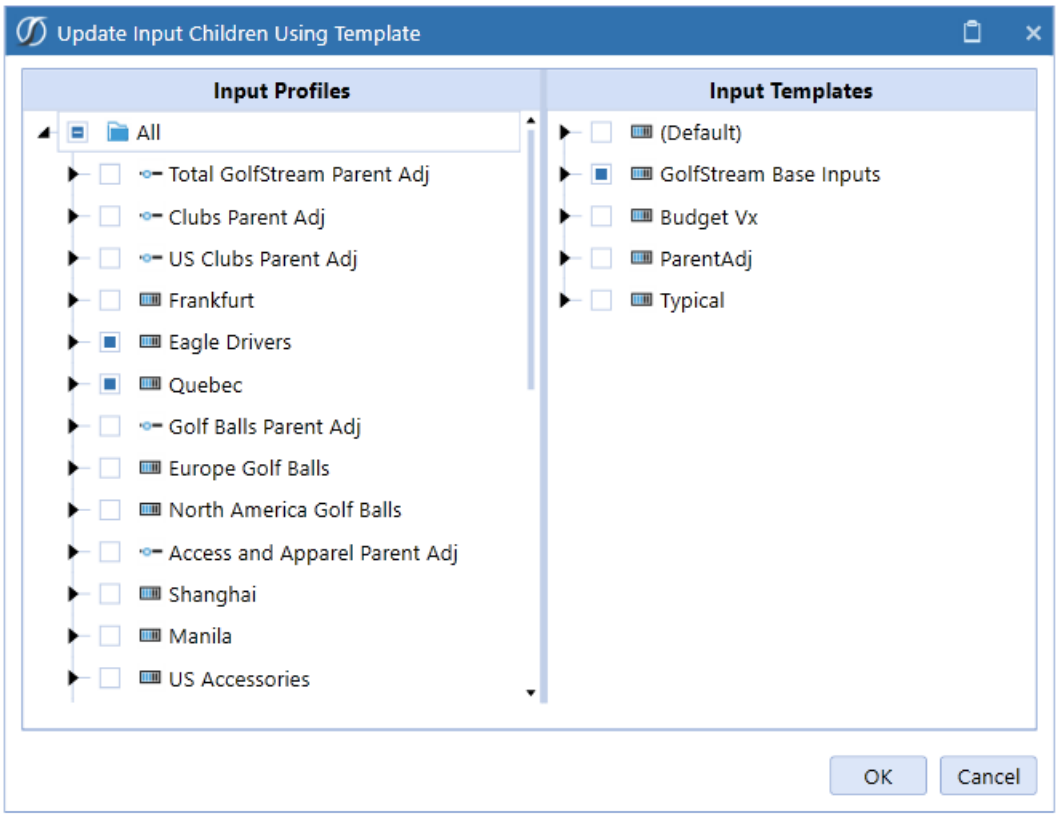
The goal of the template is to make as many common changes and updates as possible which will save time and clicks during the actual Workflow build. After a template is completed, click to navigate back to the Workflow Profile page.

Create a Base Input Workflow and apply the template.



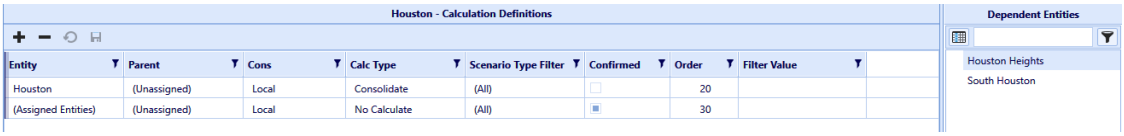
All the template settings now apply to this Base Input.

> **NOTE:** After a template is applied to a Base Input Workflow Profile, any changes made to the existing input types (Import, Forms, Journals) cannot be applied to the Workflow Profile. If a new Input Type is added to the template, this can be applied to an existing Workflow Profile. From the Workflow Profile screen, select . Select the template with new input types and the Workflow Profiles to which the changes will apply.



Using Calculation Definitions

All Workflow Profile types, except Cube Root, can create a set of Calculation Definitions. A Calculation Definition can be thought of as a macro or set of instructions performed whenever the Process Cube Step is executed.



Calculation Definitions are an incredibly valuable tool to the application designer because they take the guess work out of what needs to be calculated and when. During the Workflow hierarchy design process, the Calculation Definitions can be used to run combinations of calculations, translations, and consolidations at Workflow completion points.

When defining Calculation Definitions for a Workflow Profile, the Workflow Entity Relationship can be leveraged. This means predefined variables can be used to run calculations for Entities assigned directly to a Workflow Profile (Input Parent types) or related to the Workflow Profile through its dependency chain.

Calculation Definition Entity Placeholder Variables

Review Profile Type

Dependent Entities
This defines a calculation for all Entities that assigned to the dependent Workflow Profiles of the Review Profile. This list also includes all Entities assigned to any Named Dependent Profiles.

Named Dependent Filter
This allows a filter value to be used to specify the specific Entities that are valid dependents of the Review Profile. This is required because in most cases Named Dependents are shared services and tend to have Entity relationships that span over multiple Review Profiles. This includes all Entities assigned to a Named Dependent that may return Entities not relevant to a Review Profile.

Input Parent and Input Child Profile Type

Assigned Entities
This defines a calculation for all Entities directly assigned to the Workflow Profile.

Loaded Entities
This defines a calculation for all Entities imported by the Import Child Workflow Profiles and are dependents of the Input Parent.

Journal Input Entities

This defines a calculation for all Entities adjusted with journal entries by the Adjustment Child Workflow Profiles and are dependents of the Input Parent.

User Cell Input Entities

This defines a calculation for all Entities affected by data entry performed by the user executing the Workflow. This variable will return a different Entity list for each user. This is typically used in situations where a Workflow is setup without specifically assigned Entities but must update specific data cells across Entities owned by other Input Parent Profiles. This is referred to as a Central Input Workflow design and is usually used by corporate offices to control values in select Entities and accounts.

Input Child Profile Type

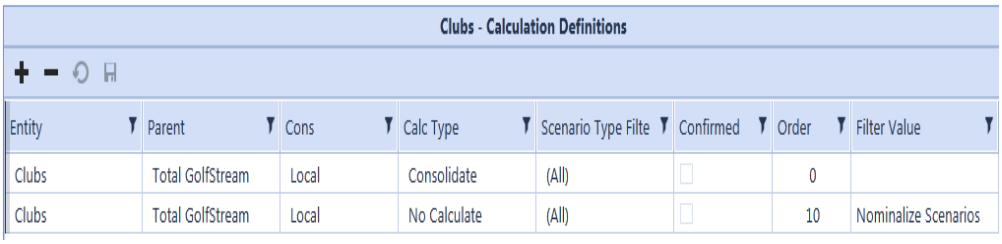
Input Child Profiles are considered extensions of their Parent Input Profile and as a result, they do not require an explicit Calculation Definition. If an Import Child does not have an explicit Calculation Definition defined, it will default to those defined by its Input Parent.

Confirmed Switch Value

Each Calculation Definition record has a Confirmed switch associated with it. This switch determines whether the Entities defined by a Calculation Definition should be subjected to the Confirmation Workflow Step. It also gives the application designer control over which Entities are subject to the Confirmation Rule validation process.

Filter Value

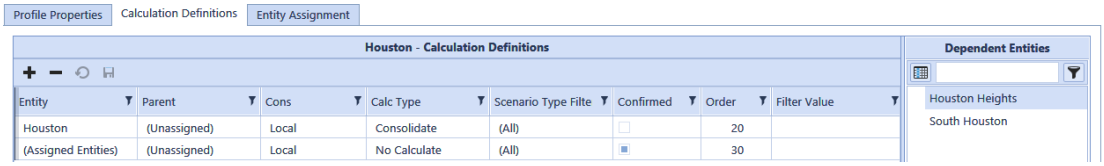
Assign a Data Management Sequence to Calculation Definitions by setting the name of the Sequence under the Filter Value and setting the Calc Type to No Calculate. Next, set up a DataQualityEventHandler Extensibility Business Rule to read the Sequence name assigned to the filter and in turn, run the Data Management sequence during the Process Cube task in the Workflow.



Entity	Parent	Cons	Calc Type	Scenario Type Filte	Confirmed	Order	Filter Value
Clubs	Total GolfStream	Local	Consolidate	(All)		0	
Clubs	Total GolfStream	Local	No Calculate	(All)		10	Nominalize Scenarios

Calculation Definitions

The second tab is where the administrator assigns the Calculation Definitions for the Workflow Profile. This will determine the type of calculation/consolidation that occurs when a user selects Process Cube during the Workflow. Multiple Entities can be entered and ordered accordingly. See Using Calculation Definitions in Workflow in the Design Guide for more details.



Entity	Parent	Cons	Calc Type	Scenario Type Filte	Confirmed	Order	Filter Value
Houston	(Unassigned)	Local	Consolidate	(All)		20	
(Assigned Entities)	(Unassigned)	Local	No Calculate	(All)		30	

Auto-assigning Entities can be done through (Assigned Entities) or (Load Entities). The Confirmed check box controls which Entities are tested by Confirmation Rules.

> **TIP:** By right clicking on any line item, a user can insert or delete a row, save, or export data.

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 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 granularity 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 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 fine-grained 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 for more information.

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

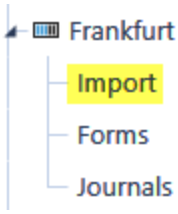
Clear all workflow channel data units loaded by the workflow unit. User-defined members and workflow data unit members are standard members of a workflow channel data unit, so data clears at a user-defined member entity, scenario, time, and account level by default.

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

Workflow Profile Data Loading Behaviors

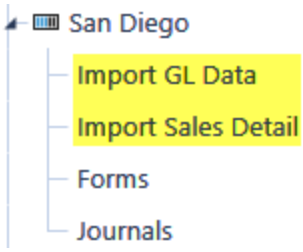
This section describes three specific data loading behavior patterns. These behaviors range from the basic data load process where one workflow profile loads one data unit, to more than one workflow profile loads one data unit.

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



This workflow profile configurationhas 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 pofiles 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.

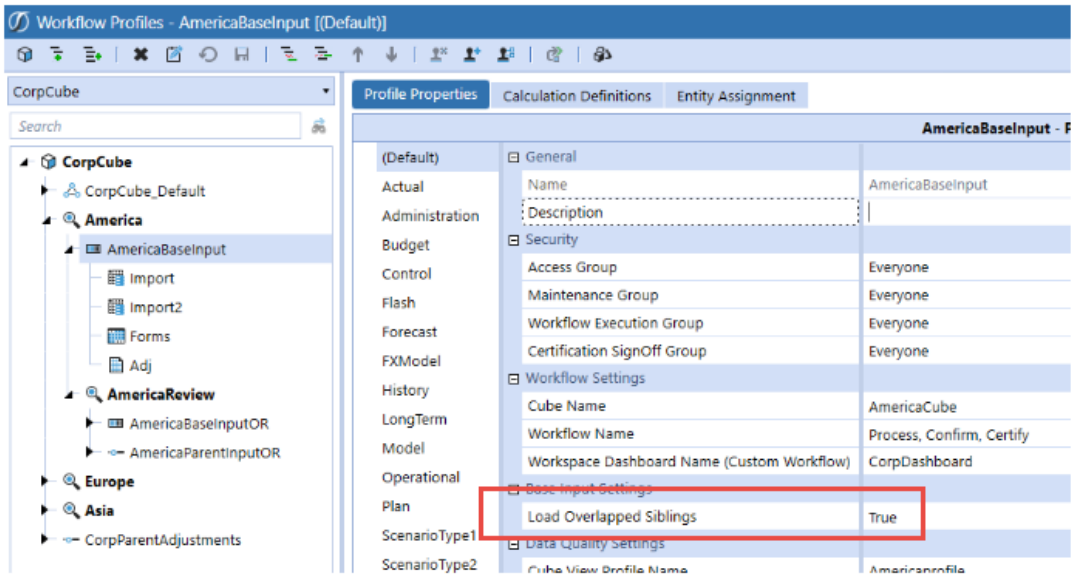
In this situation, when either Central HR Load or Houston executes a data load, the basic clear and replace data loading steps described in the previous section are used. However, Central HR Load does not control any entities so it checks and abides by the workflow and locking status of the workflow 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 (Data Entry Form or Excel, etc.) 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 Entity(s) 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.

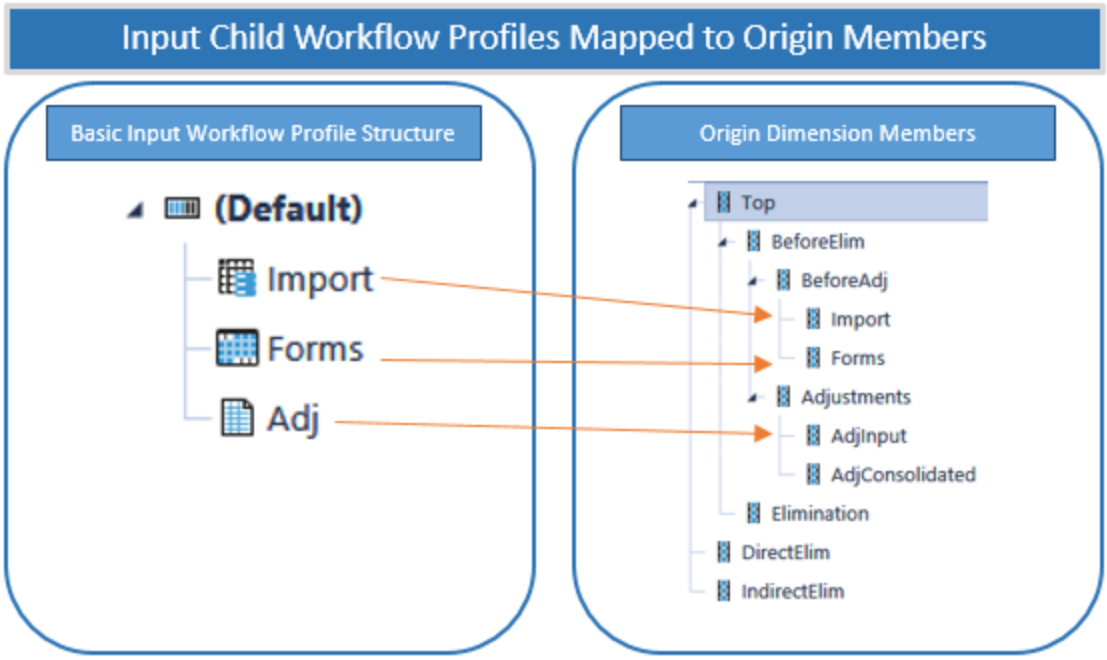
Locking Granularity

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)

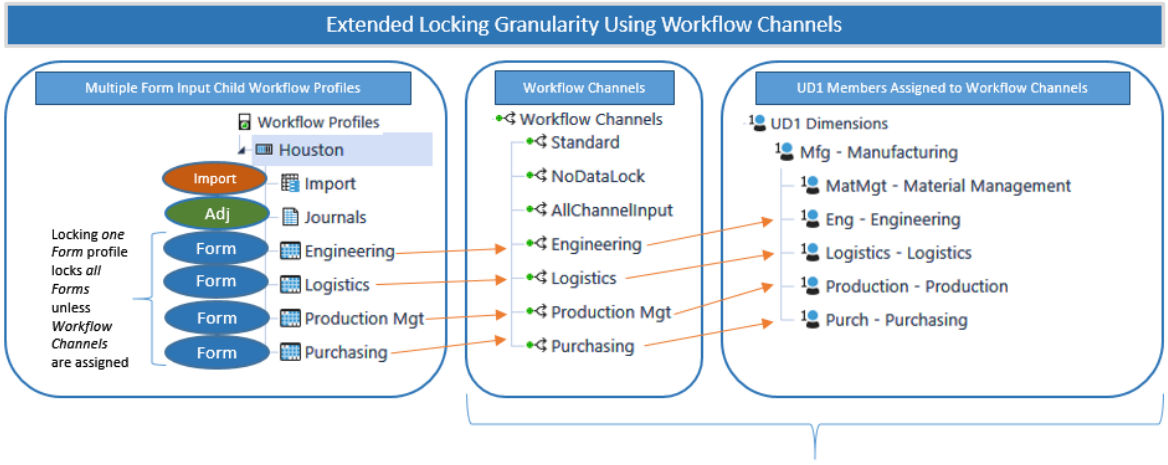
OneStream utilizes a predefined relationship between the three types of Input Child Workflow Profiles and a static Dimension called the Origin Dimension to control and lock the three basic channels of input for an Entity. This relationship provides the ability to control and lock data entry processes 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

Workflow Stage Import Methods

Workflow Stage Import has three primary methods of integrating data from the Stage Engine to meet the reporting requirement for each OneStream application model.

- **Standard**: Highly durable and auditable, stored details that target Finance Engine Cubes.
- **Direct**: In-memory, performance-focused, no storage of record details, that target Finance Engine Cubes.
- **Blend**: In-memory, high-performance import designed to blend the multi-dimensional structure with transactional data. No storage of record details or targeting external relational tables needed.

Standard Import Methods

A Consolidation Model requires that data be durable and auditable, as OneStream is functioning as the “book-of-record” for financial reporting. In this regard, the Stage Engine’s Standard Type Workflow is ideal in that each Import performed has its source and target records stored in Stage tables. This allows every current and historical period loaded to be audited and analyzed at the source and target record level used for loading data to the Finance Engine. The stored tables also allow detailed analysis from the Finance to the Stage using the Drill-Down feature.

Direct Import Methods

Certain Planning, Forecasting or Operational models may not require detailed audit and historical durability of data, with the resulting database overhead, that is required of the Consolidation Model. The Stage Engine's Direct Load Type Workflow is specifically designed to support the needs of data that is more operational in nature. The Direct Load Type's in-memory processing, and lack of storing Source and Target record detail, enhances its performance in processing compared to the Standard Type Workflow. The performance benefits of not storing source and target records is also what makes the Direct Load Type inappropriate for Consolidations, where detailed audit, history and drill-back is required.

Blend Import Methods

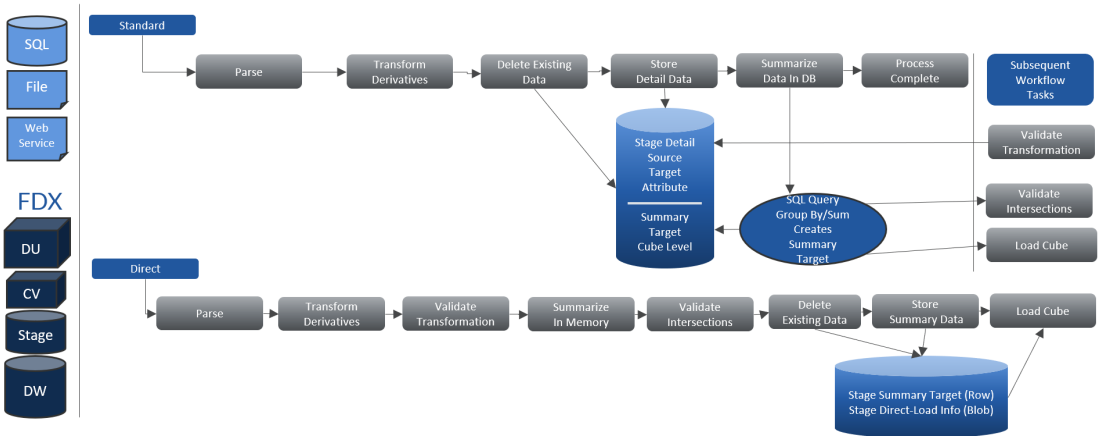
The Blend Type Workflow is an in-memory process and the integration method used by the BI Blend Engine, which writes to external relational database tables and not to a Finance Engine Cube.

The BI Blend Engine is a key element of Analytic Blend models as it is a read-only aggregate storage model. The purpose is to support reporting on large volumes of data that is not appropriate to store in a traditional Cube, such as transaction or operational data.

The Blend Type Import rationalizes the source data into a structure that is uniform and standardized for reporting by leveraging Cube Dimensions, deriving the metadata and aggregation points for the resulting BI Blend relational tables. This enables the transaction content stored in the relational tables to be aligned with the Finance Engine Cube data through common metadata and aggregation points for Analytic Blend Reporting. Refer to the *BI Blend Design and Reference Guide* for more detail.

Stored versus In-Memory Workflow Imports

The primary difference between the Standard and in-memory processing Import methods is how the source and target record details are stored in the Stage tables.



Guidelines on Volumes and Limits

Workflow Import performance is optimized by designing a Workflow structure that considers the data volumes of the total number of source records and the resulting total number of summarized target records. The transformed source records are summarized in the **StageSummaryTargetData** table. The volumes can be managed by structuring the data source loads across Workflows, gaining advantages in parallel processing.

Limit	Description
Row Limit Per Workflow	24 million summarized records
Best-Practice Recommendation	1 million summarized records

Example	Description
Single Workflow Results	24 million summarized records
Best-Practice Solution	Parse the file to be used across 24 partitioned Workflows

The benefits of efficient Workflow structure using partitioning when working with large data sources are:

- Performance gains through parallel processing
- Shorter processing times
- Faster mapping and error correction
- More transparent data validation

Standard Workflow Record Analysis

OneStream Task Activity / Load Cube presents the details of the total number of summary records in the **SelectStageSummaryTargetRows** entry.

Direct Workflow Record Analysis

The Direct Load Status / Execution Status screen displays the Summary Row Count for each load process to display the total number of summary records generated.

Blend Record Analysis

The Blend process differs from other Stage Import processes by generating additional data records, or rows, rather than one-to-one or summarizations. This is because the Analytic Blend designs require the generation of aggregation points, which add to the source rows. Detailed analysis of the BI Blend Processing Logs, Live Row Count statistics, and Task Activity BI Blend Load and Transform help guide the requirements for the application and systems. Refer to the *BI Blend Design and Reference Guide* for more detail.

In-Memory Workflow Imports

Direct Load Workflow Import

Workflow Import Type is optimized for performance by combining the Parse/Transform, Validate Transformations and Intersections, and Load Cube into a single Workflow process. This consolidation of processes, and resulting performance gains, is achieved by functioning in-memory and bypassing the overhead of writing and storing Source and Target record details to Stage database tables. The in-memory processing limits certain functionality and therefore may not be an appropriate solution for all model requirements. All other Workflow functionality remains as part of the Direct Type, such as all Business Rules, Transformation Rules and Derivative Rules.

Direct Load Use Cases

The Direct Load Workflow is designed for data that has a high frequency of change and does not demand durability for audit and history.

Common Uses

- Data Integrations where the OneStream Metadata and Source System are mirrored, allowing “* to * “ Transformation rules to pass-through all records, minimizing the need Drill-Down or Transformation analysis.

- Data that is “disposable” in nature. Typically, this may be data that has a high frequency of changes and may only be valid for a short time. Perhaps only valid for one to seven days.
- High-volume data loads, such as nightly batch loading, where optimal performance is desired. Such data is commonly deleted and reloaded frequently.
- Extended Application data moves, where data from a detailed application feeds a summarized target application.
- Where target data in OneStream is not required to be durable “book-of-record” data.

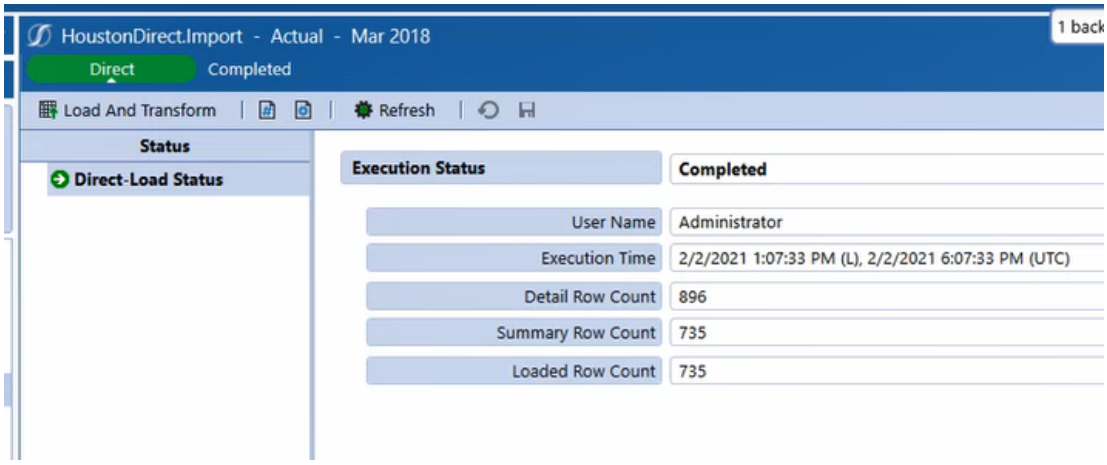
Important Limitations

A key differentiator of the Direct Load is that it does not store source and target records in Stage Database tables. This, by design, will eliminate the audit and historical archiving of Workflow Activity. Other limitations as a consequence of the in-memory Workflow are that the Drill-Down feature will not function to support analysis of records between the Finance and Stage tables.

- Direct Load Type is not appropriate where:
    - Source file import history is required for historical reference.
    - Transformation Rule history is required for historical reference.
    - Drill-Down from Finance Engine is required.
    - Text based View member values are required to be file based.
    - Data loads are required to Append to prior Imports.
- Direct Load Type does not support historical audit of workflow history, such as Import and Transformation Rule history.
- Direct Load Type does not support Re-Transform as Import records are not stored data. Data must be re-loaded.

- Transformation and Validation analysis and map correction is limited to 1000 error records per load.
- Data files cannot load to Time and Scenarios beyond the current Workflow Scenario and Time. The data record's Time and Scenario being loaded must match the Workflow Scenario and Time.

As Direct is an in-memory Workflow with only a single step for the data integration process, Load And Transform. The Direct Load Execution Status screen provides statistics to analyze the Workflow's performance. These key statistics are helpful in determining if the Workflow design is supportive of best-practice designs to optimize application performance.



- **Detailed Row Count**: The total number of Data Source and Derivative Rule records.
- **Summary Row Count**: The total number of records summarized in the Transformation process.
- **Loaded Row Count**: The recorded number of records loaded to the Finance Engine target Cube, which should always equal the Summary Row Count.

Direct Load Transformations and Validations

The Direct Load Workflow's in-memory processing results in Transformation and Validation errors that are not stored being stored in a table. The total number of errors that can be processed and presented in Validation is limited to 1000 records. If the total number of errors exceeds 1000, the data must be re-loaded to re-execute the Transformation and Validation process to generate the next batch of errors, at a maximum count of 1000 records per load.

- Total Direct Load Error Storage Limit = 50,000 records
- Error Presentation Limit = 1000 records presented per Import load

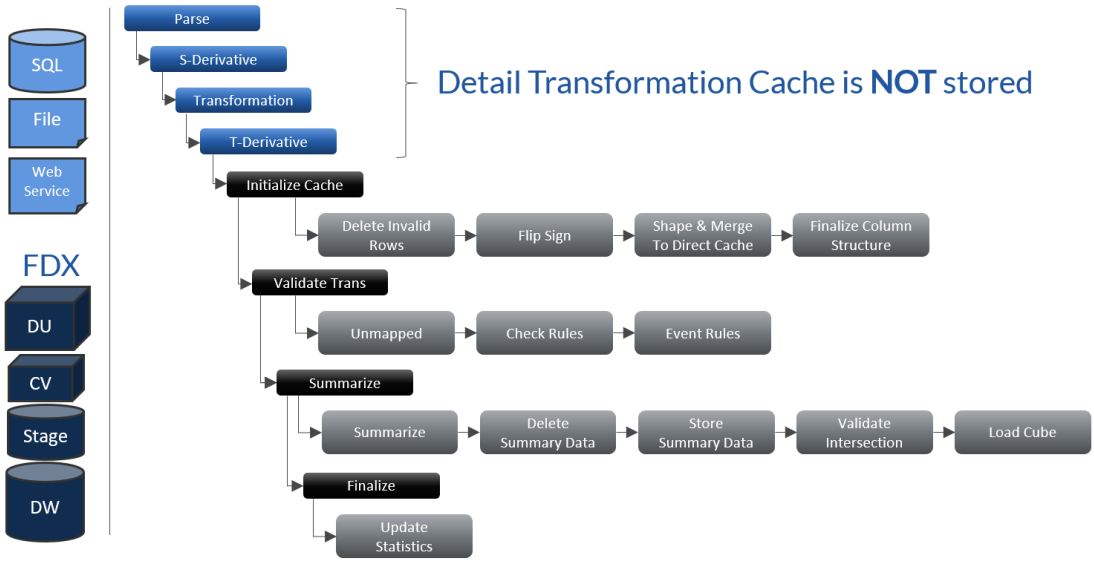
Integrations with high complexity and mapping may benefit by having a "development", Standard Workflow, to finalize the core Transformation Rules. A Standard Workflow supports pageable Validation and Intersection Error analysis, as well as the ability to Retransform source data that the Direct Load does not. The Standard Workflow also provides the Drill-Back from the Finance Engine to Stage that may streamline the data validation process. Once the core Transformation Rules are developed, a "production" Direct Load Workflow can be used, managing only the Validation exceptions.

Direct Load Workflow Implementation

The Direct Load Workflow Design and Requirements should consider strategies for configuration and data validation. Designers should consider the impact of the volume of source records and the complexity of mapping because of the Direct Load Workflow's lack of stored source and target details. Additionally, the Direct Load Transformation and Validation of records is limited to 1000 records per data load. Therefore, multiple imports may be required to resolve all mapping errors.

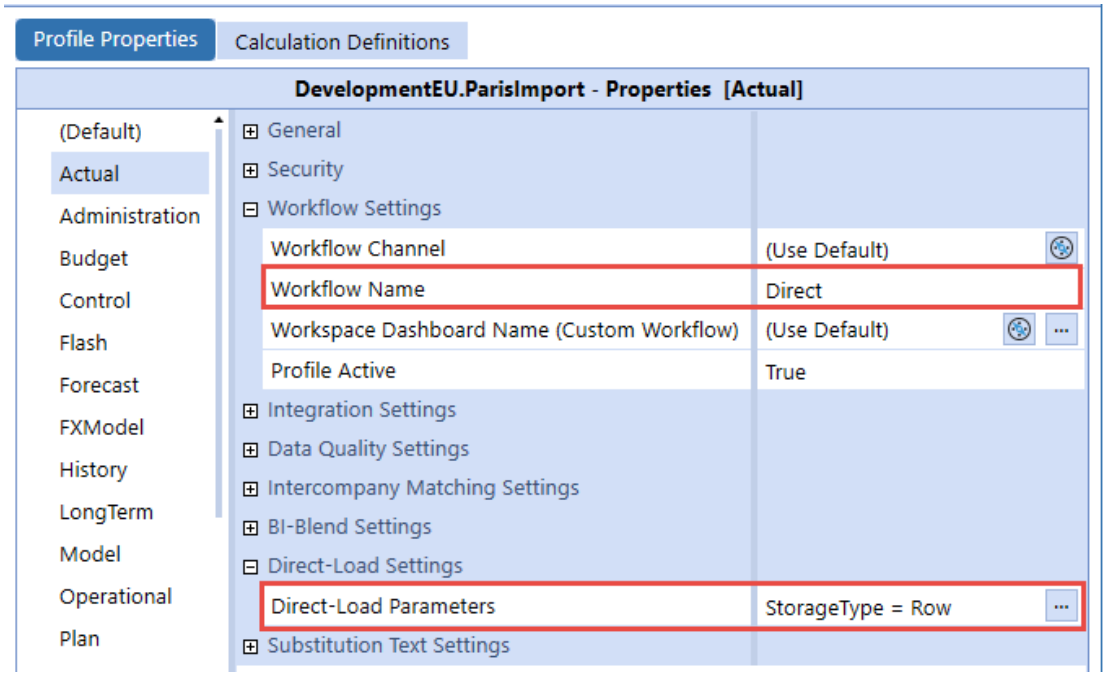
- Tightly coupled source metadata to OneStream metadata allows the use of “ * to * “ Mask maps to easily associate source data to cube data.
- Integrations with complex mapping may best make use of a Standard Workflow as “development” to provide transparency and as a platform for debugging.

Direct Load Data Flow



Direct Load Workflow Configuration

The Direct Load Type Workflow is an Import channel Workflow configured using various combinations of the “Direct” Workflow Name.



Row or Blob Summarized Data Storage

The Direct Load has two options for managing results of the Transformed records that are ultimately loaded to the Finance Engine, the default being Row.

- **Row**: Transformed summary records are stored in the StageSummaryTargetData table.
- **Blob**: Summarized records are not stored directly in the StageSummaryTargetData, but as a serialized Byte array stored in the StageDirectLoadInformation.

Sample Blob Resource Analysis

Rows	Size	Serialization & Write Time	RAM Utilization
12 Million	300-400 MB	2 Min 45 Sec	6 GB
24 Million	650-800 MB	5 Min	12 GB
36 Million	1 GB+	7 Min	21 GB

Direct Load Error and Troubleshooting

Direct Load Import Source File Data Keys must match the Workflow Data Keys. Unlike the Standard Workflow Import, the Direct Load Import cannot load data beyond the current Workflow. As an example, an Import to M1 cannot contain records for M2 in a Monthly Workflow.

Blend Workflow Import

The Blend Workflow Type utilizes the BI Blend Settings, which vary by Scenario Type. These settings allow the BI Blend Administrator to define, and optimize, the generation BI Blend tables to meet the reporting requirements.

The BI Blend Settings contain core properties used to design and structure the relational tables created by the BI Blend Engine.

BI Blend Settings Property Groups	Description
Data Controls	Defines the core data source and output structure and design of the relational tables.
Aggregation Controls	Settings to leverage Cube Dimension metadata to filter and define the relational tables.
Performance Controls	Server management and optimization settings.

BI Blend Use Cases

BI Blend is intended to provide focused reporting tables that are aggregated and saved as stored parent intersections for fast reporting at a later point in time. BI Blend is not intended to replicate and entire cube, but rather focus on specific reporting use cases that result in many parent intersections that would not perform well under Calc-On-Fly aggregation.

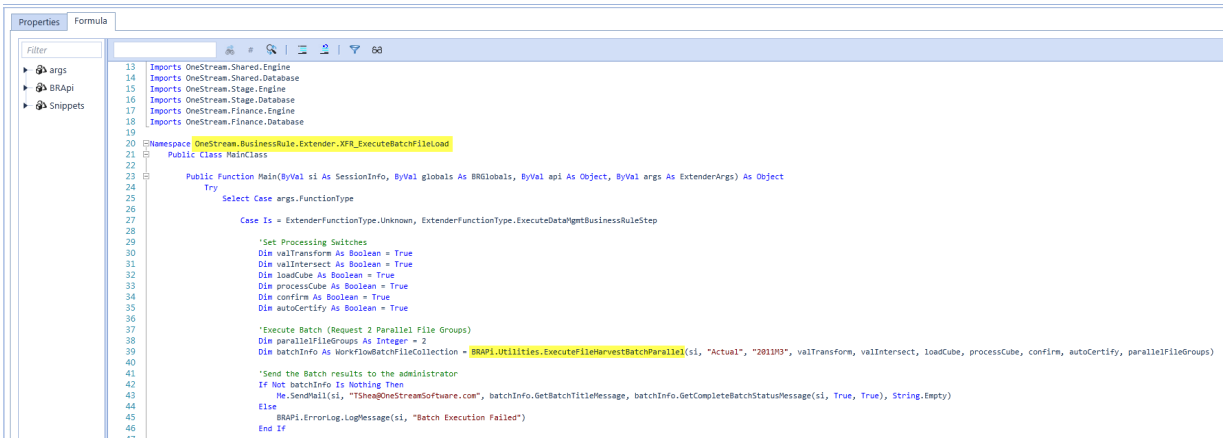
BI Blend also solves for use cases that are not pure analytic reporting problems. By leveraging OneStream hierarchies, along with BI Blend configuration settings, it is possible to aggregate on a few dimensions (Entity or Account as an example) while including transaction information (Invoice number) that is not associated with a cube. The ability to combine the dimensional structure with transaction details allows for selective enrichment of transactional data. Refer to the *BI Blend Design and Reference Guide* for more detail.

Batch File Loading

You can import and process files all the way through the Workflow certification process. In addition, as the Workflow batch process is executed, the batch processing engine raises events used to monitor the processing and notify administrators and users of the status of the batch.

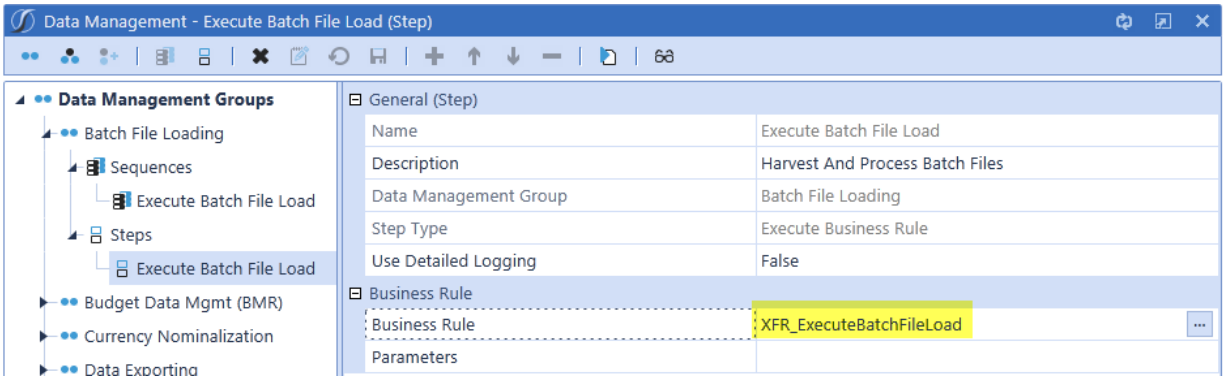
Setting Up and Using Batch File Loading

1. Create Batch Processing Extender Business Rule
   Batch file processing is executed by creating an Extender Business Rule that calls the OneStream API function BRAPi.Utilities.ExecuteFileHarvestBatch. This function also accepts switches that control the level of Workflow processing execution.



2. Create Data Management Sequence

   Batch file processing is executed by creating a Business Rule Data Management Step that calls an Extender Business Rule such as the rule example in step one.



3. Formatting File Names

   Batch processing requires each file being processed to use a specific file name format to tell the batch engine how to process the file (Format: File Id-ProfileName-ScenarioName-TimeName-LoadMethod.txt). See Batch File Name Format Specification later in this section for details on creating file names that comply with OneStream's required format.

4. Copy Files to Batch\Harvest Folder

   OneStream automatically creates a Batch\Harvest folder in each application's file share folder structure. This can be found in:

   System Tab|File Explorer|File Share|Applications|Application Name|Batch|Harvest

5. Execute the Batch

   After the files have been copied to the Harvest folder, execute the Data Management Sequence created in step one and the files will be processed.

   Manual Execution

   Navigate to Application Tab| Data Management, select the Batch Processing Data Management Sequence, and click to run the sequence.

6. Evaluate Batch Processing Results

   Task Activity Logging

   Each batch process creates a detailed Task Activity entry that provides overall status results for the batch as well as detailed information about each processed file and Workflow step.

   Batch Function Return Value

   In addition to writing information to the Task Activity Log, the batch processing function returns a detailed results object to the Extender Business Rule. This object provides information and can be programmatically evaluated and used to create custom reporting and notification.

7. Scheduling Batch Processing

   Batch file processing can run using the Windows Task Scheduler or any other scheduling tool an organization may use. This is accomplished by creating a PowerShell Script to execute a batch processing Data Management Sequence when called from the specific scheduling tool.

See Data Management Automation through PowerShell in Implementing Security for more information on executing OneStream Data Management Sequences from PowerShell scripts.

Batch File Name Format Specification

The information below provides a detailed list for each segment of the required batch file format.

Field Layout

File ID-ProfileName-ScenarioName-TimeName-LoadMethod.txt

aTrialBalance-Houston;Import-Actual-2011M1-R.txt

Field Definitions and Values

File ID

Any text value used for file identification and controlling sort order.

Profile Name

A valid Import Child Workflow Profile name. Use a ; to delimit Parent and Child Profile names.

Houston.Import becomes Houston;Import

Scenario Name

This is a valid Scenario name passed to Data Sources using the Dimension data type Current DataKey Scenario. C can be passed as a substitution variable to reference the Scenario name passed in the function call: HarvestAndProcessFiles. G can be passed as a substitution variable to reference the Global Scenario name set for the application.

Using Current Scenario: A-Houston;Import-C-2011M1-R.txt

Using Global Scenario: A-Houston;Import-G-2011M1-R.txt

Time Name

This is a valid Time name passed to Data Sources using the Dimension data type Current DataKey Time. C can be passed as a substitution variable to reference the Time name passed in the function call: HarvestAndProcessFiles. G can be passed as a substitution variable to reference the Global Time name set for the application.

Using Current Time: A-Houston;Import-Actual-C-R.txt

Using Global Time: A-Houston;Import-Actual-G-R.txt

Load Method

This is a value used to control how the file is loaded.

R = Replace, A = Append

Workflow Channels

Workflow Channels are a free form list of Members representing a logical grouping or binding point between an Input Child Workflow Profile and a specific set of accounts or the Members of a designated User Defined Dimension.

In addition, Workflow Channels are a mechanism used to increase the granularity of the standard Data Unit. They provide application designers with the ability to clear, load, and lock data at the intersection of accounts and the Members of a User Defined Dimension. See Data Units for more details.

There are three predefined default Workflow Channels when building an application: **Standard**, **NoDataLock**, and **AllChannelInput**. New metadata Members and new Workflow Profile Input Profiles are configured with default Workflow Channels. They have no effect on the granularity of application Data Units or the Workflow processes associated with clearing, loading, and locking Data Units.

Pre-build Workflow Channels

Standard
This is a basic Member without any special purpose other than to act as the default Workflow Channel for Account Members and Workflow Profile Input Children.

NoDataLock
This is a special Member that only applies to a metadata Member (Account or UDx) that should not participate in a Workflow Channel grouping scheme. Therefore, this Workflow Channel should not be assigned to Workflow Profiles but only to appropriate Metadata Members. This is the default value for any UDx Member.

Assigning this Member removes it from the Workflow Channel process and allows it to function with any Workflow Profile regardless of the Workflow Channel assigned to the Workflow Profile.

AllChannelInput
This is a special Member that only applies to a Workflow Profile Input Child (Import, Forms or Adj) and indicates the Workflow Profile can control data input processes for any Workflow Channel. AllChannelInput should not be assigned to any Metadata members, but rather used to signal that this Workflow is not participating in the Workflow Channel grouping scheme. By default, this Worklfow Channel is not assigned in the OneStream Application.

Using Workflow Channels

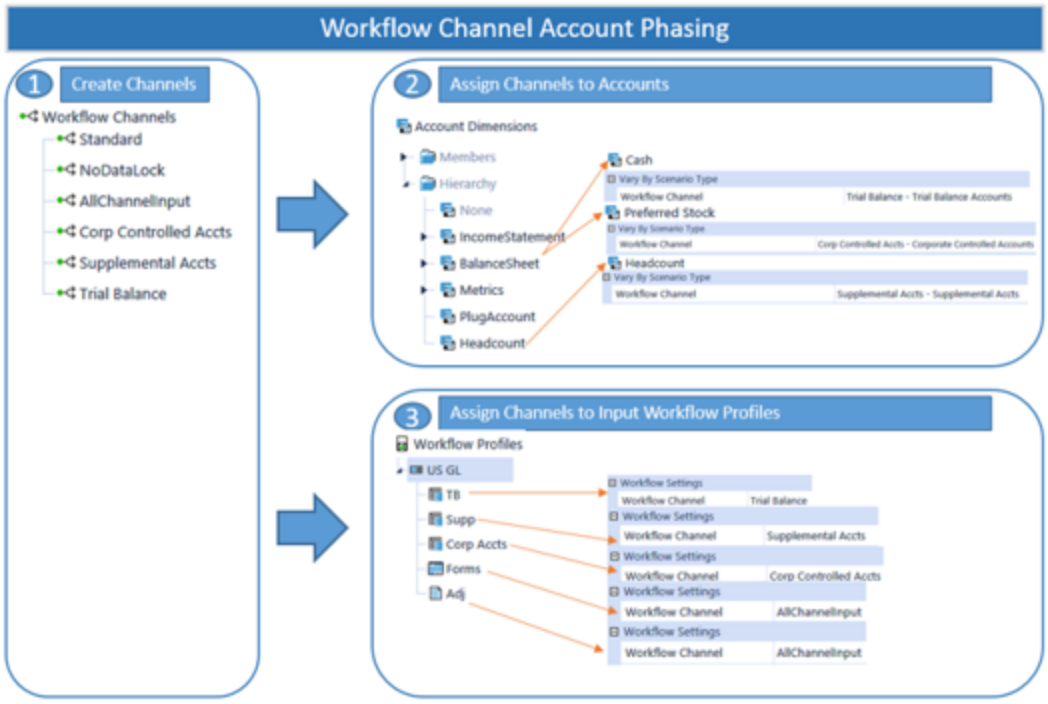
Workflow Channels allow the process to be locked down to a more granular level than the standard Workflow Profiles. This is an additional setting that can be configured to one additional Dimension. For example, this can lock down by product.

To set this, go to the **Application Tab** > **Tools** > **Application Properties**. See Workflow for more details.

Workflow Channels	
UD Dimension Type For Workflow Channels	(Not Used)

Workflow Channel Account Phasing

Using a combination of Workflow Channels and Accounts enables independent Workflow control to be applied to groups of accounts. The diagram below details the steps it takes to set up a metadata and Workflow structure that isolates process management for groups of Accounts and binds specific Workflow Profiles to control the care and feeding of these groups (data clearing, data loading, and data locking).

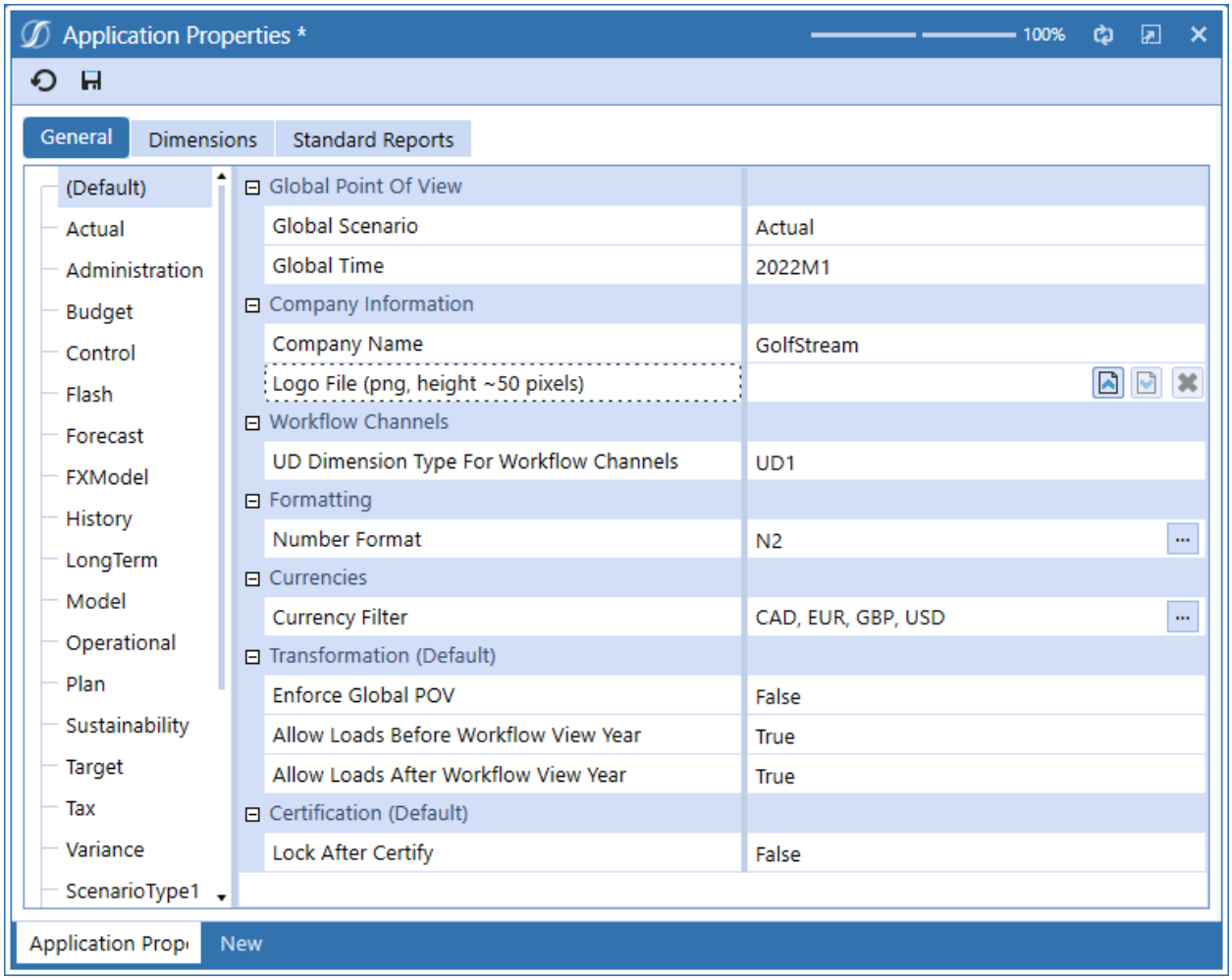


Setting Up Account Phasing

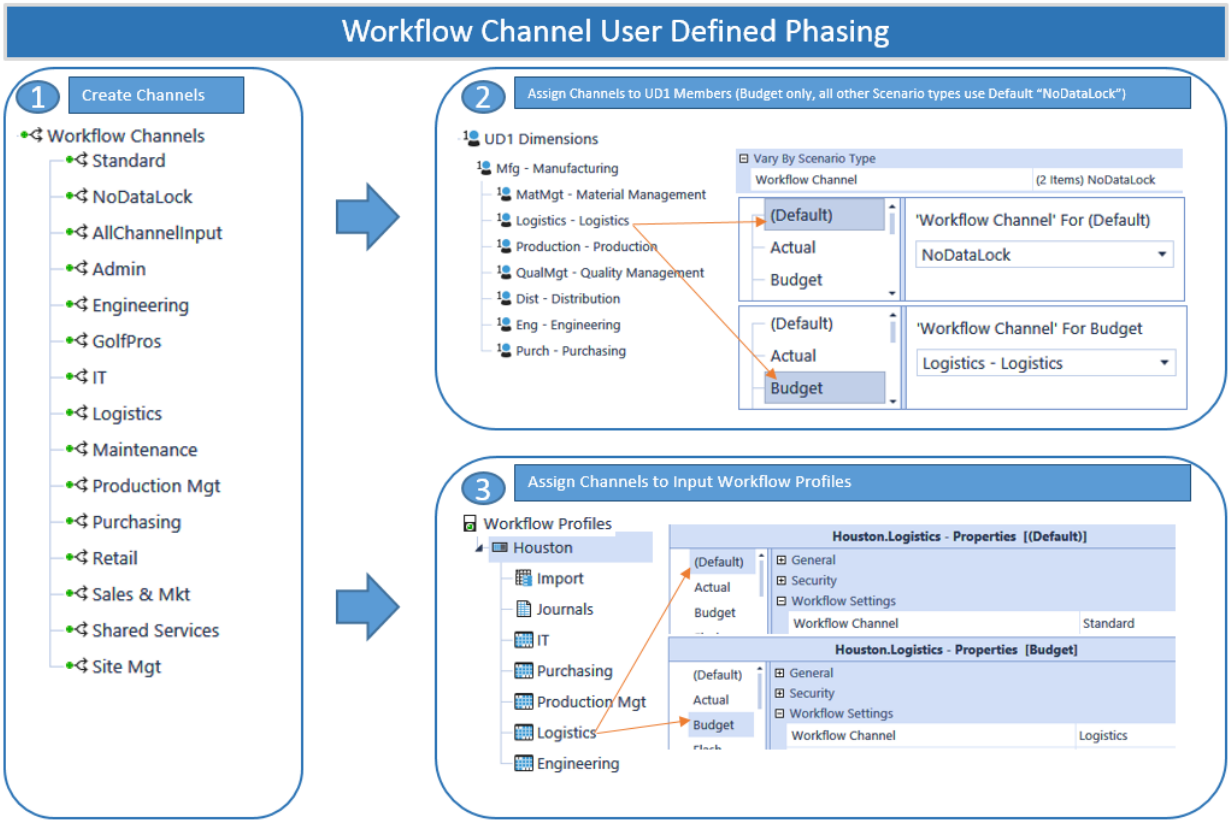
1. Create Workflow Channels to represent the groups of Accounts that should be linked together from a Workflow control perspective. In this example, three Workflow Channels have been created to provide separate control points for Basic Trial Balance Accounts, Corporate Controlled Accounts, and Statistical Accounts.
2. Tag each Account Member with the proper group to which it belongs. The Workflow Channel settings can vary by Scenario Type for both Metadata Members and Workflow Profile Members.
3. Tag each Workflow Profile Input Profile with the Workflow Channel it should control. This step hard wires the Workflow Profile to control data clearing, loading, and locking behaviors of the Metadata Members associated with the assigned Workflow Channel.

Workflow Channel User Defined Phasing

Using a combination of Workflow Channels and a specific User Defined Dimension enables independent Workflow Control to be applied to groups of User Defined Members. Before Workflow Channels can be used in conjunction with a User Defined Dimension, a single User Defined Dimension type (UD1-UD8) must be selected as the designated User Defined type to control Workflow Channel binding. This is done in Application Tab|Application Properties|User Defined Dimension Type for Workflow Channel. This selection is made at the application level and will apply to all Scenario Types and all Cubes within the application.



The diagram below details the steps to set up a metadata and Workflow structure that isolates process management for groups of User Defined Members and binds specific Workflow Profiles to control the care and feeding of these groups (data clearing, data loading, and data locking).



Workflow Channel Combined Account and User Defined Phasing

> **IMPORTANT:** When attempting to combine Workflow phasing using both Accounts and a User Defined Dimension, it requires the use of the NoDataLock Workflow Channel on Metadata Members and the use of the AllChannelInput Workflow Channel on Workflow Profiles.

Based on the examples above, if both Accounts and a User Defined Dimension are making use of Workflow Channel tagging, a situation can occur where the Workflow Channel assigned to a Workflow Profile is incompatible with either the Workflow Channel assigned to the Account Dimension or the one assigned to the User Defined Dimension.

This situation can be solved in one of two ways:

1. Assign the AllChannelInput Member to the Workflow Profile's Workflow Channel. This will allow the Workflow Profile to function in a more generic manner by limiting its usage to metadata Members tagged with a specific Workflow Channel. The only negative consequence of this approach is the Workflow locking for Workflow Profiles using this setting reverts to the Origin Member level which is less granular than the Workflow Channel level.
2. Sacrifice the Workflow Channel assignment of either the Account or the User Defined Dimension by assigning the NoDataLock Member. Assigning this Member will basically take it out of the Workflow Channel process and allow it to function with any Workflow Profile no matter what Workflow Channel is assigned to the Workflow Profile.

Using Workflow Channels Across Two Base Input Workflow Profiles

The purpose of a Workflow Channel is to bind an Input Workflow Child Profile to a set of cells in a Cube by Account and a specified User Defined Dimension Member. This relationship controls Cube Data Load/Replace granularity and cell locking granularity. If Workflow Channels were used across two Import Base Input Workflow Profiles, one importing a Trial Balance and one importing Supplemental data, and the Trial Balance Workflow Profile was locked, OneStream goes through an extensive process to check data overlap. In this case, there are two distinct Data Units defined by the Trial Balance and Supplemental Workflow Channels.

Step 1: Stage Load

The Trial Balance Workflow Channel data or the Supplemental Workflow Channel data is loaded and validated as usual. At the intersection validation step, will make sure that all intersections being processed belong to the correct Workflow Channel (Trial Balance or Supplemental). If not, the users will receive validation errors.

Step 2: Cube Load

When the Load Cube button is clicked for either Workflows, the following algorithm runs:

It checks to see if there are any other sibling Import Workflow Profiles. If there are, it will then check for overlapping Data Units within the proposed Stage load.

Workflow Channels in Use (One Channel per Workflow Import Child)
If the Workflow Channels are separate, only the data for the selected Workflow Profile will be written to the Cube. The data is only going into its own set of cells, so it does not need to consider the data in the other Workflow regardless if it is locked or not. Trial Balance and Supplemental can never overlap, the Workflow Channel guarantees it.

Workflow Channels Not in Use (or Same Channel Applied to Multiple Import Children)
In this case, it is possible that the two import siblings are attempting to load to the same intersections. Consequently, the Workflow Engine will evaluate the sibling import data content to determine if they overlap and are trying to write to the same data unit.

No Overlap
Load the Workflow Profile being processed because it is not overwriting sibling data.

Yes Overlap
Clear all data for all assigned Entities for the Import Origin Member. Next, reload the first Import Child Workflow using Replace. Then, reload the second, third, etc., Import Children in order, so the ultimate value in overlapped data units is the cumulative value from all Import Siblings.

Confirmation Rules

Confirmation Rules are used as a control to check the validity of the processed data. The rules can be setup to act as an error to the process or show a warning message. If the rule was setup as an error and it failed within the process, the user would not be able to proceed within the Workflow. The rules will process individually for each Entity associated with the Workflow Profile.

Confirmation Rules Properties

General

Name

The name of the Confirmation Rule group.

Description

A field for a more detailed description of the rule group.

Security

Access Group

This controls the user or users that have access to the rule group within the Workflow.

Maintenance Group

This controls the user or users that have access to maintain and administer the rule group.

> **NOTE:** Click in order to navigate to the Security screen. This is useful when changes need to be made to a Security User or Group before assigning it to a Confirmation Rule. Click 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. After the Group is selected, click CTRL and Double Click. This will enter the correct name into the appropriate field.

Settings

Scenario Type Name

The rule group can be made available to a specific Scenario Type or all Scenario Types.

Order

The order in which the rules will process and display within the group when the Workflow is processed.

Rule Name

A name given to the rule. This name will be seen in the Workflow, so it is best to give it a descriptive, purposeful name.

Frequency

This option will dictate how often the rule is required to run in the Workflow Profile.

All Time Periods

This runs the rules every period.

Monthly

This runs the rules every month. If this is for a weekly application, they will run the last week of each month.

Quarterly

This runs the rules every quarter, or four times a year.

Half Yearly

This runs the rules two times a year; once in June and December.

Yearly

This runs the rules once a year in December.

Member Filter

This turns on the Frequency Member Filter. Filters can then be defined in that section.

Frequency Member Filter

This only becomes available when Member Filter is chosen in the Forms Frequency options above, otherwise this will be grey. The purpose of this option is to allow the ability to filter by time.

Rule Text

A description of the rule. This will also be seen in the Workflow. The Rule Text should be a textual description of the Rule Formula associated with this rule.

Action

A drop down list containing the options Warning (Pass) or Error (Fail). If the data being evaluated does not pass the rule, these options dictate how to handle the problem. If a rule is associated with the Warning action, a warning message will display to alert the user, but the process will not stop, andyou will be allowed to Certify if there were no errors in other rules. If a rule is associated with the Error action, an error message will display, and the rule will have failed. You will not be able to proceed further until all failures have been addressed and/or resolved. If No Action is associated with the rule, the value for the given rule will just be displayed during confirmation. This data can be used by the user for informational purposes.

Failure Message

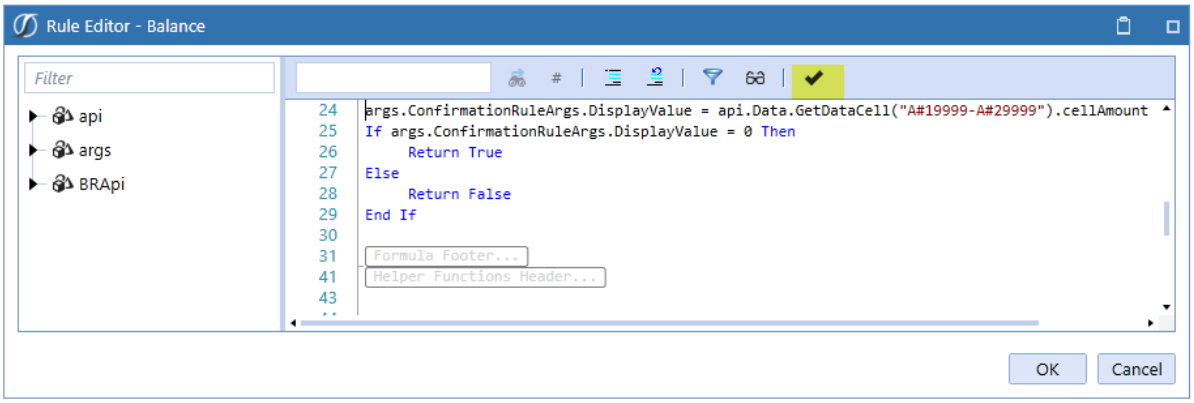
This message will be displayed to the user if an error occurs. This field will only be used if the rule is associated with the Error Action. The message should stipulate the error again and possible options of what the user should look for or do to resolve it.

Warning Message

This message will be displayed to the user if the line item was associated with the Warning Error Action. The warning will alert the user to the issue.

> **NOTE:** Messages longer than 2000 characters will be truncated with an ellipsis (…) placed in the last three characters.

> **TIP:** In order to edit the Rule Formula, right click on the line item that needs to be edited and select Edit Rule Formulas. This will open the Rule Formula Editor and the rule logic can be created or maintained.



Click the highlighted check icon, to ensure the formula was compiled successfully.

The right-click feature also allows the user to insert or delete a row, save, and export data.

Confirmation Rule Profiles

Once the Confirmation Rule Groups are created, they are organized into Confirmation Rule Profiles. The Profiles are then assigned to Workflow Profiles. Choose the icon to create a new Profile. To assign a Rule Group to a Profile, choose the icon. This allows the user to select which Groups will be in the Profile.

Under the Rule Profile Settings, choose the Cube Name and Scenario Type where this Profile can be viewed and used when designing a Workflow Profile. Assigning a Rule Profile to a Workflow Profile is done in the Application Tab| Workflow Profiles| Data Quality Settings section.

Certification Questions

Certification Question maintenance is the area where the repository of questions is maintained. The questions are answered by the assigned users and act as the certification to the data load process. Once the Certification Question Groups are created, they are organized into Certification Question Profiles. The Profiles are then assigned to Workflow Profiles. See Confirmation Rule Profiles .

Certification Questions Properties

General

Name
The name of the certification question group.

Description
The field for a more detailed description of the group.

Security

Access Group
This controls the user or users that have access to the rule group within the Workflow.

Maintenance Group
This controls the user or users that have access to maintain and administer the rule group.

NOTE: Click in order to navigate to the Security screen. This is useful when changes need to be made to a Security User or Group before assigning it to a Certification Question. Click 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, click CTRL and Double Click. This will enter the correct name into the appropriate field.

Settings

Scenario Type Name
The rule group can be made available to a specific Scenario Type or to all Scenario Types.

Order
The order in which the questions will appear to the user.

Name
A descriptive name given to the question.

Category
A drop down list of question types or categories. Select the best option for the question type.

Risk Level
Assign a risk level to the question. This will dictate the importance of the question as it pertains to being answered correctly.

Frequency
This option will dictate how often the question is required to be answered and when it will appear to the user.

All Time Periods
This displays the questions every period

Monthly
This displays the questions every month. If this is for a weekly application, they will display the last week of each month.

Quarterly
This displays the questions for every quarter, or four times a year.

Half Yearly
This displays the questions two times a year; once in June and December.

Yearly
This displays the questions once a year in December.

Member Filter
This turns on the Frequency Member Filter. Filters can then be defined in that section.

Frequency Member Filter
This only becomes available when Member Filter is chosen in the Forms Frequency options above, otherwise this will be gray. The purpose of this option is to allow the ability to filter by time.

Question Text
This is the question the user will answer. The question should be phrased to illicit a Yes or No response. The user is given a field to explain his/her answer in free text.

Response Optional
Check this box in order to make the question optional.

Deactivated
This will deem the question not active and it will not appear to the user any longer. All historical responses will be preserved if the question is not deleted.

Deactivated Date
Select the date the question was or is to be deactivated.

> **TIP:** By right clicking on any line item, a user can insert or delete a row, save, or export data.

Data Collection

Data sources are built to act as a blueprint for the types of imports that need to be done and to define how the data should be parsed and imported. These include fixed files and delimited files, which use connectors to pull the data directly from the source. Data sources also have dimensions associated with them along with specific properties. In this section you will learn about these data sources and how to leverage them to build your data collection.

OneStream recommends that any credentialed integration with the OneStream platform should change passwords regularly.

Data Sources

Data sources are blueprints of the types of imports required and define how to parse and import data. Data sources can be a fixed file, delimited file, connectors file, or a Data Management Export sequence that pull data from a source system. Data sources also have dimensions associated with them along with specific properties.

Once built, you can assign a data source to one workflow profile or to many workflow profiles sharing a common file format.

Associate a source file with a data source by clicking in the upper toolbar. The file opens in the top area of the screen and you can select fields and functions to build Data Source dimension definitions.

Data Source Properties

General properties are standard across all data types.

Name
The name of the data source.

Description
The field for a detailed description of the data source.

Security

The security properties are standard across all data types.

Access Group
Members of the assigned group have the authority to access the data source.

Maintenance Group
Members of the assigned group have the authority to maintain the data source.

> **NOTE:** Click ... 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, click CTRL double-click. This will enter the correct name into the appropriate field.

Settings

Settings are standard across all data types.

Cube Name
The cube associated with this data source which will dictate the available dimensions that can be used.

Scenario Type
This allows the profile to be assigned to a specific scenario type or all scenario types. If the data source is assigned to a specific Scenario Type, it will only be available when assigned to the Workflow Profile.

Data Structure Settings

Data structure settings are standard across all data types.

Type
This defines the source file structure. The Type can be Fixed, Delimited, Connector, or Data Management Export Sequence. Details on these types can be found below.

Data Structure Type

Tabular
This will have a line or lines specific to a single intersection with one amount.

Matrix
This will have multiple amounts on a given line using rows and columns to determine the intersection that corresponds to each amount.

Allow Dynamic Excel Loads
If set to True, users will be able to load Excel templates as well as the data file for which the data source has been built. See the section on Using Excel as a data source for information on the proper formatting of these files.

Fixed Files

Fixed files are in a column format with data in predefined columns.

Connector Settings

Connector Name
A drop down list of available Connector Type business rules. The rules will be built containing the code to connect to and pull data from a given source system or database.

Delimited Files

Delimited files have fields that are separated by a common character.

Delimited File Settings

Delimiter
The character being used to separate the distinct fields in the source file is defined here.

Quote Character
In delimited files, the fields are often put in quotes in case the delimiter is also a valid character in one of the field members. This option specifies what quote character is being used in the source file.

Connector Settings

Connector Name
A drop down list of available Connector Type business rules. The rules will be built containing the code to connect to and pull data from a given source system or database.

Connector Data Source

OneStream can connect and import data from any external system using direct database connection to the external system. This means data can be imported and processed all the way through the Workflow certification process without ever having to use a source system extract file.

Connector Settings

Connector Name
A drop-down list of available Connector Type business rules. The rules will be built containing the code to connect to and pull data from a given source system or database.

Connector Uses Files
Set to True to process files that cannot be parsed using a data source. For example, if an Excel or text file has complex formatting, this setting allows you to code the way the file should be imported rather than defining it with a data source.

Connector Business Rule

A Connector Business Rule defines the connection, data result sets, and drill-back option capabilities of an external data connection. A Connector functions as a Business Rule called by a Data Source and reveals what information is required from an external system. See Business Rules in Application Tools for an example of this rule.

Connector Data Source

Fields from the external data query results are mapped to Dimensions creating a processing behavior similar to the behavior of a Delimited File. Using this mapping process enables a Connector Data Source to use all the same built processing capabilities available with file-based Data Sources. This capability enables the design of an external data Connector to be entirely focused on connecting to and reading data from an external source instead of focusing on integrating complex business logic. The specific business logic can be added to the Data Source Dimensions in the form of a Complex Expression or Business Rule. This design methodology will help with writing the Connector Business Rule in a way that requires very little maintenance by business users.

Connector Information Request Types

GetFieldList

This is called by the Data Source designer screen when the user selects a Connector Data Source or one of its defined Dimensions. A list of available fields in the external Data Source will be visible as a list of Vb.Net Strings [List(Of String)] is requested.

GetData

This is called by the Import Workflow task when the Load and Transform button is clicked. The execution of a data query(s) that retrieves the row values for the chosen Workflow Unit is requested.

Fields

The field names returned by this query must match the field names returned by the GetFieldList request.

Where Clause

Typically the active Workflow Unit Time and Scenario values are converted to equivalent criteria values for the Time or Scenario of the external system.

Data Volume

Consider loading summarized data rather than full transaction system data replication because drill back is provided for more detailed values.

GetDrillBackTypes

Drill Back types can deliver results based on the different visualization types. This is called when a user double-clicks or right-clicks and selects Drill Back from a row in the source data load or drill down screens. A set of supported drill-back options to present to the end user as a list of DrillBackTypeInfo objects [List(Of DrillBackTypeInfo)] is requested. Drill Back types provide the Connector designer with the power to provide the end user with a menu list of drill back options.

DataGrid

This presents a grid of data rows to the end user.

TextMessage
This presents a text message to the end user.

WebUrl
This presents a website or custom HTML web content to the end user.

WebUrlPopOutDefaultBrowser
Opens a website or custom HTML web content in an external browser. From the Stage Import data grid, right-click on a data record, and select Drill Back. A dialog presents a menu of pre-configured Drill-back options. When you choose WebUrlPopOutDefaultBrowser, a standard browser session is launched, and you go to a web page based on variables.

FileViewer
This presents file contents to the end user from one of three locations.

FileShareFile
A file located in a folder in the OneStream File Share.

AppDBFile
A file stored in an application database.

SysDBFile
A file stored in a framework (System) database.

GetDrillBack
This is called when a user selects a specific Drill Back type presented by the GetDrillBackTypes request. When this action is executed, the Business Rule arguments will contain a reference to the DrillBackTypeInfo object the user selected which allows the Connector designer to determine how to get proper information to display for the DrillBackTypeInfo.

Connector Integration Prerequisites

The following items provide an overview of the major technology components involved in integrating external systems with deployment.

Determine Source System Inventory

The first step in integrating various source systems is to determine all the ones needed. This includes:

Source System Location & Identification

Database Type and Source System

Oracle, SQL, DB2, Syteline, Newstar, Lawson, PeopleSoft, Access, MAS500, etc.

> **NOTE:** The requirement for Oracle Database integrations is that all Oracle Source System TNS Profile details need to be in place on each of the OneStream application servers.

Data Query Method

Detailed Data Query, Data View, Stored Procedure, etc.

Source System Drill Back Criteria (if required)

Detailed Data Query, Data View, Stored Procedure, etc.

Source System Direct Access Credentials

A read-only type of access needs to be granted for the user account because the data from these external systems will be read. The read-only access should be granted against the productions instance of the data source as the data queries will be used to tie out data and do not present any risk to the source system themselves.

Source System 64-bit Client Data Provider

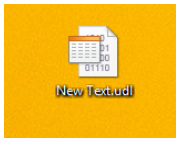
OneStream is a Microsoft .NET application with a 64-bit architecture. To communicate with any source system, a 64-bit source system client data provider needs to be available and installed on each OneStream application server. The source system's client data provider is what gives the ability to make an OLEDB or ODBC connection to the system.

Determine Connection String

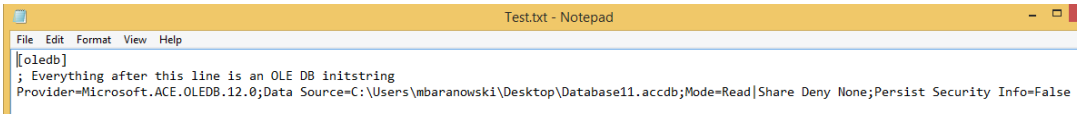
A connection string specifies information about a data source and the means of connecting to it. It is passed in code to an underlying driver or provider to initiate the connection. Whilst commonly used for a database connection, the data source could also be a spreadsheet or text file. The connection string may include attributes such as the name of the driver, server and database, as well as security information such as user name and password.

Create a Connection String from the OneStream Application Server

1. Right-click the Desktop icon of the OneStream application server and select **New > Text Document**.
2. Name the document and change the file extension from txt to udl.

   This creates a Data Link File to assist in the formation of the source system connection string.
3. Determine the DB Provider that the GL Source System is using (e.g. SQL, Oracle, etc.).
4. Determine the server name where the data resides for the GL Source System.
5. Determine the user name and password used to connect to the server for the GL Source System.
6. Determine the database name on the server where the GL Source System data resides.
7. Save the completed UDL file and then rename the extension back to txt from udl.
8. Open the text file to see the connection string provided.



Example Connection Strings

SQL Server

Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=DBName;Data Source=SQLSERVERNAME

ORACLE (11i or R12)

Provider=OraOLEDB.Oracle.1;Password=<xxxxx>;Persist Security Info=True;User ID=<username>;Data Source=frepro.world

DB2

Provider=IBMDA400.DataSource.1;Password=<xxxxx>;Persist Security Info=True;User ID=OSuser;Data Source=HUTCH400;Use SQL Packages=True

MS Access

Provider=Microsoft.ACE.OLEDB.12.0;Data Source=\\UNCFileShare\DB1.accdb;Mode=Read|Share Deny None;Persist Security Info=False

Determine the Data Query Method

To extract data from any source system, the data query method and facility need to be determined. Data can be queried through a SQL Query, a SQL View, or Stored Procedure. OneStream executes this request against the source system using the defined source system connection string and processes the returned results within OneStream.

For example, if directly pulling in Trial Balance Data is required, then the detailed query that currently makes up the existing Trial Balance Report would be necessary for OneStream to pull the same data.

SQL Query

A SQL Query can be broken down into numerous elements, each beginning with a keyword. Although it is not necessary, a common convention is to write these keywords in all capital letters. The standard sections of a SQL Query are made up of the following four elements:

SELECT

FROM

WHERE

ORDER BY

The example below is a SQL Query used to pull Trial Balance Data from several different tables in an Oracle Database:

```
SELECT
GL_SETS_OF_BOOKS.NAME
,GL_BALANCES.ACTUAL_FLAG
,GL_BALANCES.PERIOD_NAME
,GL_BALANCES.PERIOD_NUM
,GL_BALANCES.PERIOD_YEAR
,GL_CODE_COMBINATIONS.CODE_COMBINATION_ID
,GL_CODE_COMBINATIONS.SEGMENT1
,GL_CODE_COMBINATIONS.SEGMENT2
,GL_CODE_COMBINATIONS.SEGMENT3
,GL_CODE_COMBINATIONS.SEGMENT4
,GL_CODE_COMBINATIONS.SEGMENT5
,GL_CODE_COMBINATIONS.SEGMENT6
,GL_CODE_COMBINATIONS.SEGMENT7
,GL_CODE_COMBINATIONS.SEGMENT8
,GL_CODE_COMBINATIONS.SEGMENT9
,GL_CODE_COMBINATIONS.SEGMENT10
,SUM( NVL(GL_BALANCES.BEGIN_BALANCE_DR,0) - NVL(GL_BALANCES.BEGIN_BALANCE_CR,0))"OPEN BAL"
,NVL(GL_BALANCES.PERIOD_NET_DR,0) "DEBIT"
,NVL(GL_BALANCES.PERIOD_NET_CR,0) "CREDIT"
,SUM( NVL(GL_BALANCES.PERIOD_NET_DR,0) - NVL(GL_BALANCES.PERIOD_NET_CR,0))"NET MOVEMENT"
,SUM(( NVL(GL_BALANCES.PERIOD_NET_DR,0) + NVL(GL_BALANCES.BEGIN_BALANCE_DR,0))) - SUM
       (NVL(GL_BALANCES.PERIOD_NET_CR,0)+NVL(GL_BALANCES.BEGIN_BALANCE_CR,0))"CLOSE BAL"
,GL_BALANCES.CURRENCY_CODE
```

```
,GL_BALANCES.TRANSLATED_FLAG
,GL_BALANCES.TEMPLATE_ID
,FND_FLEX_VALUES_VL.FLEX_VALUE
,FND_FLEX_VALUES_VL.DESCRIPTION
,FND_FLEX_VALUES_VL.FLEX_VALUE_SET_ID

FROM
GL_BALANCES,
GL_CODE_COMBINATIONS,
GL_SETS_OF_BOOKS,
FND_FLEX_VALUES_VL

WHERE GL_CODE_COMBINATIONS.CODE_COMBINATION_ID = GL_BALANCES.CODE_COMBINATION_ID
AND GL_BALANCES.ACTUAL_FLAG = 'A'
AND GL_BALANCES.CURRENCY_CODE = GL_SETS_OF_BOOKS.CURRENCY_CODE
AND GL_BALANCES.LEDGER_ID = GL_SETS_OF_BOOKS.SET_OF_BOOKS_ID
AND GL_BALANCES.TEMPLATE_ID IS NULL
AND GL_BALANCES.PERIOD_NAME = 'Jul-14'
AND FND_FLEX_VALUES_VL.FLEX_VALUE = GL_CODE_COMBINATIONS.SEGMENT4
AND FND_FLEX_VALUES_VL.FLEX_VALUE_SET_ID = '101432874'
AND GL_CODE_COMBINATIONS.SEGMENT2 IN (2050, 2100, 2200, 2300, 2400, 2500)

GROUP BY  GL_SETS_OF_BOOKS.NAME
,GL_BALANCES.ACTUAL_FLAG
,GL_BALANCES.PERIOD_NAME
,GL_BALANCES.PERIOD_NUM
,GL_BALANCES.PERIOD_YEAR
,GL_CODE_COMBINATIONS.CODE_COMBINATION_ID
,GL_CODE_COMBINATIONS.SEGMENT1
,GL_CODE_COMBINATIONS.SEGMENT2
,GL_CODE_COMBINATIONS.SEGMENT3
,GL_CODE_COMBINATIONS.SEGMENT4
,GL_CODE_COMBINATIONS.SEGMENT5
,GL_CODE_COMBINATIONS.SEGMENT6
,GL_CODE_COMBINATIONS.SEGMENT7
,GL_CODE_COMBINATIONS.SEGMENT8
,GL_CODE_COMBINATIONS.SEGMENT9
,GL_CODE_COMBINATIONS.SEGMENT10
```

```
,NVL(GL_BALANCES.PERIOD_NET_DR,0)
,NVL(GL_BALANCES.PERIOD_NET_CR,0)
,GL_BALANCES.CURRENCY_CODE
,GL_BALANCES.TRANSLATED_FLAG
,GL_BALANCES.TEMPLATE_ID
,FND_FLEX_VALUES_VL.FLEX_VALUE
,FND_FLEX_VALUES_VL.DESCRIPTION
,FND_FLEX_VALUES_VL.FLEX_VALUE_SET_ID
HAVING SUM(( NVL(GL_BALANCES.PERIOD_NET_DR,0) + NVL(GL_BALANCES.BEGIN_BALANCE_DR,0))) -
       SUM(NVL(GL_BALANCES.PERIOD_NET_CR,0)+NVL(GL_BALANCES.BEGIN_BALANCE_CR,0)) <> 0
```

SQL View

In many cases, creating a SQL View of data to provide information to OneStream is a more preferred option and typically simplifies the complexity of the query.

In the example below, the customer can combine several data tables required in the source system, and present the data in one View for OneStream to query:

```
SELECT
SEGMENT1 As Entity
SEGMENT2 As Establishment
SEGMENT3 As France_Account
SEGMENT4 As US_Account
SEGMENT5 As Cost_Center
SEGMENT6 As Family
SEGMENT7 As Product_Line
SEGMENT8 As Interco
SEGMENT9 As Future
PERIOD_YEAR As Year
PERIOD_MONTH As Month
CURRENCY_CODE As Currency_Code
CLOSE_NET_BALANCE As Net_Balance
SET_OF_BOOKS_ID As Set_Of_Books_ID
FROM APPS.XXSWM_ONESTREAM_GL_BALANCES
```

Stored Procedure

The example below is a SQL Stored Procedure used to pull Trial Balance Data from several different tables in a SQL Database.

In this example, the Entity, Year, and Period are passed to the Stored Procedure:

```
spGLCalcTrialBalance 'ASCC', '2013', 6
```

Apply Connection String to XFAppServerConfig.xml File

When the connection string is created, then the database connections can be centralized in the Server Configuration under the App Server Configuration File. Under Databases, click on (Collection) for Database Server Connections and the Database Server Connections will appear. The string will then be placed in the Connection String under Connection String Settings. The name of the connection string will be used as part of the source connector.

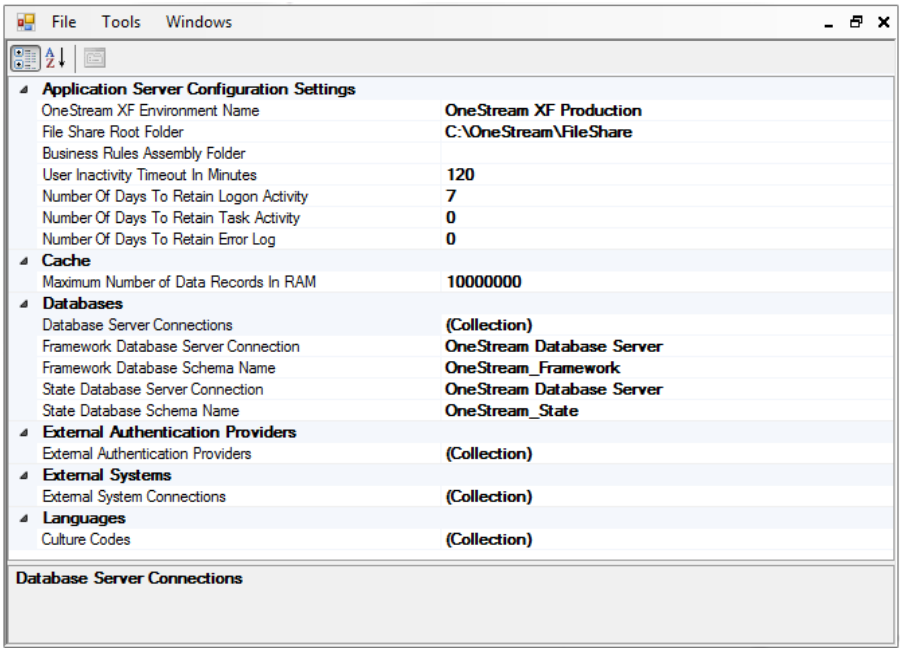
Defining External Data Connections

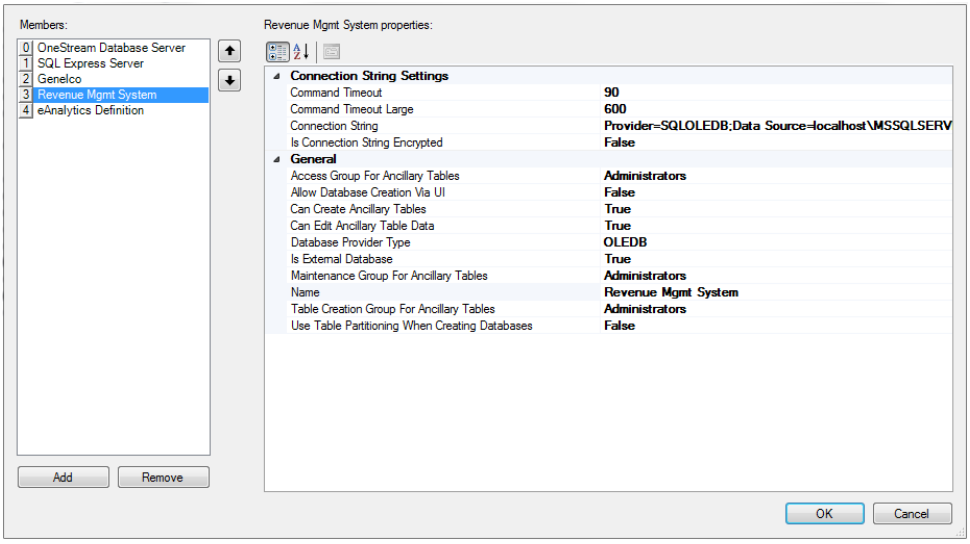
Application Server Configuration File
Creating Named External ODBC / OLEDB Connection

Step 1: Required ODBC/OLEDB Connection Software
Any client ODBC/OLEDB drivers must be installed on each application server for the OneStream application to make a connection to the external database. This way the administrator knows what type of database engine contains which Data Source.

Step 2: Creating the Connection String
The application server configuration file must be modified to add a named external database connection that can be used by the Connector Business Rule and custom reports.

Example of the Server Configuration Utility:





Step 3: Creating an External Database Test Query

The best way to prototype the queries needed to create a Connector Business Rule is to create a set of Dashboard Data Adapters to be used as a test bed.

As a best practice, create a new Dashboard Maintenance Unit named EXS The Connector Name. The prefix EXS stands for External System and will provide administrators with an immediate understanding of the Maintenance Unit's contents. The three steps below explain how to create this.

Step 1

Create a new Data Adapter for each type of query needed to proto type (GetFieldList, SelectData, Drill Back, etc.)

Example of a Data Adapter being used to get all fields in the source table of the external database connection:

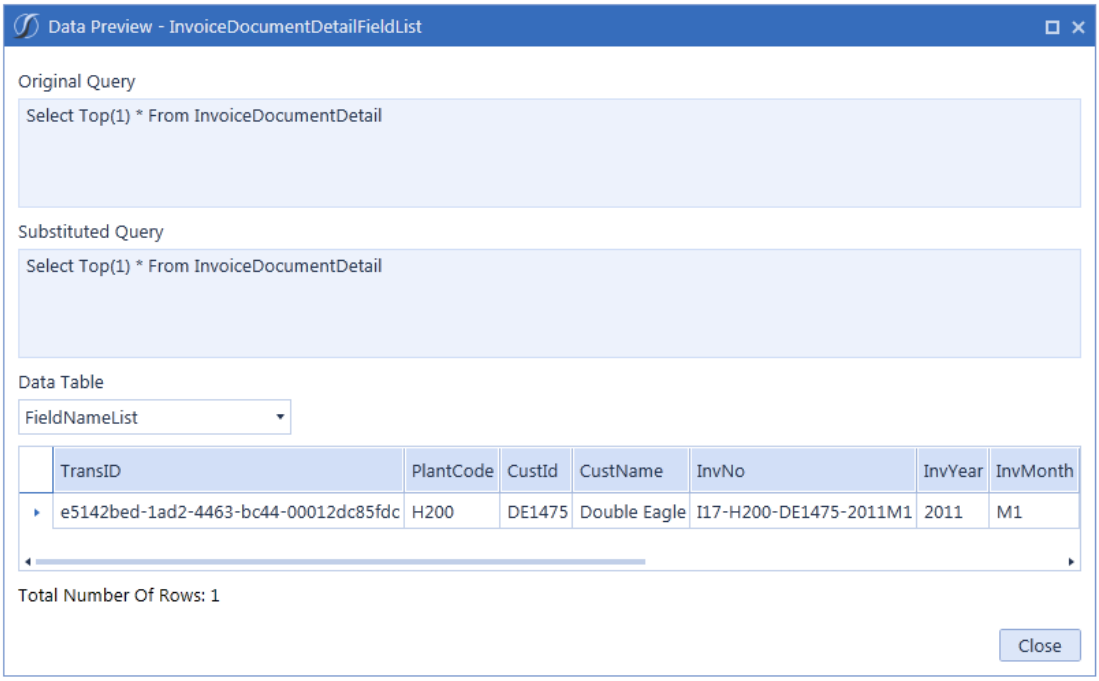
⊟ General (Data Adapter)	
Name	InvoiceDocumentDetailFieldList
Description	List Of Fields in Source Table
Maintenance Unit	External System (Revenue Mgmt Houston)
⊟ Data Source	
Command Type	SQL
Database Location	External
External Database Connection	Revenue Mgmt System
SQL Query	Select Top(1) * From InvoiceDocumentDetail
Results Table Name	FieldNameList

Step 2

Click in the Dashboard administration toolbar to test the query.

Step 3

Evaluate the results of the query. The Data Adapter test only returns a small subset of rows from the query, but it specifies the actual number of rows that will return during an actual query execution.



Building Data Connectors

ODBC / OLEDB Connectors

GetFieldList

Select Query against the external database. There will be a manual list of strings returned for each field.

GetData

The selected statement should match GetFieldList. Add criteria for Scenario and Time and map the OneStream Workflow Unit Scenario and Time values to corresponding values in the source system as a Where Clause criteria value.

GetDrillBackTypes

This shows the set of drill back options provided to the user.

GetDrillBack

This executes the selected drill back type for the current source data row.

Custom API Connectors

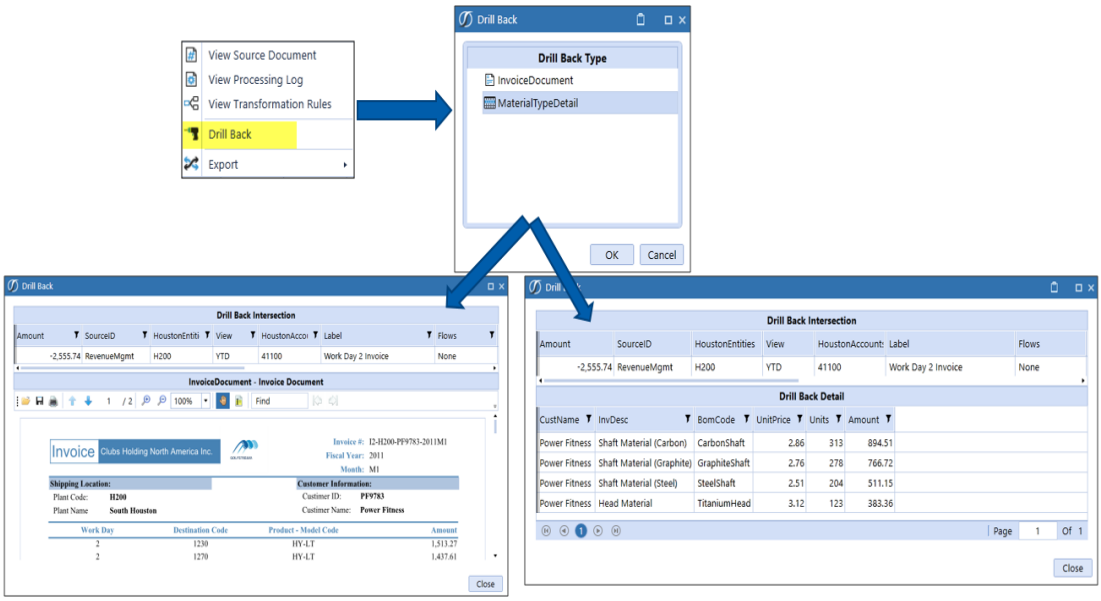
Uses OneStream's External Server Technology. IIS needs to be recycled on all application servers and followed by all web servers after adding the external named connection.

Drill Back

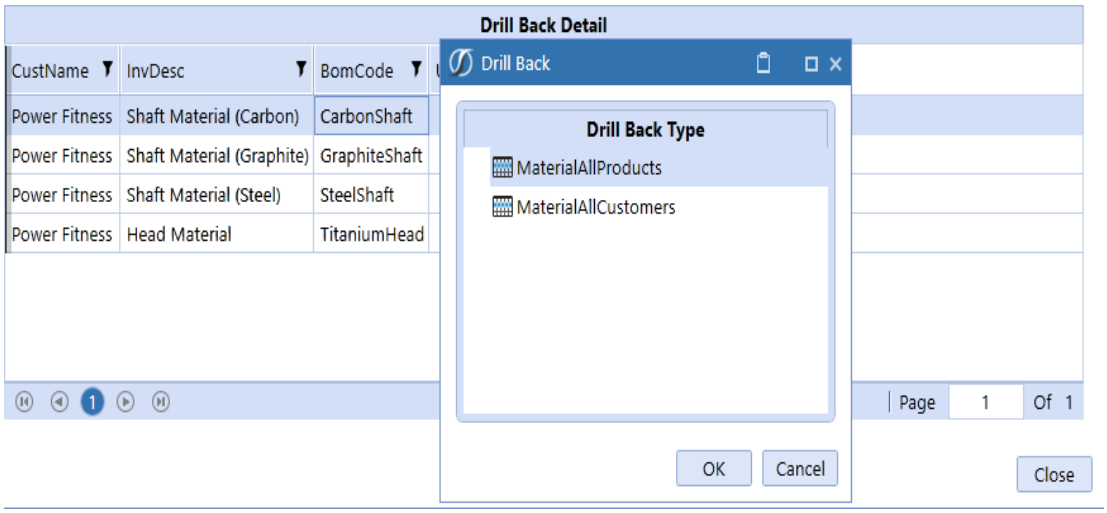
Using a SQL connector allows a user to drill back to a source system and show detailed records from a document, PDF, website. The Connector Data Source, configured by the author, provides a menu of data viewing options such as Year to Date, Month to Date, Invoice Documents or Material Type Detail. Utilizing this feature can reduce the amount of data imported into the Financial Model by allowing analysis to occur at the source system.

Viewing Data

After data is loaded into the Stage, a user can right-click on a data row and select Drill Back. This will bring up the pre-configured options from which the user can choose.



If more detail is needed, another level of Drill Back can be performed. This is configured in the Connector Business Rule and can drill back and around source systems. These nested drill paths can provide as much detail as an application requires.



Key API, Args, or BRAPI Examples

```
Namespace OneStream.BusinessRule.Connector.RevenueMgmtHouston
    Public Class MainClass
        Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As ConnectorArgs) As Object
            Try
                'Get the query information
                Dim connectionString As String = GetConnectionString(si, globals, api)

                'Get the Field name list or load the data
                Select Case args.ActionType
                    Case Is = ConnectorActionTypes.GetFieldList
                        'Return Field Name List
                        Dim fieldListSQL As String = GetFieldListSQL(si, globals, api)
                        Return api.Parser.GetFieldNameListForSQLQuery(si, DbProviderType.OLEDB, connectionString, true, fieldListSQL, false)

                    Case Is = ConnectorActionTypes.GetData
                        'Process Data
                        Dim sourceDataSQL As String = GetSourceDataSQL(si, globals, api)
                        api.Parser.ProcessSQLQuery(si, DbProviderType.OLEDB, connectionString, true, sourceDataSQL, false, api.ProcessInfo)
                        Return Nothing

                    Case is = ConnectorActionTypes.GetDrillBackTypes
                        'Return the list of Drill Types (Options) to present to the end user
                        Return  Me.GetDrillBackTypeList(si, globals, api, args)

                    Case Is = ConnectorActionTypes.GetDrillBack
                        'Process the specific Drill-Back type
                        Return Me.GetDrillBack(si, globals, api, args, args.DrillBackType.DisplayType, connectionString)
                End Select

            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
```

```
        'Create a Connection string to the External Database
        Private Function GetConnectionString(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer) As String
            Try

                'Connection String Method
                '------------------------------------------------------------
'               Dim connection As New Text.StringBuilder'
'               connection.Append("Provider=SQLOLEDB.1;")
'               connection.Append("Data Source=LocalHost\MSSQLSERVER2008;")
'               connection.Append("Initial Catalog=SampleData;")
'               connection.Append("Integrated Security=SSPI")
'               Return connection.ToString

                'Named External Connection
                '------------------------------------------------------------
                Return "Revenue Mgmt System"

            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
```

```
    'Create the field list SQL Statement
    Private Function GetFieldListSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer) As String
        Try
            'Create the SQL Statement
            Dim sql As New Text.StringBuilder

            sql.Append("SELECT Top(1)")
            sql.Append("TransID, PlantCode, CustId, CustName, InvNo, InvYear, InvMonth, InvDesc, GLAccount, WorkDay, ProdModel, BomCode, UnitPrice, Units, Amount, DestinationCode
            sql.Append("FROM InvoiceDocumentDetail ")

            Return sql.ToString

        Catch ex As Exception
            Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
        End Try
    End Function
```

```
    'Create the data load SQL Statement
    Private Function GetSourceDataSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer) As String
        Try
            'Create the SQL Statement
            Dim statement As New Text.StringBuilder
            Dim selectClause As New Text.StringBuilder
            Dim fromClause as New Text.StringBuilder
            Dim whereClause as New Text.StringBuilder
            Dim orderByClause as New Text.StringBuilder

            selectClause.Append("SELECT ")
            selectClause.Append("TransID, PlantCode, CustId, CustName, InvNo, InvYear, InvMonth, InvDesc, GLAccount, WorkDay, ProdModel, BomCode, UnitPrice, Units, Amount, Destina

            fromClause.Append("FROM InvoiceDocumentDetail ")

            whereClause.Append("WHERE ")
            'Get the YEAR from the current XF workflow unit TimeKey
            whereClause.Append("(")
            whereClause.Append("InvYear = " & TimeDimHelper.GetYearFromId(api.WorkflowUnitPk.TimeKey).ToString)
            whereClause.Append(")")

            'Get the MONTH from the current XF workflow unit TimeKey
            whereClause.Append(" And ")
            whereClause.Append("(")
            whereClause.Append("InvMonth = 'M" & TimeDimHelper.GetSubComponentsFromId(api.WorkflowUnitPk.TimeKey).Month.ToString & "'")
            whereClause.Append(") ")

            'Select Houston Plant Codes
            whereClause.Append(" And ")
            whereClause.Append("(")
            whereClause.Append("PlantCode IN('H200','H210')")
            whereClause.Append(") ")

            orderByClause.Append("ORDER BY ")
            orderByClause.Append("PlantCode, CustId, WorkDay, ProdModel, DestinationCode")

            'Create the full SQL Statement
            statement.Append(selectClause.ToString)
            statement.Append(fromClause.ToString)
            statement.Append(whereClause.ToString)
            statement.Append(orderByClause.ToString)
```

```
'Create the drill back options list
Private Function GetDrillBackTypeList(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As ConnectorArgs) As List(Of DrillBackTypeInfo)
    Try
        'Create the SQL Statement
        Dim drillTypes As New List(Of DrillBackTypeInfo)

        drillTypes.Add(New DrillBackTypeInfo(ConnectorDrillBackDisplayTypes.FileShareFile, New NameAndDesc("Invoice Document","Invoice Document")))
        drillTypes.Add(New DrillBackTypeInfo(ConnectorDrillBackDisplayTypes.DataGrid, New NameAndDesc("Material Type Detail","Material Type Detail")))

        Return drillTypes

    Catch ex As Exception
        Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
    End Try
End Function

'Execute specific drill back type
private Function GetDrillBack(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As ConnectorArgs, ByVal drillBackType As ConnectorDrillBackDisplayTypes, ByVal connectionStr
    Try
        Select case drillBackType
            case is = ConnectorDrillBackDisplayTypes.FileShareFile
                'Show FileShare File
                Dim drillBackInfo as new DrillBackResultInfo
                drillBackInfo.DisplayType = ConnectorDrillBackDisplayTypes.FileShareFile
                drillBackInfo.DocumentPath = Me.GetDrillBackDocPath(si, globals, api, args)
                return drillBackInfo

            case is = ConnectorDrillBackDisplayTypes.DataGrid
                'Return Drill Back Detail
                Dim drillBackSQL As String = GetDrillBackSQL(si, globals, api, args)
                Dim drillBackInfo as new DrillBackResultInfo
                drillBackInfo.DisplayType = ConnectorDrillBackDisplayTypes.DataGrid
                drillBackInfo.DataTable = api.Parser.GetXFDataTableForSQLQuery(si, DbProviderType.OLEDB, connectionString, true, drillBackSQL, false, args.PageSize, args.PageNumber)
                return drillBackInfo

            case else
                return Nothing
        End Select

    Catch ex As Exception
        Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
    End Try
End Function
```

```
 'Create the drill back Document Path
 Private Function GetDrillBackDocPath(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As ConnectorArgs) As String
     Try
         'Get the values for the source row that we are drilling back to
         Dim sourceValues as Dictionary(Of string, Object) = api.Parser.GetFieldValuesForSourceDataRow(si, args.RowID)
         If (Not sourceValues Is Nothing) And (sourceValues.Count > 0) then
             Return "Applications/GolfStream_v24/DataManagement/RevenueMgmtInvoices/" & sourceValues.Item(StageConstants.MasterDimensionNames.Attribute1).ToString & ".pdf"
         Else
             Return String.Empty
         End If
     Catch ex As Exception
         Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
     End Try
 End Function
```

```
'Create the drill back SQL Statement
Private Function GetDrillBackSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As ConnectorArgs) As String
    Try
        'Get the values for the source row that we are drilling back to
        Dim sourceValues as Dictionary(Of string, Object) = api.Parser.GetFieldValuesForSourceDataRow(si, args.RowID)
        If (Not sourceValues Is Nothing) And (sourceValues.Count > 0) then

            Dim statement As New Text.StringBuilder
            Dim selectClause As New Text.StringBuilder
            Dim fromClause as New Text.StringBuilder
            Dim whereClause as New Text.StringBuilder
            Dim orderByClause as New Text.StringBuilder

            'Create the SQL Statement
            selectClause.Append("SELECT ")
            selectClause.Append("CustName, InvDesc, BomCode, UnitPrice, Units, Amount ")

            fromClause.Append("FROM InvoiceMaterialDetail ")

            whereClause.Append("WHERE ")
            'Get the YEAR from the source recrod
            whereClause.Append("(")
            whereClause.Append("InvYear = " & TimeDimHelper.GetYearFromId(sourceValues.Item(StageTableFields.StageSourceData.DimWorkflowTimeKey).ToString))
            whereClause.Append(")")

            'Get the MONTH from the source record
            whereClause.Append(" And ")
            whereClause.Append("(")
            whereClause.Append("InvMonth = 'M" & TimeDimHelper.GetSubComponentsFromId(sourceValues.Item(StageTableFields.StageSourceData.DimWorkflowTimeKey)).Month.ToString & "'")
            whereClause.Append(")")

            whereClause.Append(" And ")
            whereClause.Append("(")
            whereClause.Append("PlantCode = '" & sourceValues.Item(StageConstants.MasterDimensionNames.Entity).ToString & "'")
            whereClause.Append(")")

            whereClause.Append(" And ")
            whereClause.Append("(")
            whereClause.Append("InvNo = '" & sourceValues.Item(StageConstants.MasterDimensionNames.Attribute1).ToString & "'")
            whereClause.Append(")")

            whereClause.Append(" And ")
            whereClause.Append("(")
            whereClause.Append("ProdModel = '" & sourceValues.Item(StageConstants.MasterDimensionNames.UD2).ToString & "'")
            whereClause.Append(")")
```

```
                    whereClause.Append(" And ")
                    whereClause.Append("(")
                    whereClause.Append("DestinationCode = '" & sourceValues.Item(StageConstants.MasterDimensionNames.UD3).ToString & "'")
                    whereClause.Append(")")

                    whereClause.Append(" And ")
                    whereClause.Append("(")
                    whereClause.Append("CustID = '" & sourceValues.Item(StageConstants.MasterDimensionNames.UD4).ToString & "'")
                    whereClause.Append(")")

                    orderByClause.Append("ORDER BY ")
                    orderByClause.Append("BomCode")

                    'Create the full SQL Statement
                    statement.Append(selectClause.ToString)
                    statement.Append(fromClause.ToString)
                    If args.ClientFilterRequest.length > 0 then
                        statement.Append(whereClause.ToString)
                        statement.Append(" And ")
                        statement.Append(args.ClientFilterRequest)
                    Else
                        statement.Append(whereClause.ToString)
                    End If
                    If args.ClientSortRequest.Length > 0 then
                        statement.Append(args.ClientSortRequest)
                    Else
                        statement.Append(orderByClause.ToString)
                    End if
                    'ErrorHandler.LogMessage(si, statement.ToString)
                    Return statement.ToString
                Else
                    Return String.Empty
                End If
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
    End Class
End Namespace
```

```
Namespace OneStream.BusinessRule.Connector.RevenueMgmtHouston
    Public Class MainClass
        Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals,
ByVal api As Transformer, ByVal args As ConnectorArgs) As Object
            Try
                'Get the query information
                Dim connectionString As String = GetConnectionString(si, globals, api)

                'Get the Field name list or load the data
                Select Case args.ActionType
                    Case Is = ConnectorActionTypes.GetFieldList
                        'Return Field Name List
                        Dim fieldListSQL As String = GetFieldListSQL(si, globals, api)
                        Return api.Parser.GetFieldNameListForSQLQuery(si, DbProviderType.OLEDB,
 connectionString, true, fieldListSQL, false)

                    Case Is = ConnectorActionTypes.GetData
                        'Process Data
                        Dim sourceDataSQL As String = GetSourceDataSQL(si, globals, api)
```

```
                        api.Parser.ProcessSQLQuery(si, DbProviderType.OLEDB, connectionString,
true,
                               sourceDataSQL, false, api.ProcessInfo)
                        Return Nothing

                    Case is = ConnectorActionTypes.GetDrillBackTypes
                        'Return the list of Drill Types (Options) to present to the end user
                        Return  Me.GetDrillBackTypeList(si, globals, api, args)

                    Case Is = ConnectorActionTypes.GetDrillBack
                        'Process the specific Drill-Back type
                        Return Me.GetDrillBack(si, globals, api, args,
args.DrillBackType.DisplayType,
                               connectionString)
                End Select

                Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        'Create a Connection string to the External Database
        Private Function GetConnectionString(ByVal si As SessionInfo, ByVal globals As BRGlobals,
               ByVal api As Transformer) As String
            Try

                'Connection String Method
                '-------------------------------------------------------------
'                Dim connection As New Text.StringBuilder'
'                connection.Append("Provider=SQLOLEDB.1;")
'                connection.Append("Data Source=LocalHost\MSSQLSERVER2008;")
'                connection.Append("Initial Catalog=SampleData;")
'                connection.Append("Integrated Security=SSPI")
'                Return connection.ToString

                'Named External Connection
                '-------------------------------------------------------------
                Return "Revenue Mgmt System"
```

```
                Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        'Create the field list SQL Statement
        Private Function GetFieldListSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals,
               ByVal api As Transformer) As String
            Try
                'Create the SQL Statement
                Dim sql As New Text.StringBuilder

                sql.Append("SELECT Top(1)")
                sql.Append("TransID, PlantCode, CustId, CustName, InvNo,
                InvYear, InvMonth, InvDesc, GLAccount, WorkDay, ProdModel, BomCode, UnitPrice,
Units,
                       Amount, DestinationCode ")
                sql.Append("FROM InvoiceDocumentDetail ")

                Return sql.ToString

                Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        'Create the data load SQL Statement
        Private Function GetSourceDataSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals,
               ByVal api As Transformer) As String
            Try
                'Create the SQL Statement
                Dim statement As New Text.StringBuilder
                Dim selectClause As New Text.StringBuilder
                Dim fromClause as New Text.StringBuilder
                Dim whereClause as New Text.StringBuilder
                Dim orderByClause as New Text.StringBuilder
```

```
                selectClause.Append("SELECT ")
                selectClause.Append("TransID, PlantCode, CustId, CustName,
 InvNo, InvYear, InvMonth, InvDesc, GLAccount, WorkDay, ProdModel, BomCode, UnitPrice,Units,
Amount,
                       DestinationCode ")

                fromClause.Append("FROM InvoiceDocumentDetail ")

                whereClause.Append("WHERE ")
                'Get the YEAR from the current XF Workflow Unit TimeKey
                whereClause.Append("(")
                whereClause.Append("InvYear = " & TimeDimHelper.GetYearFromId
(api.WorkflowUnitPk.TimeKey).ToString)
                whereClause.Append(")")

                'Get the MONTH from the current XF Workflow Unit TimeKey
                whereClause.Append(" And ")
                whereClause.Append("(")
                whereClause.Append("InvMonth = 'M" & TimeDimHelper.GetSubComponentsFromId
                       (api.WorkflowUnitPk.TimeKey)
                .Month.ToString & "'")
                whereClause.Append(") ")

                'Select Houston Plant Codes
                whereClause.Append(" And ")
                whereClause.Append("(")
                whereClause.Append("PlantCode IN('H200','H210')")
                whereClause.Append(") ")

                orderByClause.Append("ORDER BY ")
                orderByClause.Append("PlantCode, CustId, WorkDay, ProdModel, DestinationCode")

                'Create the full SQL Statement
                statement.Append(selectClause.ToString)
                statement.Append(fromClause.ToString)
                statement.Append(whereClause.ToString)
                statement.Append(orderByClause.ToString)
```

```
                Return statement.ToString

            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        'Create the drill back options list
        Private Function GetDrillBackTypeList(ByVal si As SessionInfo,
 ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As ConnectorArgs) As List(Of
DrillBackTypeInfo)
            Try
                'Create the SQL Statement
                Dim drillTypes As New List(Of DrillBackTypeInfo)

                drillTypes.Add(New DrillBackTypeInfo
(ConnectorDrillBackDisplayTypes.FileShareFile,
 New NameAndDesc("Invoice Document","Invoice Document")))
                drillTypes.Add(New DrillBackTypeInfo(ConnectorDrillBackDisplayTypes.DataGrid,
 New  NameAndDesc("Material Type Detail","Material Type Detail")))

                Return drillTypes

                Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        'Execute specific drill back type
        private Function GetDrillBack(ByVal si As SessionInfo, ByVal globals As BRGlobals,
 ByVal api As Transformer,
 ByVal args As ConnectorArgs, ByVal drillBackType As ConnectorDrillBackDisplayTypes,
ByVal connectionString as String) As DrillBackResultInfo
            Try
                Select case drillBackType
                    case is = ConnectorDrillBackDisplayTypes.FileShareFile
```

```
                        'Show FileShare File
                        Dim drillBackInfo as new DrillBackResultInfo
                        drillBackInfo.DisplayType = ConnectorDrillBackDisplayTypes.FileShareFile
                        drillBackInfo.DocumentPath = Me.GetDrillBackDocPath(si, globals, api,
args)
                        return drillBackInfo

                    case is = ConnectorDrillBackDisplayTypes.DataGrid
                        'Return Drill Back Detail
                        Dim drillBackSQL As String = GetDrillBackSQL(si, globals, api, args)
                        Dim drillBackInfo as new DrillBackResultInfo
                        drillBackInfo.DisplayType = ConnectorDrillBackDisplayTypes.DataGrid
                        drillBackInfo.DataTable = api.Parser.GetXFDataTableForSQLQuery(si,
                        DbProviderType.OLEDB, connectionString, true, drillBackSQL,
 false, args.PageSize, args.PageNumber)
                        return drillBackInfo

                        case else
                        return Nothing
                End Select

                Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

        'Create the drill back Document Path
        Private Function GetDrillBackDocPath(ByVal si As SessionInfo,
ByVal globals As BRGlobals,ByVal api As Transformer, ByVal args As ConnectorArgs) As String
            Try
                'Get the values for the source row that we are drilling back to
                Dim sourceValues as Dictionary(Of string, Object) =
api.Parser.GetFieldValuesForSourceDataRow(si, args.RowID)
                If (Not sourceValues Is Nothing) And (sourceValues.Count > 0) then
                    Return "Applications/GolfStream_v24/DataManagement/RevenueMgmtInvoices/"
```

```
 & sourceValues.Item(StageConstants.MasterDimensionNames.Attribute1).ToString & ".pdf"
                Else
                    Return String.Empty
                End If
                Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function

    'Create the drill back SQL Statement
        Private Function GetDrillBackSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals,
 ByVal api As Transformer, ByVal args As ConnectorArgs) As String
            Try
                'Get the values for the source row that we are drilling back to
                Dim sourceValues as Dictionary(Of string, Object) =
api.Parser.GetFieldValuesForSourceDataRow(si, args.RowID)
                If (Not sourceValues Is Nothing) And (sourceValues.Count > 0) then

                    Dim statement As New Text.StringBuilder
                    Dim selectClause As New Text.StringBuilder
                    Dim fromClause as New Text.StringBuilder
                    Dim whereClause as New Text.StringBuilder
                    Dim orderByClause as New Text.StringBuilder

                    'Create the SQL Statement
                    selectClause.Append("SELECT ")
                    selectClause.Append("CustName, InvDesc, BomCode, UnitPrice, Units,Amount ")

                    fromClause.Append("FROM InvoiceMaterialDetail ")

                    whereClause.Append("WHERE ")
                    'Get the YEAR from the source record
                    whereClause.Append("(")
                    whereClause.Append("InvYear = " &  TimeDimHelper.GetYearFromId
(sourceValues.Item
                    (StageTableFields.StageSourceData.DimWorkflowTimeKey).ToString))
                    whereClause.Append(")")
```

```
                    'Get the MONTH from the source record
                    whereClause.Append(" And ")
                    whereClause.Append("(")
                    whereClause.Append("InvMonth = 'M" & TimeDimHelper.GetSubComponentsFromId
(sourceValues.Item
                    (StageTableFields.StageSourceData.DimWorkflowTimeKey))
                    .Month.ToString & "'")
                    whereClause.Append(")")

                    whereClause.Append(" And ")
                    whereClause.Append("(")
                    whereClause.Append("PlantCode = '" & sourceValues.Item
                       (StageConstants.MasterDimensionNames.Entity)
                    .ToString & "'")
                    whereClause.Append(")")

                    whereClause.Append(" And ")
                    whereClause.Append("(")
                    whereClause.Append("InvNo = '" & sourceValues.Item
                       (StageConstants.MasterDimensionNames.Attribute1)
                    .ToString & "'")
                    whereClause.Append(")")

                    whereClause.Append(" And ")
                    whereClause.Append("(")
                    whereClause.Append("ProdModel = '" & sourceValues.Item
                       (StageConstants.MasterDimensionNames.UD2).ToString& "'")
                    whereClause.Append(")")

                    whereClause.Append(" And ")
                    whereClause.Append("(")
                    whereClause.Append("DestinationCode = '" & sourceValues.Item
                       (StageConstants.MasterDimensionNames.UD3).
                    ToString & "'")
                    whereClause.Append(")")

                    whereClause.Append(" And ")
```

```
                    whereClause.Append("(")
                    whereClause.Append("CustID = '" &sourceValues.Item
(StageConstants.MasterDimensionNames.UD4)
                    .ToString & "'")
                    whereClause.Append(")")

                    orderByClause.Append("ORDER BY ")
                    orderByClause.Append("BomCode")

                    'Create the full SQL Statement
                    statement.Append(selectClause.ToString)
                    statement.Append(fromClause.ToString)
                    If args.ClientFilterRequest.length > 0 then
                        statement.Append(whereClause.ToString)
                        statement.Append(" And ")
                        statement.Append(args.ClientFilterRequest)
                    Else
                        statement.Append(whereClause.ToString)
                    End If
                    If args.ClientSortRequest.Length > 0 then
                        statement.Append(args.ClientSortRequest)
                    Else
                        statement.Append(orderByClause.ToString)
                    End if
                    'ErrorHandler.LogMessage(si, statement.ToString)
                    Return statement.ToString
                Else
                    Return String.Empty
                End If
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
    End Class
End Namespace
```

Data Management Export Sequences

Instead of a file or connector, use a data management export sequence in a workflow import. This can be comprised of multiple data management steps. See "Data Management" in Application Tools for more information.

Data Management Settings

Data Export Sequence Name
The name of the data export sequence being used for this data source.

Source Dimensions

Each data source has assigned Source dimensions. To add Source dimensions to a data source, click **Add** in the toolbar and select a dimension. These dimensions correspond with the dimensions in the cube specified in data source properties.

Source Dimension Properties

You can specify the following properties for a selected Source dimension.

Settings

Data Type
Not all data types will be available for every Source dimension.

DataKey Text
This will read the value from the file as defined in the position settings.

Stored DataKey Text
This will override the position settings and force the Time value to be a constant value for every line.

Global DataKey Time
This will use the Time value from the current Global POV time being processed.

Current DataKey Time
This will use the Time Value from the current Workflow POV.

Current DataKey Scenario
This will use the Scenario value from the current Workflow POV.

Matrix DataKey Text
This is used as a Data Type when the data source is setup as a Matrix Load with multiple periods and used to identify the defined Start Position and Length of a single period.

Text
This will read the value from the file as defined in the position settings.

Stored Text
This will override the position settings and force the value to be a constant value for every line.

Matrix Text
This will have multiple amounts on a given line using rows and columns. This will determine the intersection that corresponds with each amount when more than one column contains the same dimension.

Label
This will read the value from the file as defined in the position settings.

Stored Label
This will override the position settings and force the value to be a constant for every line.

Numeric
This defines the numeric amount field for the data source. This field will be read and stored as a number, not as text or string.

Position Settings

Position settings are the definition of where the Source dimension will be found in the source file. For both fixed width and delimited files, there are tools in the toolbar and an attached file that assist in populating these values. Highlight the specific area to assign it to a dimension (for a delimited file, it only needs to be a portion of the column). The highlight will appear in red. When the defined area is selected, click . This will commit the selection to the dimension and the corresponding values will be populated in either the Start Position and Length fields for Fixed Width or the Column Number field for Delimited. To clear this selection without committing it to the dimension, click .

Start Position (Fixed Files Only)
This is the numerical representation of the starting point for a line item.

Length (Fixed Files Only)
This defines how many characters will be taken from the start position.

A Fixed data source with a start position of 20 and a length of five will start with the 20th character and include the next five characters.

Column (Delimited Files Only)
A delimited data source will use a column number. If the column number is four, that dimension will be represented by the 4th column in the source file.

Connector Settings

Source field names will be provided by the Connector business rule assigned to the connector data source. These field names will either be explicitly listed out in the business rule, or dynamically returned from a SQL query. Source field names for Data Management Sequences are provided and always contain the same list.

Logical Expression and Override Settings

Logical Operator
Allows the ability to assign a .NET scripting functionality to a dimension of a data source.

Complex Expression
This selection is used when .NET scripting is needed for the dimension, but not needed elsewhere. The script used in the complex expression will only be available within that dimension.

Business Rule
This selection is used when .NET scripting is needed for the dimension and the script is available in the Business Rule Library.

Logical Expression
This is the name of the business rule assigned to the dimension when Business Rule is selected for Logical Operator.

Static Value
This is an override setting which allows a hard-coded value to be assigned to a dimension rather than being read from a file or data source.

Text Fill Settings

Leading Fill Value
Characters entered in this field will precede whatever value is brought in from the file upon import. Lead Fill Mask = xxx, data value = 00, results value = x00

Trailing Fill Value

Characters entered in this field will be placed after any value brought in from the file upon import.

Trail Fill Mask = xxx, Data Value = 00, Results Value = 00x

Substitution Settings

Substitution Old Value (Find)

If the value entered in this field is encountered in the dimension, it will be replaced with what is entered in the Substitution New Value.

Single Value = value1

Multiple Values = value1^value2

Substitution New Value (Replace)

This will replace the value in Substitution Old Value if it occurs in the dimension.

Single Value = value1

Multiple Values = value1^value2

Empty String | Null|

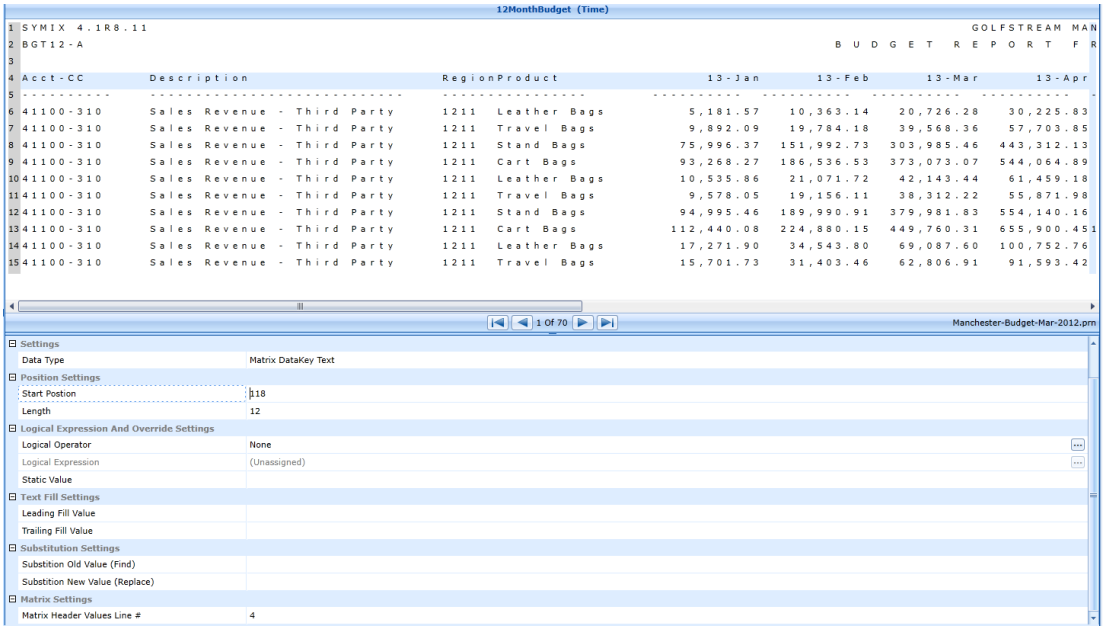
Single Space |Space|

Matrix Settings

This setting is only available when Matrix is set as the Data Structure Type.

Matrix Header Values Line #

This setting indicates which row to look to for the dimension being matrixed. For example, if months are listed across columns on line four, and time is the intended Member to be matrixed, a 4 would be entered in this field.



Numeric Settings

These settings are only available in the Amount Source dimension which will help with the formatting and properties of the amount values.

Thousand Indicator

Enter the character used to separate thousands in the value. For example, for the value 1,000 the Thousand Indicator is “,”. This can also be done by highlighting the character in the file and clicking

.

Decimal Indicator

Enter the character used to separate decimals in the amount value. This can also be done by highlighting the character in the file and clicking

.

Currency Indicator

Enter the currency symbol for the respective currency. This can also be done by highlighting the character in the file and clicking .

Positive Sign Indicator

If the amount values in the file contain text characters to dictate a positive value, enter the characters here. This can also be done by highlighting the character(s) in the file and clicking .

Negative Sign Indicator

If the amount values in the file contain text characters to dictate a negative value, enter the characters here. This can also be done by highlighting the character(s) in the file and clicking .

Debit / Credit Mid-Point Position

If debits and credits are in the same amount field, but are offset within that column, a midpoint can be entered here. Values to the left of the midpoint are considered a debit while values to the right are a credit.

Factor Value

The amount being imported is factored by the value entered in this field.

Rounding

The available options for Rounding are Not Rounded and the values 1 – 10. Not Rounded will not round the values. If a value between 1 and 10 is selected, the value will be rounded to the corresponding digit.

Zero Suppression

If the import process should not include zero values, set this to True. To import 0 values, set this to False.

Text Criteria for Valid Numbers
Fill in the criteria for numbers that are valid.

Bypass Settings

Bypass allows an administrator to look for a specific value in a column or an entire line. If a value is found, that line will not be processed. In order to setup the Bypass dimension, highlight the value to skip. Click to skip the value only if it is found in the exact position. Click to skip the value if it appears anywhere on a line.

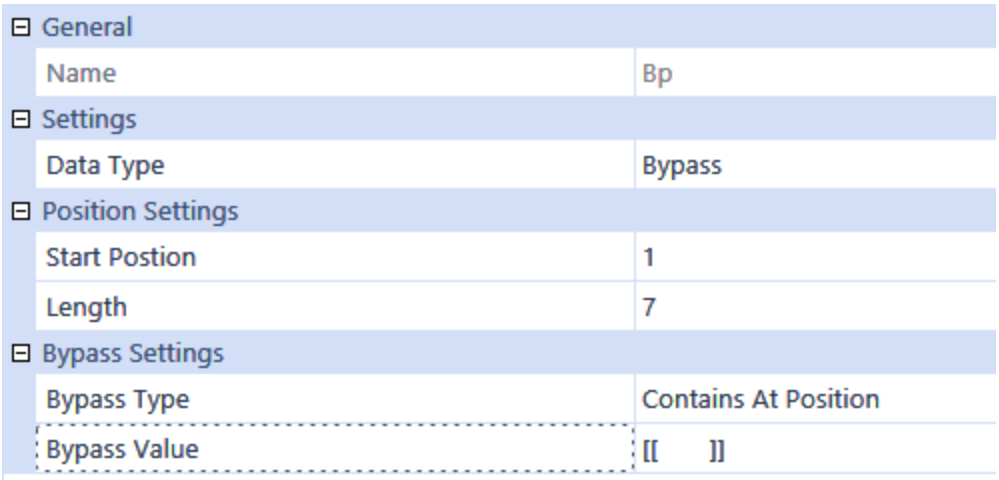
Bypass Type

Contains at Position
This switch will tell the data source to skip an entire line of a file if the Bypass Value is found at the specified location in the Position Settings section.

Contains Within Line
This switch will tell the data source to skip an entire line of a file if the Bypass Value is found anywhere on the line.

Bypass Value
The value defined will indicate an entire line should be skipped when found in a specific location, or anywhere on the line.

> **TIP:** Create a bypass in a fixed data source for blank spaces by specifying the position settings and entering double square brackets around the number of blank spaces in the Bypass Value. This can be used if an import will encounter an area in the data source containing blank spaces in the location specified.



Stored Text Settings

Text Criteria to Bypass in Storage Buffer

This field provides the opportunity to enter a value or string of values indicating a bypass of the record being read even if you specify a stored value.

Single Value = value1

Multiple Values = value1^value2

Stored Value Line #

The line number to be used repeatedly to obtain value for each record's importing regardless of whether it is on the line.

Using Excel as a Data Source

An Excel file can be imported without having to configure the data source completely by setting Allow Dynamic Excel Loads to True and configuring an Excel template. See "Loading Data via Excel Templates or CSV" in Data Collection Guides for more information.

Forms

Forms Channel Workflow

To minimize form maintenance, Cube View and Excel XFSetCell updates are not tied to specific Forms. Association is at the Input Type level, not the individual Form level. The Forms Input Type determines if you can update data from Excel. If the Forms channel is completed, but the process is not certified, you can import data from Excel using XFSetCell or a Cube View. If a cell is updated, the Analytic Engine traces the cell by:

1. Identifying the Workflow that owns the Entity.
2. Checking the Input Child Workflow Profile.

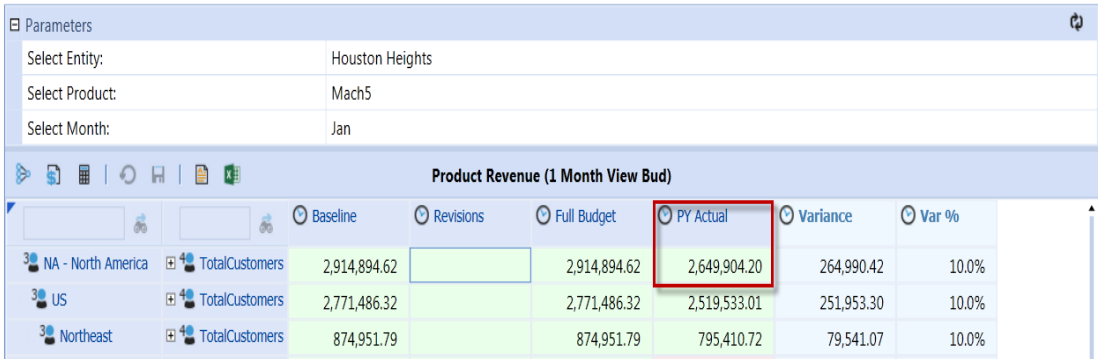
This determines if the Form Input Type is enabled for the Scenario Type. If not, the Form Input Type is disabled and cannot update cells from a Cube View in the web, Cube View, or XFSetCell function from Excel.

If Form Input Type is enabled, the Analytic Engine checks the full Workflow Status for the active Form Input Type. If the Workflow is locked or the Parent Workflow is certified, cells are not updated. If the Workflow indicates updates can occur, the Process Cube task of the Workflow and all ancestor Parent Workflows are impacted.

Form Allocations

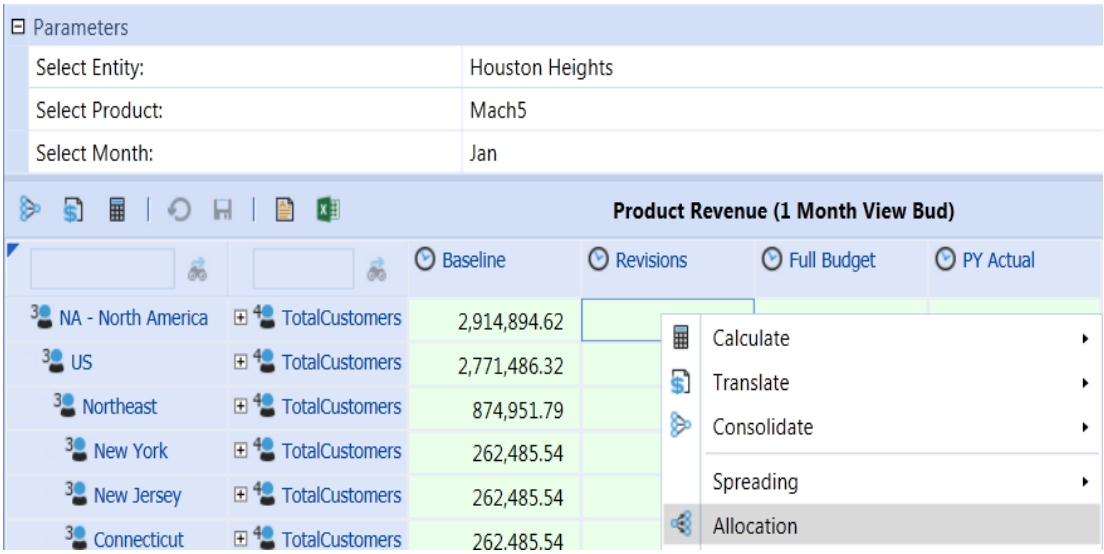
Advanced Distribution

In the example below, an advanced distribution is used on a Product Revenue Form. This allocation will take the previous year's actual data, increase it by 20% and populate the current year's revenue budget revisions for all regions and customers.

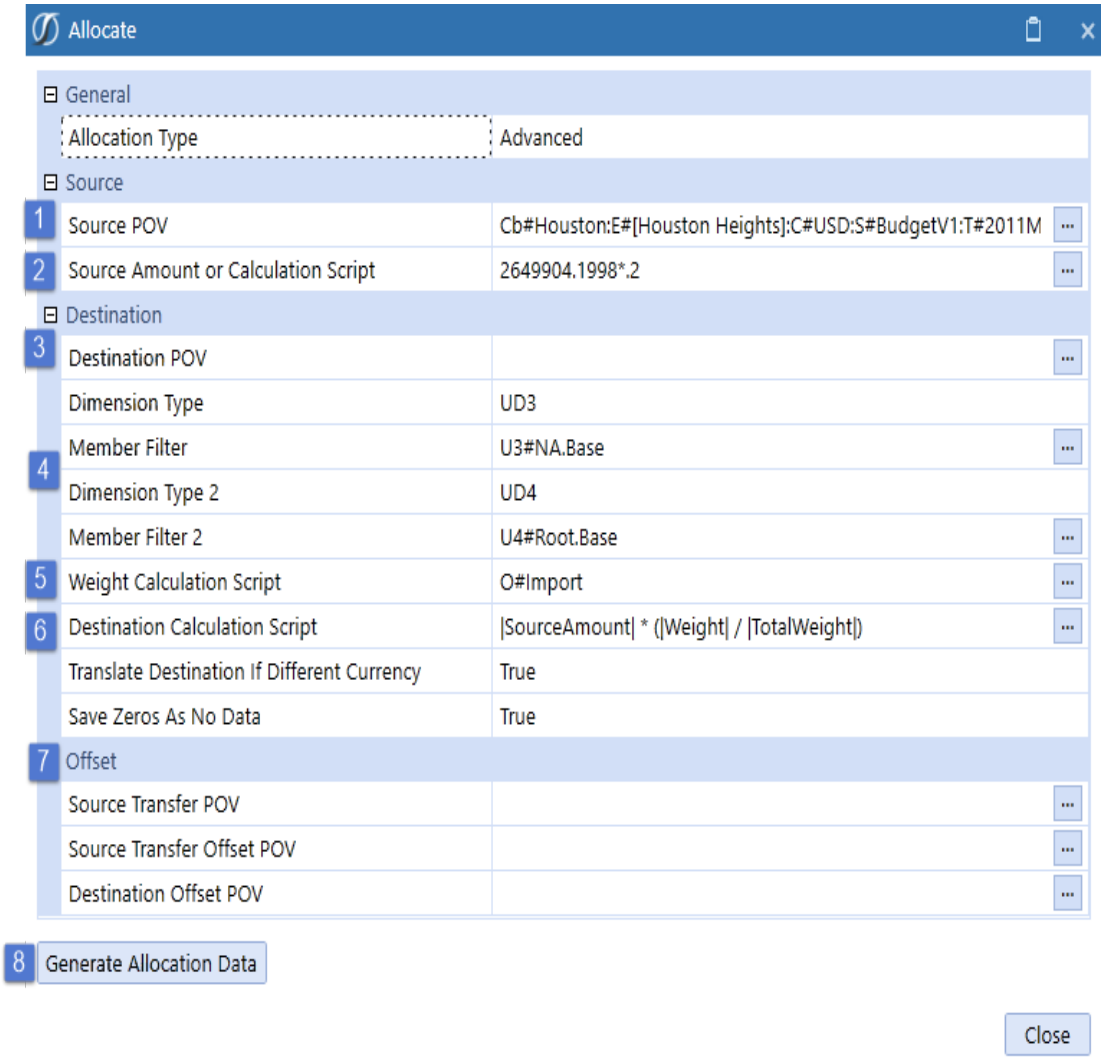


The allocation data is being written to a form which will then populate the Revisions column. The sum of the Baseline and Revisions will then create the new Full Budget for each Region and Customer.

Right-click the first data cell in the Revisions column and select Allocation. This helps create the Source and Destination POV.



By default, the Allocation dialog will open to the last Allocation processed. Select the Allocation Type desired (e.g., Advanced).



1. **Source POV**

   The Source POV defaults to the last cell selected for Allocation. Every Dimension is represented in the POV. In this example, it defaults to the data cell under Revisions because that is where the allocation option was selected.

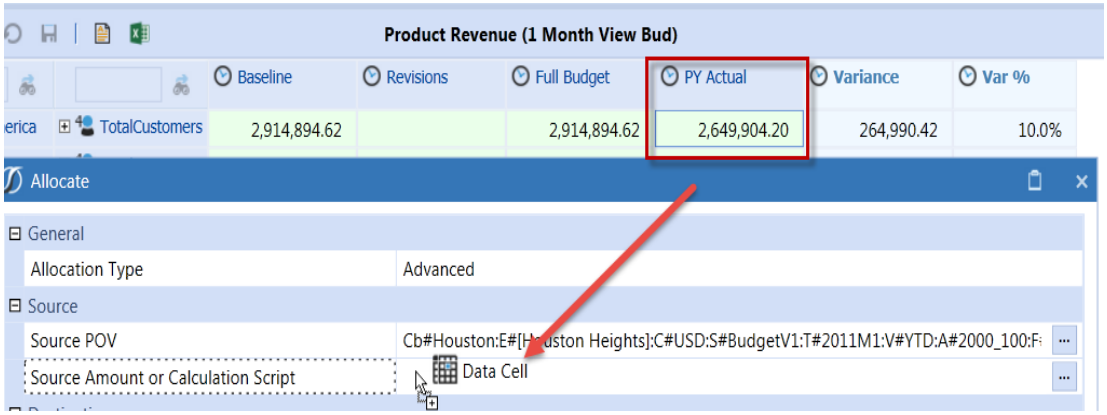
Cb#Houston:

```
E#[HoustonHeights]:C#USD:S#BudgetV1:T#2011M1:V#YTD:A#2000_
100:F#None:O#Forms:I#None:U1#None:U2#Mach5:U3#NA:U4#TotalCustomers:U5#None:U6#None:U7#
None:U8#None
```

Users can also select a data cell from the grid and drag and drop the cell's POV into this field. The Source POV is the default Source Amount for the allocation.

2. **Source Amount or Calculation Script**

   To override the Source POV value, enter a source amount or a calculation script. In this example, the PY Actual value is used for the Source Amount. Click on the cell in the grid and drag and drop the value into the Source Amount property. To increase this amount by 20%, the value is multiplied by .2.



3. **Destination POV**

   This is where the allocation is applied. In this example, the Destination POV is blank because it is using the same Members from the Source POV. Users can also drag and drop a data cell's POV.

4. **Dimension Type/Member Filters**
   These properties override the Destination POV and allow allocations to occur to several Members at a time. In this example, the UD3 (Regions) and UD4 (Customers) Dimensions are specified and therefore will override the UD3 and UD4 Members in the Destination POV.

5. **Weight Calculation Script**
   The Weight Calculation Script determines how the allocation is weighted. Any Members not specified in the script derive from the Destination POV. In this example, the weight for each Member is determined by the imported value in the Baseline column.

6. **Destination Calculation Script**
   The default Destination Calculation Script is |SourceAmount|* (|Weight|/|TotalWeight|). Additional calculations may be added to this field to customize how the weight calculation is performed. This example uses the default calculation.

7. **Offset**
   The offset properties are optional and not used in this example.

8. **Generate Allocation Data**
   After the allocation dialog is complete, click this button to see the allocation data before applying it.

The allocation results dialog provides information on all the allocation destinations, weight information, and displays all the data rows that will be updated upon selecting Save Allocation Data. Check the Show All Dimensions box to see every Dimension intersection for each data row. After the allocation data is saved, the form data will update and store the data to the Cube.

Allocation Results

Source POV

Cb#Houston:E#[Houston Heights]:P#?:C#USD:S#BudgetV1:T#2011M1:V#YTD:A#2000_100:F#None:O#Forms:I#None:U1#None:U2#Mach5:U3#NA:U4#TotalCustomers:U5#None:U6#None:U7#None:U8#None

Source POV Amount 0.00

|SourceAmount| 529,980.84

|NumDestinations| 483

|TotalWeight| 2,914,894.62

Show All Dimensions

Amount	Is NoData	|Weight|	Type	UD3	UD4
4,980.47		27,392.60	Destination	Michigan	Active Hub
6,445.32		35,449.24	Destination	Michigan	Double Eagle
161.13		886.23	Destination	Michigan	East Sports
336.91		1,853.03	Destination	Michigan	EZ Sporting Goods
146.48		805.66	Destination	Michigan	Golf Hub
981.45		5,397.95	Destination	Michigan	Golf Pro

Save Allocation Data Cancel

Results:

Product Revenue (1 Month View Bud)

		Baseline	Revisions	Full Budget	PY Actual	Variance	Var %
Midwest	TotalCustomers	397,676.06	72,304.74	469,980.80	361,523.69	108,457.11	30.0%
Michigan	TotalCustomers	99,419.01	18,076.18	117,495.20		117,495.20	
	Active Hub	27,392.60	4,980.47	32,373.07		32,373.07	
	Double Eagle	35,449.24	6,445.32	41,894.56		41,894.56	
	East Sports	886.23	161.13	1,047.36		1,047.36	
	EZ Sporting Goods	1,853.03	336.91	2,189.94		2,189.94	
	Golf Hub	805.66	146.48	952.15		952.15	
	Golf Pro	5,397.95	981.45	6,379.40		6,379.40	

> **NOTE:** The Var % column updated itself to 30% from 10% because of the additional 20% added to the allocation. The Full Budget column also updated itself with the new total from the Baseline and Revisions columns.

Form Templates

Form templates can be setup to allow manual data entry. The entries can be done in a cube view from an Excel file or from the Spreadsheet feature (OneStream Windows App only). Each form template group has an assigned cube view, dashboard or Excel file. Forms can also be loaded using an Excel or CSV template. See "Loading Form Data" in Data Collection Guides for more information.

Applying Literal Value Parameters to Form Templates

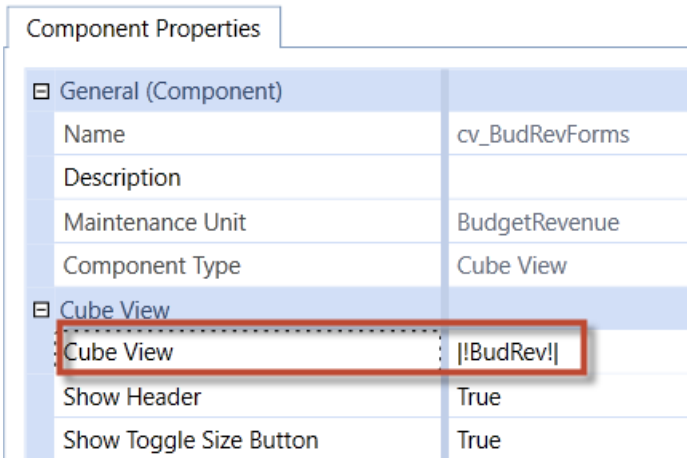
A Delimited List Parameter containing several Cube View or Spreadsheet names can be applied to a Dashboard to use the Dashboard in multiple Form Templates via the Name Value Pairs Form Template Property. Instead of creating multiple Dashboards to assign to multiple Form Templates, users can define the Parameter name thus defining which Cube View or spreadsheet the specific form should use. This approach helps in achieving a common toolbar and look for all data entry forms.

The example below uses Cube Views, however, if the forms are driven from Spreadsheets, a Spreadsheet Dashboard Component can also be used.

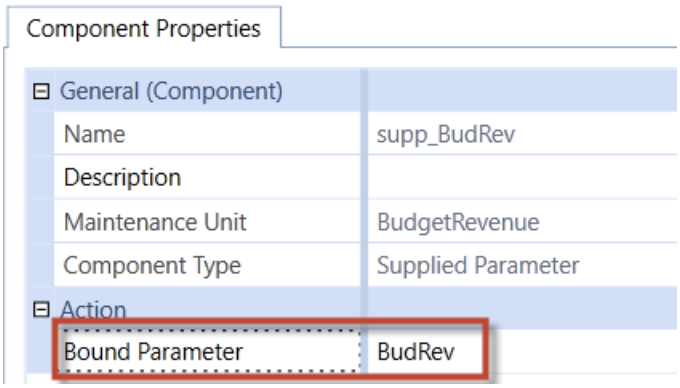
1. Design the Cube Views necessary for data entry. After the Cube Views are complete, create a Dashboard Maintenance Unit.
2. Within the Dashboard Maintenance Unit, create a Delimited List Dashboard Parameter specifying all Cube View names in both the Value Items Property.

General (Parameter)	
Name	BudRev
Description	Budget Revenue Input Forms NA Course Mgmt
User Prompt	
Maintenance Unit	BudgetRevenue
Sort Order	0
Data Source	
Parameter Type	Delimited List
Default Value	
Display Items (comma delimited)	Design NA Revenue (1 Month View),Event Coordination NA Revenue (1 Month View),Services NA Revenue (1 Month View)
Value Items (comma delimited)	Design NA Revenue (1 Month View),Event Coordination NA Revenue (1 Month View),Services NA Revenue (1 Month View)

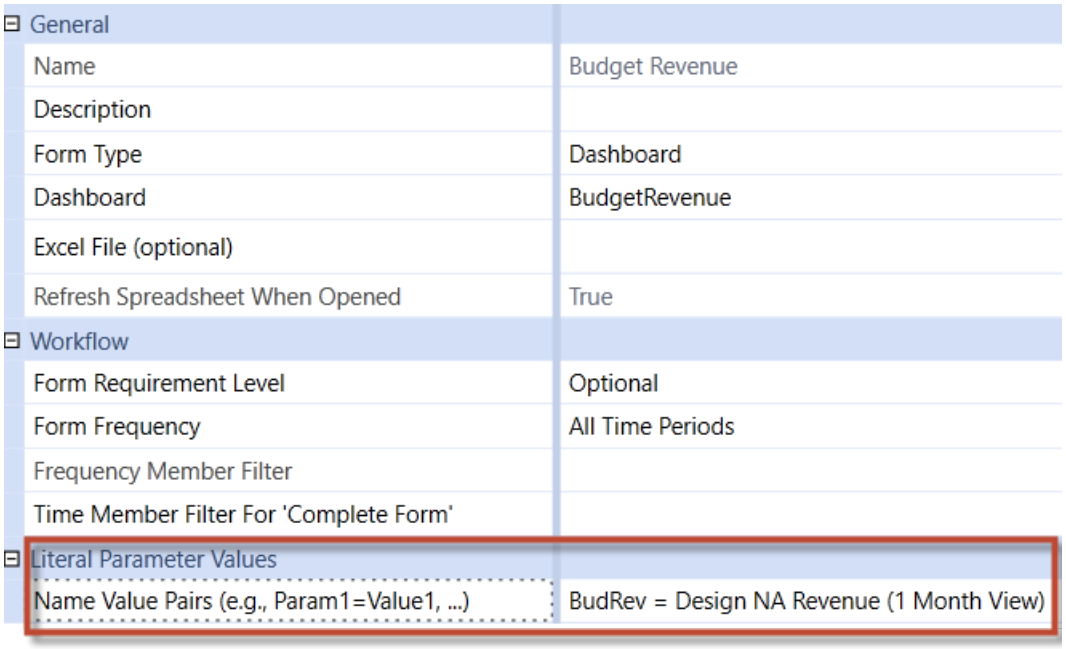
3. Create a Cube View Dashboard Component and enter the Parameter name in Cube View Property enclosed in Pipes and Exclamation Marks.



4. Create a Supplied Parameter Dashboard Component to pass the Parameter value from the Dashboard to the Form Template. Specify the Parameter Name in the Bound Parameter property.



5. Create a Dashboard with a Uniform Layout Type and assign the Cube View and Supplied Parameter Components to it.
6. Create a Form Template and set the Form Type to Dashboard and assign the desired Dashboard.
7. Define which Cube View this specific Form should use in the Name Value Pairs property by hardcoding a specific Cube View name from the Delimited List Parameter.



When the Form Template is used in the Workflow, the specified Cube View will display for data entry.

Form Template Profiles

After the form template groups are created, they are organized into form template profiles. The profiles are then assigned to workflow profiles.

- Click to create a new profile.
- Click to assign a group to a profile. This allows the user to select which groups will be in the profile.
- Assign a profile to a workflow profile by clicking Application > Workflow Profiles | Form Settings, or Journal Settings for journal template profiles.

> **NOTE:** Click to access the Security screen to modify users and groups before assigning them to a form template profile. To assign groups, click and begin typing the group name. Groups are filtered, so you can select the group, press CTRL and then double-click. This specifies the correct name into the appropriate field.

Security

Access Group
Members of this group have access to the Form Template group.

Maintenance Group
Members of this group have the authority to maintain the Form Template group.

> **NOTE:** Click to navigate to the Security page so you can modify users or groups before assigning them. Click and begin typing the name of the Security group in the blank field. As the first few letters are typed, groups are filtered, making it easier to find the right group. Select the group, click CTRL and double-click.

Form Template Properties

Form details can also be loaded using an Excel or CSV template. See Loading Form Data in Data Collection Guides for more information.

General

Name
Name of the form template.

Description
This allows for a more descriptive definition of the form template.

Form Type

Cube View
Select this to utilize a cube view for the form's data entry method.

Dashboard
Select this to utilize a dashboard for the form's data entry method.

Spreadsheet (OneStream Windows App Only)
Select this for the Form to be visible using the Spreadsheet feature in OneStream Windows App. When selected, it is the same functionality as attaching an Excel file, but the spreadsheet only exists within OneStream.

Cube View/Dashboard

Select the cube view or dashboard that will be associated with this form template. Click and begin typing the name of the cube view or dashboard in the blank field. Names are filtered making it easier to select items. If the name is unknown, click Cube View or Dashboard Group and scroll through the list. Select a cube view or dashboard, click CTRL and then double-click.

Spreadsheet (OneStream Windows App Only)

This spreadsheet should have been built with an embedded cube view or retrieve functions in order to interact with form data. See Navigating the Excel Add-In for more information on this functionality that is available within the Excel Add-in and the Spreadsheet feature in OneStream Windows App. Upload a pre-created Excel file for data entry when this option is selected. Click to upload an Excel file. Once selected, the file name in the Excel File space will appear.

- Click to delete an uploaded Excel file.
- Click to download a copy of the uploaded Excel file.
- Click to open the uploaded Excel file.

Excel File (Optional)

An Excel file may be used for data entry. Click to upload an Excel file. Once selected, the file name in the Excel File space will appear. You can also:

- Click to delete an uploaded Excel file.
- Click to download a copy of the uploaded Excel file.
- Click to open the uploaded Excel file.

Workflow

Form Requirement Level

Not Used
This setting is used if the form is no longer in use and shows the form as Deprecated in the Workflow.

Optional
This setting will allow the user to enter data via the form if desired.

Required
This setting will make the form mandatory for any process to which it is assigned

Form Frequency

All Time Periods
This allows the form to display every period

Monthly
This allows the form to display every month. If this is for a weekly application, they will display the last week of each month.

Quarterly
This allows the form to display every quarter, or four times a year.

Half Yearly
This allows the form to display two times a year; once in June and December.

Yearly

This allows the form to display once a year in December.

Member Filter

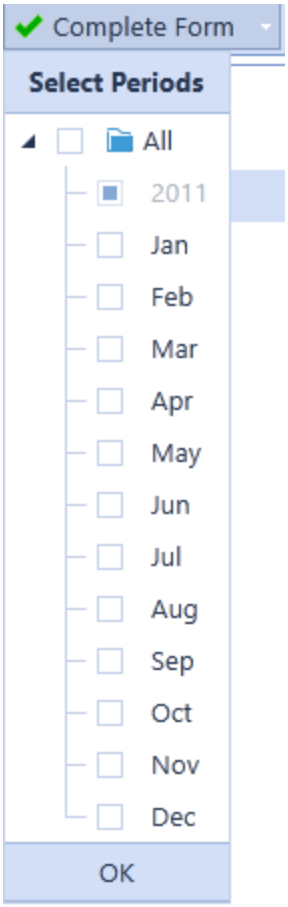
This turns on the Frequency Member Filter. Filters can then be defined in that section.

Frequency Member Filter

This only becomes available when Member Filter is chosen in the Forms Frequency options above, otherwise this is gray for all the others. The purpose of this option is to allow the ability to filter by time.

Time Filter for Complete Form

The Time filter will dictate how often the form is required. The example below is what will be shown when selecting Complete Form using the filter T#WFYear.Base.



Literal Parameter Values

If the dashboard selected for the form, contains a parameter for the cube view or Spreadsheet name, enter the name/value pair which specifies which cube view the form will use in the Workflow.
Example: ParameterName = CubeViewName

See "Applying Literal Value Parameters to Form Templates" in Data Collection Guides for more information about the Literal Parameter Value feature.

Parameters (Cube View Forms Only)

Parameter Type (for Parameters 1 through 6)
Parameter types allow for different values or variables to be passed to the cube view.

Literal Value
The value you specify will be hard coded.

Input Value
This allows the user to enter or change the value.

Delimited List
This provides a distinct list of values populated in the Parameter Type.

Member List
This will produce a flat list of Members to show to the user.

Member Dialog
Similar to the Member list, this allows the user to select a member, but through a pop-up Member Selection dialog which also has search capabilities. This is more appropriate for a dimension such as Accounts or Entities where the user can choose a base or Parent member by traversing a hierarchy.

Variable Name from Cube View
Place a parameter in this field to replace another parameter used in a cube view. For example, a prompt such as |!MyEntityName!| will suppress the Parameter dialog from appearing when opening the form.

User Prompt
This prompts the user based on the question entered here.

Default Type
Set the default value so it is not blank.

Dimension Type
Choose the dimension being prompted

Member Filter
Allows the ability to put in a defined filter. (for example, E#Root.WFProfileEntities)

Form Template Group Properties

This section includes general group properties.

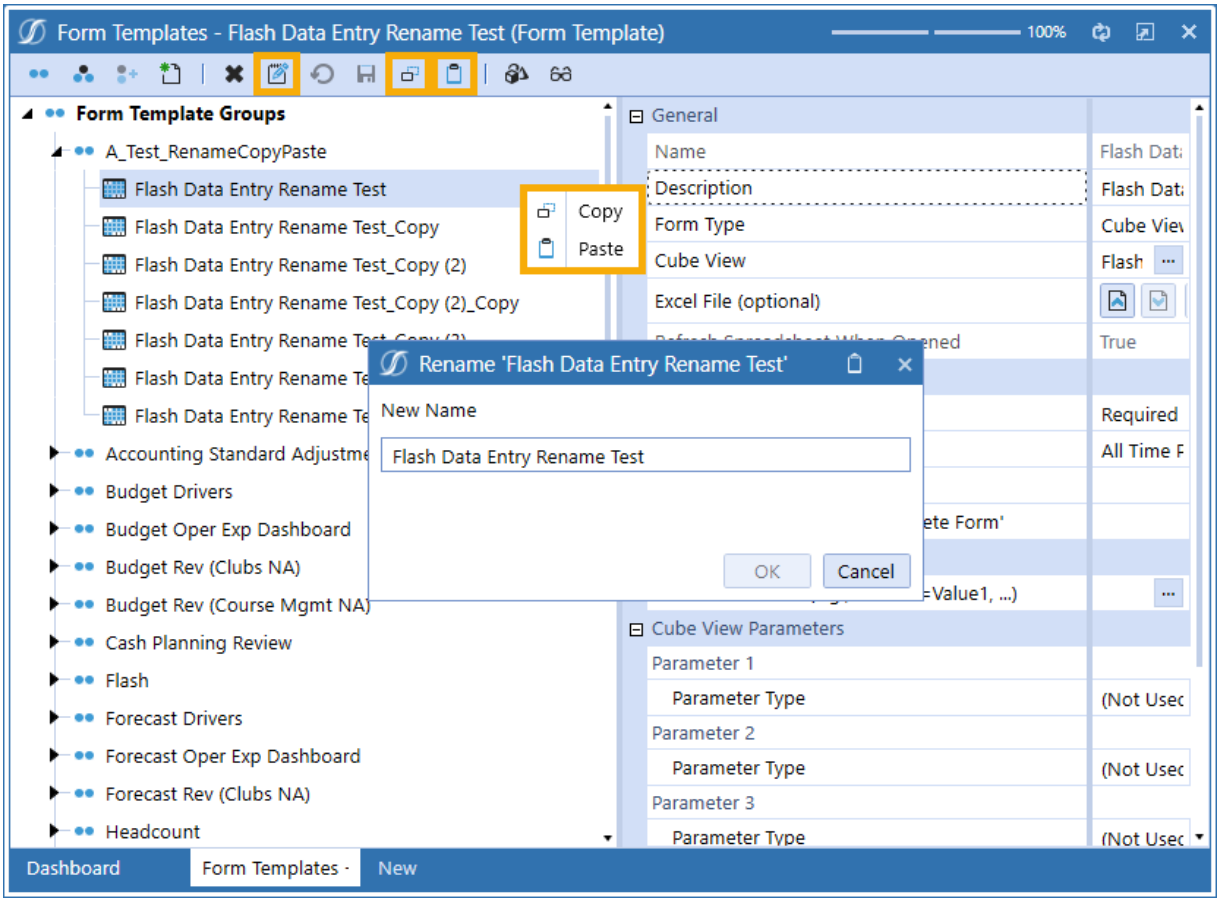
General (Form Template Group)

Name
The name of the form template group.

Description
A short description of the template group such as how or where it is used.

Managing Form Templates

Form Templates can be renamed, copied, and pasted from the toolbar. Form templates can be copy and pasted using a right-click.



Rename a form template by clicking the Rename icon from the toolbar. All associated links for the Workflow Profile and Dashboard Groups will remain valid as the form template is based on a unique identifier, not the name.

Copy a form template by clicking the Copy icon from the toolbar or right-clicking the form template name and selecting Copy.

Paste a form template by clicking the Paste icon from the toolbar or right-clicking the form template name and selecting Paste.

Transformation Rules

Transformation rules help map data from source systems to the financial model. The different member scripts and how you can use them with transformation rule types are described below. These are listed in the order in which the rules run during the Validate step in a workflow.

Transformation Rules Toolbar

Create Group

Use this to create a Transformation Rule Group under each dimension.

Create Profile

See Transformation Rule Profiles.

Manage Profile Members

See Transformation Rule Profiles.

Create and Populate Rule Profile

Selecting this will create a Transformation Rule Group and a Transformation Rule Profile with the same name. The Rule Profile will already be populated with each Dimension Rule Group and update as the Group is updated.

Delete Rule Profile and Groups

This can only be done by selecting a specific Transformation Rule Profile. It will delete the Transformation Rule Profile and all non-shared Groups within the Profile.

Delete Selected Item

Use this to delete one item such as one Rule Group or Profile.

Rename Selected Item

This allows a user to rename Transformation Rules, Groups or Profiles.

Cancel All Changes Since Last Save

This will undo any unsaved changes made.

Export Selected Group to a TRX File

Use this to export a Transformation Rule Groups to use in another application or group. This exports a comma delimited file which are valid for only certain Transformation Rule designs. Logical Operators and complex expressions are not supported. Application Tools, Load/Extract of application *.XML files should be used to manage Transformation Rules.

Import a TRX file into the Selected Group

Use this to import Transformation Rule Groups from another application or group. This comma delimited file format may not be valid for all Transformation Rule designs. Properties such as Logical Operators and complex expressions are not supported. The *.TRX import is intended as a utility to import Transformation Rules from compatible legacy systems. Application Tools, Load/Extract of application *.XML files should be used to manage Transformation Rules.

General Settings

General

Name

The name of the Transformation Rule group.

Description
A short description of the rule or where it is used.

Security

Access
Members of this group will have access to the Transformation Rule group.

Maintenance
Members of this group have the authority to maintain the Transformation Rule group as well as load/extract from the Transformation Rules page.

> **NOTE:** Click 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. After the Group is selected, press CTRL and double-click to add the correct name into the appropriate field.

Settings

Cube Dimension Name
The specific dimension to which the rule group is assigned.

Mapping Types

Mapping Types allow the data to be mapped in different ways with the possibility of using conditional rules, wild cards, ranges and others.

> **IMPORTANT:** The characters **!**, **?**, and **|** are used in operations for Transformation Rules. These characters will cause errors if used with Source Members or Target Members.

One-to-One

One-to-One mapping allows one Source dimension member to be mapped or transformed to one target dimension member explicitly. No member scripts are used.

Source Value
This is the value for the related Cube dimension in a defined data field.

Description (Optional)
This is a description of the mapping rule.

Target Value
The Dimension Library member to which the Source Value is being mapped or transformed.

Order (Optional)
The one-to-one mapping rules will be processed in the order of the (defaulted) alpha numerical sort. The Order field allows a value to be assigned to a record which will allow a custom sort order of the mapping table.

Composite

Composite mapping is a way to do a map conditionally. Dimensional tags may be used to include another dimension in the evaluation.

The * character is used to represent any number of characters. The ? character acts as a placeholder for a single character. In the following example, any similarly formatted account starting with 199 and an entity of Texas would be mapped to a specific target.

> **Example:** A#[199?-???*]:E#[Texas]

In the following example of an entity composite rule using the ? character, records for all entities with accounts that start with H that have 10 or fewer characters would be captured, for example, accounts named Head, Heads, and Headcount.

> **Example:** E#[*]:A#[H?????????]

Composite rules stop processing when a record meets the criteria. So, if the record meets the criteria for more than one rule, set the order to ensure they are mapped correctly.

For example, using the following records, we can map those with Head to the entity Rhode Island, Heads to the entity Maine, and Headcount to be bypassed by setting the order from the shortest account name (narrowest search) to the longest account name (broadest search).

Amount (Raw)	Sourc	Eastern	Time	View	SubSidAccour
10.00	Files	Connecticut	(Current)	YTD	Head
10.00	Files	Mass	(Current)	YTD	Head
10.00	Files	VT	(Current)	YTD	Head
1,000.00	Files	Connecticut	(Current)	YTD	Headcount
1,000.00	Files	Mass	(Current)	YTD	Headcount
1,000.00	Files	VT	(Current)	YTD	Headcount
100.00	Files	VT	(Current)	YTD	Heads
100.00	Files	Mass	(Current)	YTD	Heads
100.00	Files	Connecticut	(Current)	YTD	Heads

So, we would set the mapping to run in the following order:

1. E#[*]:A#[H???] to map all accounts that start with H and have 4 or fewer characters
2. E#[*]:A#[H????] to map all accounts that start with H and have 5 or fewer characters
3. E#[*]:A#[H?????????] to map all accounts that start with H and have 10 or fewer characters

Type Composite

Rule Name	Description	Rule Expression	Target Val	Logical Operator	Order
AllwithHeadcounttoBypass		E#[*]:A#[H????????]	(Bypass)	None	30
AllwithHeadstoMaine		E#[*]:A#[H????]	Maine	None	20
AllWithHeadToRI		E#[*]:A#[H???]	RI	None	10

So first, the records for the Head account would map to Rhode Island. Then, the records for the Heads account would map to Maine. And last, the records for the Headcount account would be bypassed.

Amount	Eastern	Time	View	SubSidAccour
3,000.00	(Bypass)	2023M12	YTD	Headcount
300.00	Maine	2023M12	YTD	Heads
30.00	RI	2023M12	YTD	Head

Range

A range mapping gives the upper limit and lower limit of a range. Any member that falls within this range will be mapped to the corresponding target member.

Range rules do not stop processing when a record meets the criteria. If the ranges overlap and a member falls within multiple ranges, that member will be mapped only to the final range that is processed. So, for a member in multiple ranges, set the order so that the last rule run is applied.

If an administrator wants the range of source Accounts from 11202 to 11209 to map to Account 12000, enter 11202~11209 under Rule Expression with ~ as the separator. Range rules use character sets not integers. So, use balanced character sets to ensure the full range is captured. For example, a range of 4 to 3000 must be entered as 0004~3000 under Rule Expression.

List

List mapping allows the user to create a delimited list of members that all map to the same target.

If an administrator wants the list of Accounts 41137, 41139 and 41145 to map to Account 61000, enter 41137;41139;41145 under Rule Expression with ; as the separator.

Mask

Wild cards are used in mask mappings. The wildcard characters for these mappings are * and ?. The * character is used to represent any number of characters. 27* would capture 270, 2709, or 27XX-009. The ? character acts as a placeholder for a single character. 27?? Would capture 2709, but would not capture 27999 or 2700-101.

Double-sided wild cards can be used as well in Mask transformation rules. For example, *000* would capture any account number with character(s) before and after the 000 sequence.

The following properties are standard for Composite, Range, List and Mask Mapping Types.

Rule Name
This is a unique name assigned to each mapping rule that will also determine the default sort and processing order.

Description (Optional)
The description of the mapping rule.

Rule Expression
This is where the specific source field processing rule is placed. For example, 27* would capture 270, 2709, or 27XX-009.

Target Value
The Dimension Library member to which the Source Value is being mapped or transformed. The use of wild cards to define Target Values is not recommended. The following exceptions apply to Target Value wild card usage:

- The ? character is not supported.
- The * character is not supported when used as a prefix (left side) to a target value.
- The * character used as a suffix (right side) will yield the Source value.

Logical Operator
This provides the ability to extend a normal mapping rule with VB.NET scripting functionality.

Expression Type:

None (Default)
No script is assigned or employed for this related transformation rule.

Business Rule
This selection is used when .NET scripting is needed for the dimension and the script is available within the Business Rule Library.

Complex Expression
This selection is used when .NET scripting is needed for the dimension but will not need to be used elsewhere. The script used in the complex expression will only be available within that dimension.

Order (Optional)
The Order field allows a value to be assigned to a record which will allow a custom sort order of the mapping table.

Derivative

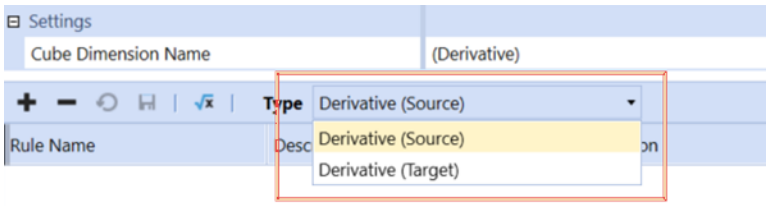
Derivative rules apply logic to stage data and generate additional records in the Stage environment. Derivative rule types include Source and Target.

Source Derivative Rules

Source derivative rules are created to apply logic defined by the inbound source data in the Stage environment. This will create new members as a data record, which are stored in the stage area, to be mapped to a cube or a temporary member.

Target Derivative Rules

Target derivative rules apply logic defined by the post-transformed stage environment data. This will create new members as a data record, which are stored in the stage area. Since these are post-transformation, any records stored as final will not be processed by transformation rules.



Rule Name
A unique name given to a particular derivative rule.

Description
A detailed description used in the Label field of the data imported.

BI Blend Derivative Rules

This is a class of derivative rules specifically designed for use with the external database tables generated from the BI Blend Workflow.

- BlendUnit All
- BlendUnit Base
- BlendUnit Parent

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

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

Derivative Rule Expressions

Below are some examples of the syntax used to write derivative rules. The variation between the use of Source or Target derivative rules is by the definition of the rule. Source derivative rules reference the inbound record members. Target type derivative rules reference the post-transformed records. In the rule expression samples, assume these rules will run in this order presented in the example.

Import	
Account	Entity
10200123	TX47
10299999	GB11
14000000	TX47
14300000	GB11

Validate		
Source Derivative	Transformation Rules	Target Derivative
A#[102*]= AdminExp	AdminExp	A#[*Exp]=TotExp
	ItExp	
	GenAdmExp	

Rule Expression	Expression Type	Derivative Type	Notes
A#[11*]=Cash	None	Final	Accounts that start with 11 aggregate to a new Account called Cash and stored in Stage.
A#[12*]=AR	None	Interim	Accounts that start with 12 aggregate to a new Account called AR, but not stored.
A#[1300-000;Cash]=CashNoCalc	None	Interim (Exclude Calc)	The Derivative Account Cash is excluded because the calc is excluded. The CashNoCalc interim Account is created as an aggregate of Account 1300-000, but not stored.

Rule Expression	Expression Type	Derivative Type	Notes
A#[1310-000;Cash]=CashIncludeCalc	None	Interim	The two Accounts (1310-000 and Cash) aggregate to equal the new Derivative Account called CashIncludeCalc because the calc is included. Note the use of the semicolon (;) as a list separator.

The following rules create additional rows in the Stage area when importing data based on logic.

Rule Expression	Expression Type	Derivative Type	Notes
A#[1000~1999]<<New_:E#[Tex*]=TX	None	Applies to All Types	Creates a new row in the Stage for any Account that falls between 1000 and 1999 in the source data, but will add a suffix to it. Account 1010 will create a new row for Account New_1010. The end of the rule syntax shows each Entity name starting with “Tex” will be created as the Entity called TX in these new Stage rows.

Rule Expression	Expression Type	Derivative Type	Notes
A#[2000~2999]>>_:Liability:U2#[*]= None:U3[*]=None:U4#[*]=None	None	Applies to All Types	Creates a new row in the Stage for every Account between 2000 and 2999 with a prefix. Account 2300 will come into a new row as 2300_ Liability. The rest of the rule means all UD2, UD3 and UD4 Dimension Members will be set as the None Member.

Rule Expression	Expression Type	Derivative Type	Notes
A#[3000~3999]@3:E#[Tex*]@1,1	None	Applies to All Types	Takes the first three digits of each Account between 3000 and 3999 to create new rows in Stage. Each Entity starting with Tex will be shown as “T” since the @1,1 syntax starts at the first position of the string and looks one character to the right.

Logical Operators

Logical Operators extend a normal mapping rule with VB.Net scripting functionality.

Derivative Expression Types

These are used to perform additional calculations on the transformed Member’s data.

None
This is the default and no changes will be made.

Business Rule
A business rule runs on the resulting Derivative member data. This business rule must have Derivative as its type.

Complex Expression
Write a script here instead of a shared business rule and it will run against the resulting Derivative member's data.

Multiply
This will multiply the resulting Derivative member's value by another specified value.

Divide
This will divide the resulting Derivative member's value by another specified value.

Add
This will add the resulting Derivative member's value by another specified value.

Subtract
This will subtract a specified value from the resulting Derivative Member's value.

Create If > x
A derivative record is created if the value is greater than a specified threshold.

Create If < x
A derivative record is created if the value is less than a specified threshold.

Returns the past value (lag) per the number of years set in the **Math Value** field of the rule. **Math Value** = 1 by default. The lag operator looks at the first date in the set, then returns the lag value for the number of years specified. In the data set shown below, the value 12 would be returned if the **Lag (Years)** value is set to 1:

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

Lag (Months)

Returns the past value (lag) per the number of months set in the **Math Value** field of the rule. **Math Value** = 1 by default. The lag operator looks at the first date in the set, then returns the lag value for the number of months specified. In the data set shown below, the value 9.99 would be returned if the **Lag (Months)** value is set to 1:

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

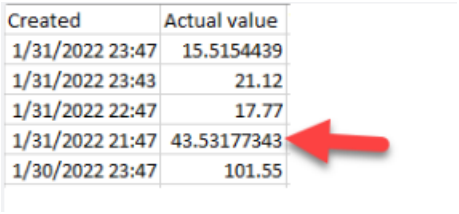
Lag (Days)

The Lag (Days) logical operator returns the past value (lag) per the number of days set in the **Math Value** field of the rule. The data set is first sorted in descending order by the Created date. The lag operator looks at the first date in the set, then returns the lag value for the number of days specified. In the data set shown below, the value 23.39486017 would be returned if the **Lag (Days)** value is set to 2:

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

Lag (Hours)

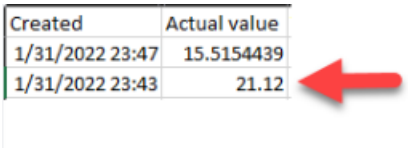
Returns the past value (lag) per the number of hours set in the **Math Value** field of the rule. The data set is first sorted in descending order by the Created date/time. The lag operator looks at the first date/time in the set, then returns the lag value for the number of hours specified. In the data set shown below, the value 43.53177343 would be returned if the **Lag (Hours)** value is set to 2:



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

Lag (Minutes)

Returns the past value (lag) per the number of minutes set in the **Math Value** field of the rule. The data set is first sorted in descending order by the Created date/time. The lag operator looks at the first date/time in the set, then returns the lag value for the number of minutes specified. In the data set shown below, the value 21.12 would be returned if the **Lag (Minutes)** value is set to 4:



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

Lag (Seconds)

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

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

Lag Change (Seconds)

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

15.51544 - 17.77 = -2.25

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

Lag Change (Minutes)

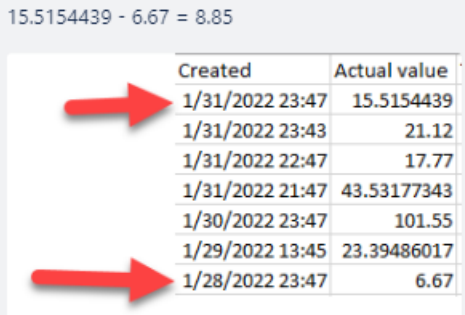
Returns the difference between the latest value and a past value (lagged value), per the number of minutes set in the **Math Value** field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of minutes set in the rule. The difference between the latest value and the lag value is returned. In the data set shown below, the value -5.60 would be returned if the **Lag Change (Minutes)** value is set to 4:

15.5154439 - 21.12 = -5.60

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

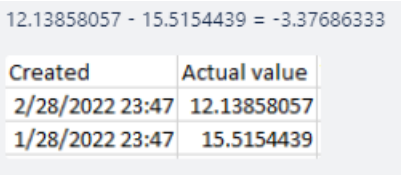
Lag Change (Days)

Returns the difference between the latest value and a past value (lagged value), per the number of days set in the **Math Value** field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of days set in the rule. The difference between the latest value and the lag value is returned. In the data set shown below, the value 8.85 would be returned if the **Lag Change (Days)** value is set to 3:



Lag Change (Months)

Returns the difference between the latest value and a past value (lagged value), per the number of months set in the **Math Value** field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of months set in the rule. The difference between the latest value and the lag value is returned. In the data set shown below, the value -3.37686333 would be returned if the **Lag Change (Months)** value is set to 1:



Lag Change (Years)

Returns the difference between the latest value and a past value (lagged value), per the number of years set in the **Math Value** field of the rule. The data set is sorted in descending order by the Created date/time. The lag change operator looks at the first date/time in the set, then looks at the lag value based on the number of years set in the rule. The difference between the latest value and the lag value is returned. In the data set shown below, the value 3.37686333 would be returned if the **Lag Change (Years)** value is set to 1:

15.5154439 - 12.13858057 = 3.37686333

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

Lag Change (Step Back)

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

15.5154439 - 6.67 = 8.8454439

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

Derivative Rule Expressions

Derivative rule expressions determine how derived records are grouped. Groupings occur on an attribute, unless the rule expression is explicitly set to “None.”

In the following example, notice that attribute A4 is not specified in the rule. As a result, groupings occur based on A4. It corresponds to the Responsibility column (assigned to column in load file) in the BI Blend tables.

Example: **A#[TicketCost]=LagDays2_GroupbyResponsibility:A1#[]=None:A2#[]=None:xBDT#[2022-01-01 00;00;00~2022-12-31 23;59;59]!M**

When a BI Blend file is loaded to the workflow, the calculation based on the logical operator is performed on each grouping.

Group By Day

Rule expressions in Transformation Rules can group derived records by day. In the rule expression shown below, the !D at the end of the expression indicates that the derivative records are grouped by day.

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

When used with a logical operator, the calculation is performed on each grouping. For example, if the Average logical operator is used on the data set shown below, the average is calculated for each day. If the day only has 1 value, that value is returned.

Created	Actual value
1/31/2022 23:47	15.5154439
1/31/2022 23:43	21.12
1/31/2022 22:47	17.77
1/31/2022 21:47	43.53177343
1/30/2022 23:47	101.55
1/29/2022 13:45	23.39486017
1/28/2022 23:47	6.67
1/27/2022 8:02	25.51319627
1/26/2022 8:02	17.69
1/25/2022 8:02	16.77
1/24/2022 8:02	13.33
1/24/2022 8:03	25.87
1/24/2022 8:04	99.22
1/24/2022 8:05	155.55

Group By Month

Rule expressions in Transformation Rules can group derived records by month. In the rule expression shown below, the !M at the end of the expression indicates that the derivative records are grouped by month.

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

When used with a logical operator, the calculation is performed on each grouping. For example, if the Average logical operator is used, the average is calculated for each month. If the month only has 1 value, that value is returned.

Group By Year

Rule expressions in Transformation Rules can group derived records by year. In the rule expression shown below, the !Y at the end of the expression indicates that the derivative records are grouped by year.

Example: **A#[TicketCost]=CountGroupByYear:A1#[]=None:A2#[]=None:A4#[*]=None:xBDT#[2015-01-01 00;00;00~2022-12-31 23;59;59]!Y**

When used with a logical operator, the calculation is performed on each grouping. For example, if the Count logical operator is used on the data set shown below, the following counts are returned for each year:

Group By All

Rule expressions in Transformation Rules can group all derived records into one group when specified in a rule expression. In the rule expression shown below, the !All at the end of the expression indicates that the derivative records are grouped into one group that contains all records.

Example: **A#[TicketCost]=CountGroupByAll:A1#[]=None:A2#[]=None:A4#[*]=None:xBDT#[2015-01-01 00;00;00~2022-12-31 23;59;59]!All**

When used with a logical operator, the calculation is performed on the whole data set. For example, if the Count logical operator is used on the data set shown below, a count of 15 is returned:

Created	Actual value
1/31/2020 23:47	15.5154439
1/30/2020 23:43	21.12
1/29/2020 22:47	17.77
1/28/2020 21:47	43.53177343
2/15/2021 23:47	101.55
2/16/2021 13:45	23.39486017
2/17/2021 23:47	6.67
3/27/2022 8:02	25.51319627
3/26/2022 8:02	17.69
3/24/2022 8:02	16.77
3/23/2022 8:02	16.77
3/22/2022 8:02	16.77
4/24/2022 8:02	16.77
4/23/2022 8:02	16.77
5/22/2022 8:02	16.77

Derivative Types

Derivative types determine if the resulting Derivative Member will be created, not created, or if the member will be calculated.

Interim
This will not be stored in the Stage area and cannot be mapped to a target member. It can be used within other subsequently run derivative rules.

Interim (Exclude Calc)
This is similar to Interim but will be excluded in other derivative calculations.

Final
This will be stored in the Stage area and available to be mapped to a target member.

Final (Exclude Calc)
This is similar to Final but will be excluded in other derivative calculations.

Check Rule
This is a custom validation rule that uses the same syntax as member filters and can be applied to the source data during the Validation task of the workflow.

Check Rule (Exclude Calc)
This is similar to Check Rule but will be excluded in other derivative calculations.

Order

The Order field allows a value to be assigned to a record which will allow a custom sort order of the mapping table.

Lookup

The Lookup transformation rule has a versatile configuration and can be used as a table for formulas, business rules or as a simple look up.

Transformation Rule Profiles

Once transformation rule groups are created, they are organized into transformation rule profiles. The profiles are assigned to workflow profiles.

Click to create a new profile.

Click to assign a rule group to a profile so users can select groups for a profile.

Under Rule Profile Settings, choose the cube and scenario type where the profile can be viewed and used when designing a workflow profile. Assign a rule profile to a workflow profile by clicking **Application** > **Workflow Profiles** > **Import** > **Integration Setting**s.

Loading Data With Excel Templates or CSV Files

Use Excel templates or CSV files to load data into stage using the Import method in Workflow, Forms data, Journals, Cell Detail, Data Attachments or to Custom Tables. An Excel file can be imported without having to configure the data source completely by setting Allow Dynamic Excel Loads to True and configuring an Excel template. This section describes the configuration required for this integration.

Loading Stage Data

Import Excel Template

When importing data into the Stage via Excel, a specific template must be used for OneStream to read and load the data. The user must first create Dimension Tokens (e.g. E#, A#, S#, etc.) to organize the data correctly. The Dimension Token specifies the specific type of data in any given column. For example, if the column header is E#, OneStream will read every row in that column as an Entity name when loading into Stage. After the Dimension Tokens are specified, create a Named Range beginning with xfd. There can multiple xfd Named Ranges across multiple tabs within an Excel workbook. The Dimension Tokens must be the Named Range's first row.

The following Dimension Tokens are used within an Import Excel Template. Please note these tokens can be in any order on the Excel template.

Dimension Tokens	Meaning
A#	Account: each row below will list the accounts to be imported
AMT#	Amount: using the AMT.ZS# header will automatically apply zero suppression to this import. Tip: Apply zero suppression to a Matrix-style Excel import template by using this same .ZS extension on the AMT column.
F#	Flow
IC#	Intercompany

Dimension Tokens	Meaning
E#	Entity
C#	Consolidation
S#	Scenario
T#	Time Period
V#	View
O#	Origin
UD1#-UD8#	Each row must have a value even if a User Defined Member is not used in the application. Create a Static Value of None for any UD Members where this applies. Ex. UD5#:[None] UX# or UDX# can be used for all User Defined Dimensions.
LB#	Label: This is used for an Account description related to a line of data. It is imported just for reference purposes and not stored in the Cube.
SI#	Source ID: This is a key for data imported into Stage. This typically includes a reference to the Entity being loaded but depends on the implementation. It is a best practice to have only one Source ID per Named Range and these can be the same or different for every Named Range imported for one Excel workbook.

Dimension Tokens	Meaning
TV#	Text Value: this is used to store large amounts of textual data.
A1# through A20#	Attribute Dimensions: these 20 Dimensions can each store 100 characters of text.
AV1# through AV12#	Attribute Value Dimensions: these 12 Dimensions can store numeric data.

Header Abbreviations

Static Value

Use :[] to fix a specific Member to the entire column creating a Static Value for the specified Source Dimension. For example, F#:[None] imports the None Flow Member for every Flow row within the Named Range. This syntax applies to all Dimension Tokens.

Data Sources allow text values to be loaded as a View Member from the same row as the numeric value. Specify #Annotation, #VarianceExplanation, #AuditComment, #Footnote, or #Assumption as the Static Text Value of the TextValue Source Dimension and a new row will be created for the comment row. For example, use TV#:[#Annotation] to add an additional Annotation row.

Business Rule

Pass a Business Rule for any specified Source Dimension to set a specific value.

```
AMT#:[]:[BusinessRuleNameThatSetsAValue]
```

Matrix Member

This repeats for each Member. For example, if there were twelve time periods in the named range the syntax would be as follows:

```
T#:[]:[]:[2012M3]
```

To use Current/Global Scenario and Time, use .C# and .G# which creates a Static Value for the Time and Scenario within the Named Range. T.C# and S.C# returns the current Workflow Time and Scenario. T.G# and S.G# returns the Global Time and Scenario.

Import Data Extracted via Data Management

Any type of data (Import, Forms, or Journals) extracted to a CSV file through a Data Management Job can be imported into Stage via an Extensibility Business Rule. This simplifies the migration of data between applications.

Example

```
Dim objXFResult As XFResult = BRApi.Finance.Data.SetDataCellsUsingCsvFile(si,  filePath,
delimiter,
       originFilter, targetOriginMember, loadZeros)
```

When using this BRApi make sure to specify the Origin Filter which determines the type of data desired from the file (Import, Forms or Adjustments), and the Target Origin Member which determines where the data will be stored upon loading the file.

Loading Form Data

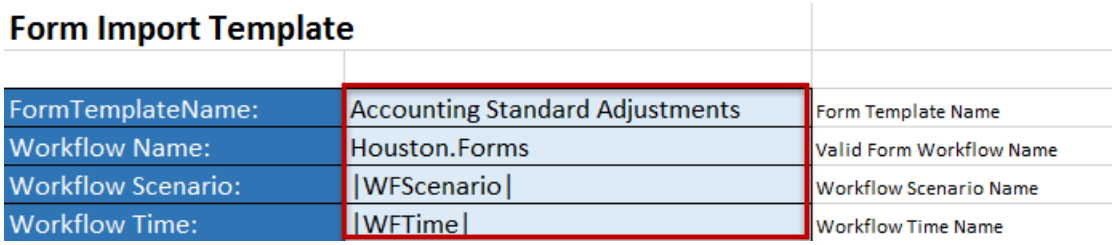
Form Excel Template

When loading Form data via Excel, a specific template must be created to determine the form properties, the Dimensions to which the data is loaded, the data entry amount, and any data attachment information.

OneStream reads this template using a specific Named Range which is explained later in this section. Ensure the following information is included in the Named Range.

Property Tokens

The first four rows of the Named Range in the Excel template must include the following token definitions:



Form Import Template		
FormTemplateName:	Accounting Standard Adjustments	Form Template Name
Workflow Name:	Houston.Forms	Valid Form Workflow Name
Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	|WFTime|	Workflow Time Name

Form Template Name

Enter the Form Template name intended for the Form data load.

Workflow Name

Enter the Form Workflow name. For example, if the name of the Workflow Profile is Houston, and the Form input type is named Forms, enter Houston.Forms.

Workflow Scenario

Enter the current Workflow Scenario such as Actual, Budget, etc. To dynamically use the current Workflow Scenario, use the |WFScenario| Substitution Variable.

Workflow Time

Enter the current Workflow Time Period. To dynamically use the current Workflow Time, use the |WFTime| Substitution Variable.

Dimension Tokens

Next, create the Dimension Tokens necessary to load the form data to the correct Dimensions in OneStream. The Dimension tokens need to be the column header for each data row. The standard tokens used determine the Cube, Entity, Parent, Account, Flow, Intercompany, the User Defined Members, and an Amount. Refer to Loading Stage Data for the syntax. The form specific tokens are as follows:

HD#

Has Data

Enter Yes or No to specify whether the row has data.

AN#

Annotation

AS#

Assumption

AD#

Audit Comment

FN#

Footnote

VE#

Variance Explanation

Header Abbreviations

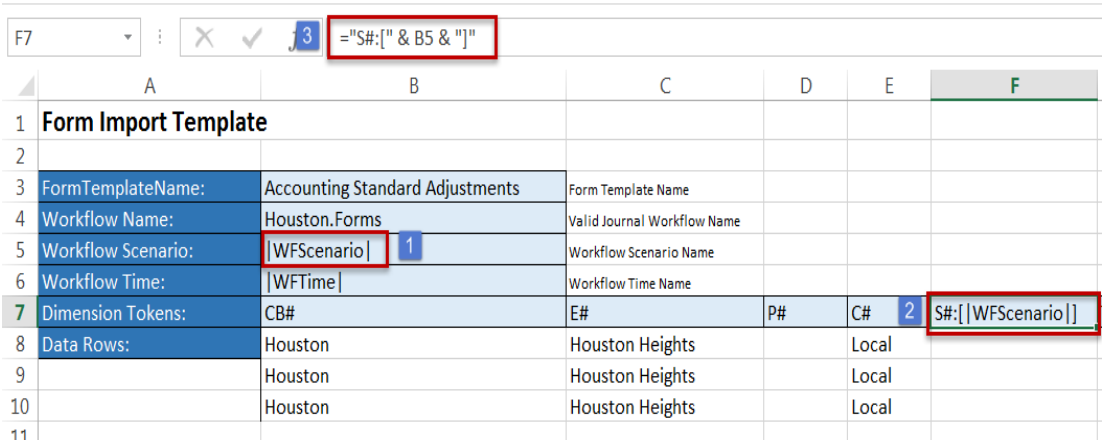
Static Value

Use :[] to fix a specific Member to the entire column creating a Static Value for the specified Source Dimension. For example, F#:[None] imports the None Flow Member for every Flow row within the Named Range. This syntax applies to all Dimension Tokens.

Using Substitution Variables

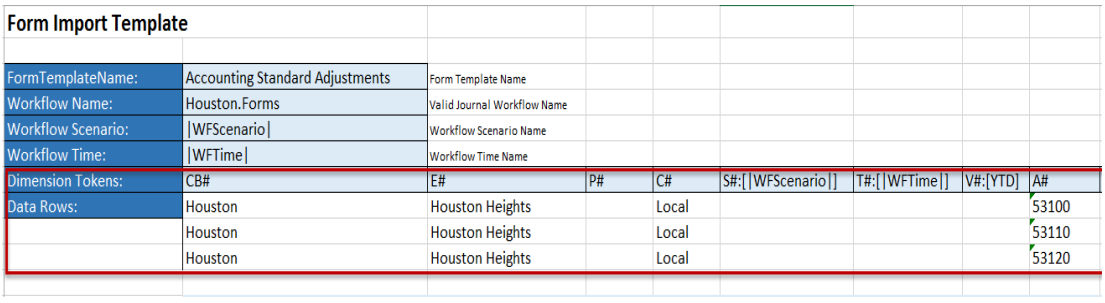
If a Substitution Variable is used to define the Workflow Scenario or Workflow Time Tokens, link the Scenario and Time Dimension Tokens using the Ampersand (&) Excel Function and referencing the Excel cell.

Example:



1. The Workflow Scenario Token, located in cell B5, is using a Substitution Variable to dynamically reference the user's current Scenario.
2. The Scenario Dimension Token needs to reference that Substitution Variable to ensure the correct Scenario is used and the template functions properly.
3. The syntax to reference the cell B5 is ="S#:[" & B5 &"]"
   This references the correct variable and displays it in the proper cell.

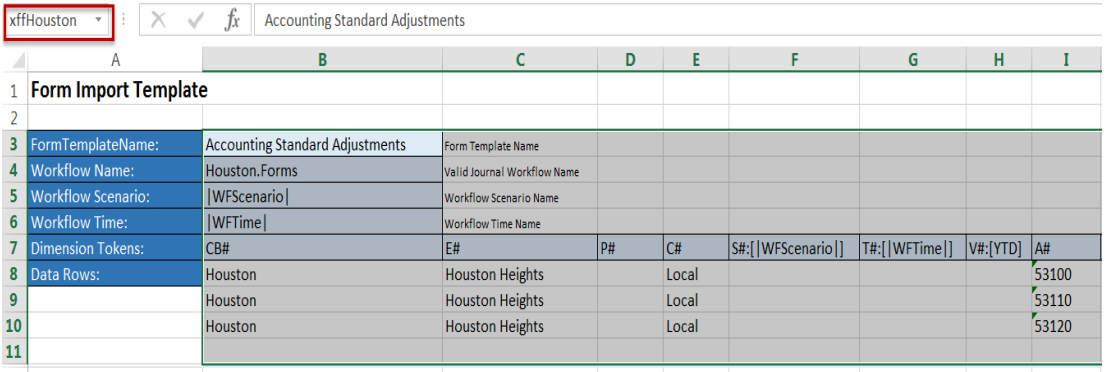
After the Dimension Tokens are configured, enter the data in the corresponding column. The Dimensions can be in any order.



F#	O#:[Forms	IC#	UD1#	UD2#	UD3#	UD4#	UD5#:[IFRS Adj]	UD6#	UD7#	UD8#	AMT#
None		None	None	None	None	None		None	None	None	9,000.00
None		None	None	None	None	None		None	None	None	9,000.00
None		None	None	None	None	None		None	None	None	12,000.00

HD#:[Yes]	AN#	AS#	AD#	FN#	VE#

The final step is to create a Named Range beginning with XFF making sure to include the definition of each property, the Dimension tokens, and the data rows. The Named Range must begin with XFF for OneStream to read and load the form data correctly. Multiple XFF Named Ranges can be used across multiple tabs.



	A	B	C	D	E	F	G	H	I
1	Form Import Template								
2									
3	FormTemplateName:	Accounting Standard Adjustments	Form Template Name						
4	Workflow Name:	Houston.Forms	Valid Journal Workflow Name						
5	Workflow Scenario:	|WFScenario|	Workflow Scenario Name						
6	Workflow Time:	|WFTime|	Workflow Time Name						
7	Dimension Tokens:	CB#	E#	P#	C#	S#:[|WFScenario|]	T#:[|WFTime|]	V#:[YTD]	A#
8	Data Rows:	Houston	Houston Heights		Local				53100
9		Houston	Houston Heights		Local				53110
10		Houston	Houston Heights		Local				53120
11									

Form Matrix Excel Template

A form matrix template is used to load multiple amount columns to multiple time periods at once. In this template, the Time Dimension Token is combined with the amount to identify which amount should load to which period.

This template uses the same property tokens as a regular Excel Form template shown above.

Form Matrix Import Template

FormTemplateName:	Accounting Standard Adjustments		Form Template Name
Workflow Name:	Houston.Forms		Valid Journal Workflow Name
Workflow Scenario:	|WFScenario|		Workflow Scenario Name
Workflow Time:	|WFTime|		Workflow Time Name

In the Matrix Form template, Amount and Time must be specified in the same column. A third Dimension can be specified (e.g., Scenario) if desired. The example below is indicating the Amount Column using AMT# and then specifying to which Time Members the Amount detail belongs.

AMT#::T#2011M1	AMT#::T#2011M2	AMT#::T#2011M3	
1,000.00	2,000.00	3,000.00	
2,000.00	4,000.00	6,000.00	
3,000.00	6,000.00	9,000.00	

Form CSV Template

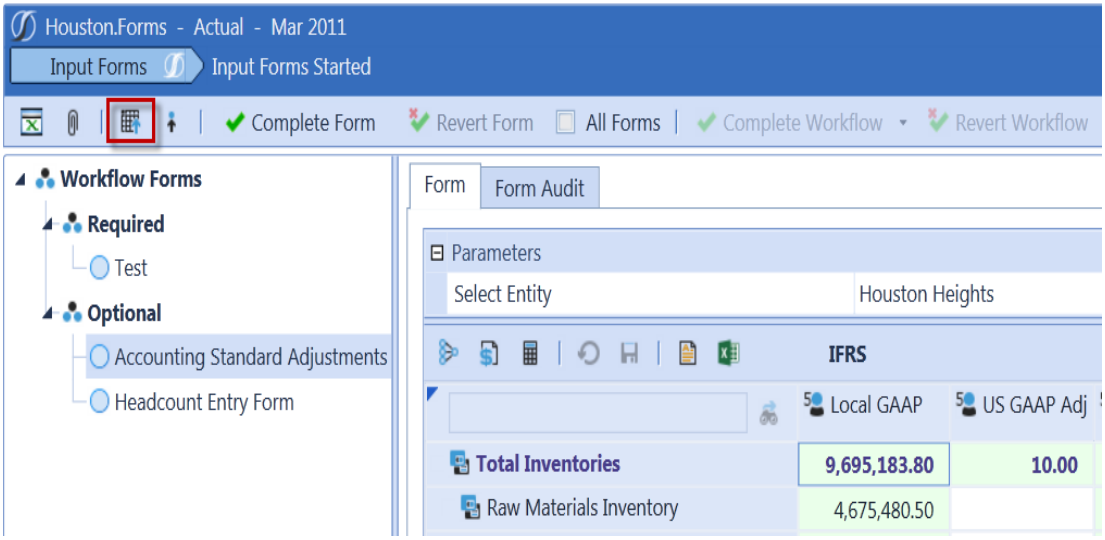
To set up a CSV template for a Form, the Header and Detail values must be specified.

	A	B	C	D	E
2	!RowType (H=Header)	FormTemplateName	WFProfileName	WFScenarioName	WFTimeName
3	!RowType (D=Detail)	FormTemplateName	Cube	Entity	Parent
	H	Accounting Standard Adjustments	Houston.Forms	Actual	2011M4
	D	Accounting Standard Adjustments	Houston	Houston Heights	
	D	Accounting Standard Adjustments	Houston	Houston Heights	
4	D	Accounting Standard Adjustments	Houston	Houston Heights	
	D	Accounting Standard Adjustments	Houston	South Houston	
	D	Accounting Standard Adjustments	Houston	South Houston	
	D	Accounting Standard Adjustments	Houston	South Houston	

1. Column A Specifies Row Type
   In the first two rows of Column A, create two Row Type Parameters specifying the Header and the Detail. In the example above, !RowType (H=Header) and !RowType (D=Detail) are used to tag the corresponding rows with H or D identifying where the Header and Detail information is located in the CSV file.
2. Row One Specifies the Headers
   After the Header Parameter is configured, enter the form column headers. The required Form Headers are FormTemplateName, WFProfileName, WFScenarioName, and WFTimeName. For more details on these, refer to Form Excel Template.
3. Row Two Specifies the Details
   After the Detail Parameter is configured, enter the form detail headers. The required Form Detail Headers are FormTemplateName, all 18 standard Dimensions, Amount, HasData, Annotation, Assumptions, AuditComment, Footnote, and VarianceExplanation. For more details on these, refer to Form Excel Template.
4. Header and Detail Tags
   The Form data is driven by how each row in column A is tagged. Any rows tagged with H load as the Headers and any row tagged with D load as the details.

Loading Form Details via Workflow

After the template is configured, users can load it directly into OneStream during the Workflow process. While the Form Input type is selected, click the icon in the Form toolbar. This allows the user to select the desired Excel or CSV template and load it into OneStream. After the file is loaded, the data will appear in the Form grid and it auto-saves upon importing to the Cube.



Loading Form Details via Business Rules

Users can load Form details from an Excel Template or CSV file by configuring an Extensibility Business Rule and using the ImportandProcessForms BRApi. Within the BRApi, define the Session Info, file path, and Form actions.

Load Example

```
'BRApi.Forms.Data.ImportAndProcessForms(si, filePath, save, complete, throwOnError)
```

Loading Journal Data

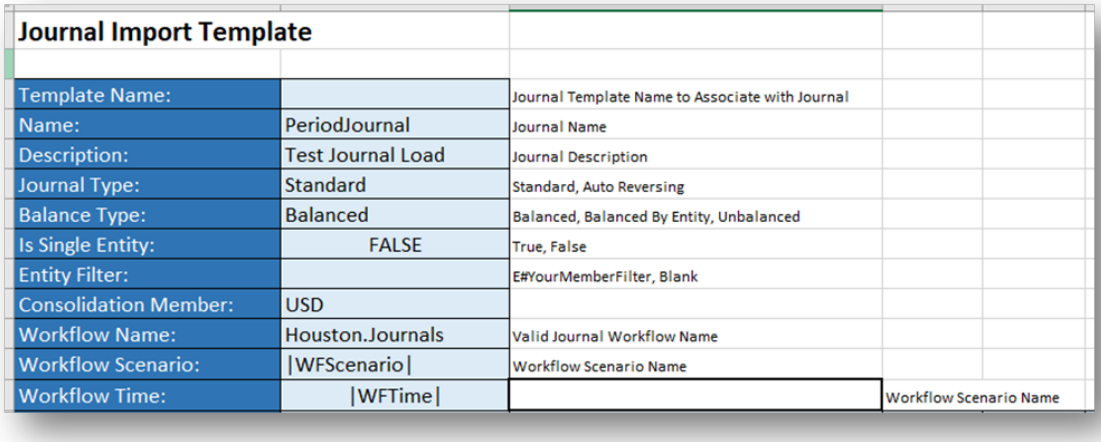
Journal Excel Template

When loading journal entries via Excel a specific template must be created to determine the journal properties, the Dimensions to which the entries are being made, and the debit and credit amounts.

OneStream reads this template using a specific Named Range which is explained later in this section. Ensure the following information is included in the Named Range.

Property Tokens

The first eleven rows of the Named Range in the Excel template must include the following token definitions:



Journal Import Template			
Template Name:		Journal Template Name to Associate with Journal	
Name:	PeriodJournal	Journal Name	
Description:	Test Journal Load	Journal Description	
Journal Type:	Standard	Standard, Auto Reversing	
Balance Type:	Balanced	Balanced, Balanced By Entity, Unbalanced	
Is Single Entity:	FALSE	True, False	
Entity Filter:		E#YourMemberFilter, Blank	
Consolidation Member:	USD		
Workflow Name:	Houston.Journals	Valid Journal Workflow Name	
Workflow Scenario:	|WFScenario|	Workflow Scenario Name	
Workflow Time:	|WFTime|		Workflow Scenario Name

Template Name

Enter a template name is applicable. If this is a free form journal, leave this blank.

Name

Enter the name of the journal.

Description

If desired, enter a description for the journal.

Journal Type*

Standard or Auto-reversing. Allocation is not supported for Excel or CSV import.

Balance Type*

Balanced, Balanced by Entity, or Unbalanced

Is Single Entity*

True or False

Entity Filter*

Use a Member Filter to specify the Entities used with this journal.

Consolidation Member

Enter the specific currency or Local Member of Consolidation.

Workflow Name

Enter the Journal Workflow name. For example, if the name of the Workflow Profile is Houston, and the Adj input type is named Journals, enter Houston.Journals.

Workflow Scenario

Enter the Workflow Scenario or make the template dynamic by entering the |WFScenario| Substitution Variable.

Workflow Time

Enter the Workflow Time or make the template dynamic by entering the |WFTime| Substitution Variable. Workflow Time support two fields. The available cell immediately to the right is option representing the CubeTimeName. This field can be used when the Scenario's Workflow Tracking Frequency is Yearly, and the Input Frequency is Monthly. For example, the Workflow Time would be |WFTime| or 2019 and the CubeTimeName would be the period to post, 2019M7.

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	|WFTime|	

When Tracking Frequency is Yearly and Input Frequency is Monthly:

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	2019M7	

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	|WFTime|	2019M7

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	|WFTime|	|WFYear|M7

*See Journal Templates Data Collection for details on these Journal properties.

Dimension Tokens

Next, create the Dimension Tokens necessary to load the journal to the correct Dimensions in OneStream. The Dimension tokens need to be the column header for each data row. The standard tokens used determine the Cube, Entity, Parent, Account, Flow, Intercompany, the User Defined Members, and a Label if needed. Refer to Loading Stage Data for the syntax. The journal specific tokens are as follows:

AMTDR#

This indicates the debited amount.

AMTCR#

This indicates the credited amount.

Header Abbreviations

Static Value

Use :[] to fix a specific Member to the entire column creating a Static Value for the specified Source Dimension. For example, F#:[None] imports the None Flow Member for every Flow row within the Named Range. This syntax applies to all Dimension Tokens.

After the Dimension Tokens are setup, enter the data in the corresponding column.

Template Example:

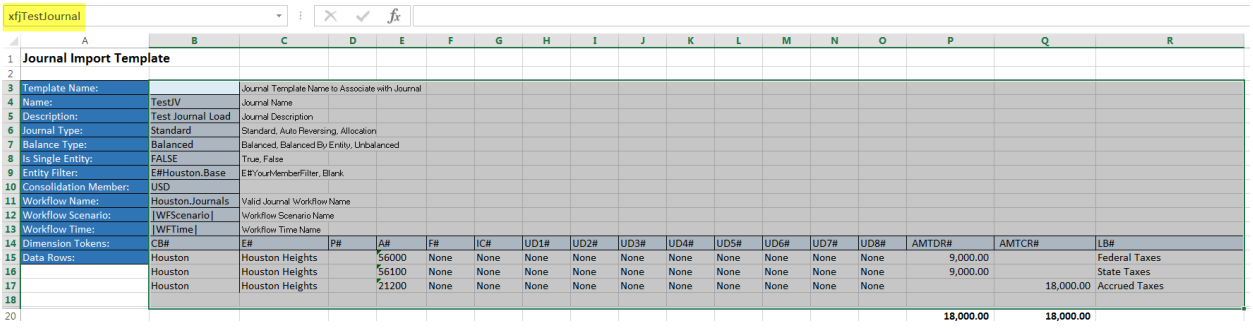
Journal Import Template

Template Name:		Journal Template Name to Associate with Journal															
Name:	TestJV	Journal Name															
Description:	Test Journal Load	Journal Description															
Journal Type:	Standard	Standard, Auto Reversing, Allocation															
Balance Type:	Balanced	Balanced, Balanced By Entity, Unbalanced															
Is Single Entity:	FALSE	True, False															
Entity Filter:	E#Houston.Base	E#YourMemberFilter, Blank															
Consolidation Member:	USD																
Workflow Name:	Houston.Journals	Valid Journal Workflow Name															
Workflow Scenario:	|WFScenario|	Workflow Scenario Name															
Workflow Time:	|WFTime|	Workflow Time Name															
Dimension Tokens:	CB#	E#	P#	A#	F#	IC#	UD1#	UD2#	UD3#	UD4#	UD5#	UD6#	UD7#	UD8#	AMTDR#	AMTCR#	LB#
Data Rows:	Houston	Houston Heights		56000	None	None	None	None	None	None	None	None	None	None	9,000.00		Federal Taxes
	Houston	Houston Heights		56100	None	None	None	None	None	None	None	None	None	None	9,000.00		State Taxes
	Houston	Houston Heights		21200	None	None	None	None	None	None	None	None	None	None		18,000.00	Accrued Taxes
															18,000.00	18,000.00	

The final step is to create a XFJ Named Range making sure to include the definition of each property, the Dimension tokens, and the data rows. The Named Range must begin with XFJ for OneStream to read and load it correctly. Multiple XFJ Named Ranges can be used within the template over multiple tabs.

> **NOTE:** Loading of Journal Templates or previously exported Journal data only requires a Parent (P#) column value to be populated if the target Consolidation dimension member being updated is OwnerPreAdj or OwnerPostAdj. Otherwise, this entry can be left blank.

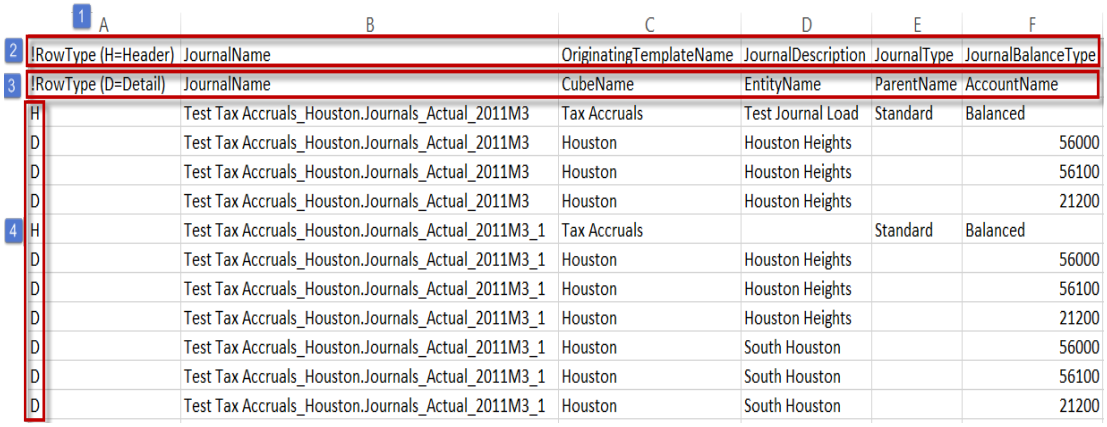
Named Range Example:



> **NOTE:** The Named Range only covers each property definition.

Journal CSV Template

To set up a CSV template for a Journal, the Header and Detail values must be specified.



	A	B	C	D	E	F
2	!RowType (H=Header)	JournalName	OriginatingTemplateName	JournalDescription	JournalType	JournalBalanceType
3	!RowType (D=Detail)	JournalName	CubeName	EntityName	ParentName	AccountName
	H	Test Tax Accruals_Houston.Journals_Actual_2011M3	Tax Accruals	Test Journal Load	Standard	Balanced
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3	Houston	Houston Heights		56000
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3	Houston	Houston Heights		56100
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3	Houston	Houston Heights		21200
4	H	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Tax Accruals		Standard	Balanced
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	Houston Heights		56000
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	Houston Heights		56100
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	Houston Heights		21200
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	South Houston		56000
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	South Houston		56100
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	South Houston		21200

1. Column A Specifies Row Type

   In the first two rows of Column A, create two Row Type Parameters specifying the Header and the Detail. In the example above, !RowType (H=Header) and !RowType (D=Detail) are used to tag the corresponding rows with H or D identifying where the Header and Detail information is located in the CSV file.

2. Row One Specifies the Headers

   After the Header Parameter is configured, enter the Journal Column Headers. The required Journal Headers are JournalName, OrigniatingTemplateName, JournalDescription,JournalType, JournalBalanceType, IsSingleEntity, EntityMemberFilter, ConsName, WFProfileName, WFScenarioName, WFTimeName and CubeTimeName. For more details on these, refer to the Journal Excel Template.

3. Row Two Specifies the Details

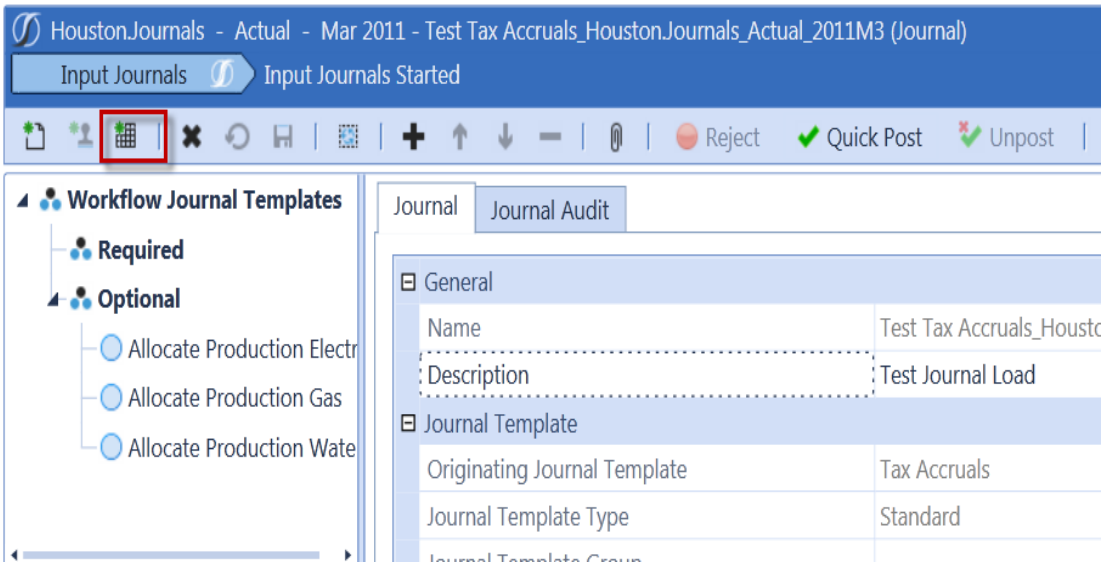
   After the Detail Parameter is configured, enter the Journal Detail Headers. The required Journal Detail Headers are JournalName, CubeName, EntityName, ParentName, AccountName, FlowName, ICName, all UDNames, DebitAmount, CreditAmount, and LineDescription. For more details on these, refer to Journal Excel Template.

4. Header and Detail Tags

   The Journal data is driven by how each row in column A is tagged. Any rows tagged with H load as the Headers and any row tagged with D load as the details.

Loading Journal Details via Workflow

After the template is configured, users can load it directly into OneStream during the Workflow process. While the Journal Input type is selected, click the icon in the Journal toolbar. This lets you select the desired Excel or CSV template and load it into OneStream. After the file is loaded, the journal line items will appear in the journal and the user can save it to the Cube.



Extracting/Loading Journal Details via BRApi

Users can export journal details into a CSV or XSLX file and load journals from a CSV or XSLX file by configuring an Extensibility Business Rule. This allows journal details to be extracted from one application and loaded into another. Users can also extract the journal, make changes, and re-load.

Extract Journals to CSV

Use BRApi.Journals.Data.ExportJournalstoCSV and define the session, filepath, Workflow Profile, Scenario, Time Filter, and Journal Status.

BRApi.Journals.Data.ExportJournalsToCsv(si, filePath, "Houston", "Actual", "T#|WFYear|.Base", "Posted")

Extract Journals to XSLX

Export journals associated with defined parameters to an XSLX file using the ExportJournalsToXSLX BRApi. It is sometimes easier to read and modify journal data in an XSLX format as compared to CSV files that are generated by the ExportJournalsToCSV BRApi. Use the parameters listed in the following table.

Parameter	Description
si	Session information
serverFilePath	Name and location to save the file
wfProfileName	Name of the Workflow Profile
wfScenarioName	Name of the Scenario

Parameter	Description
wfTimeMemberFilter	Filters workflow profile times. For example, if the scenario's Workflow Tracking Frequency is set to Quarterly, and Input Frequency is set to Monthly, the filter searches for journals associated with the Workflow Tracking Frequency. Time must be prefixed with T#. You can type a comma separated list of times such as T#2022H1, T#2022H2.
journalName	Filters journals that contain the specified text
journalStatus	Status of the journal. You can search by status or use All to return all statuses. You cannot type a comma separated list.

The following code example uses the ExportJournalsToXslx BRApi:

```vb
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.Extender.ExportJournalstoXSLX
    Public Class MainClass
        Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Object, ByVal args As ExtenderArgs) As Object
            Try
                Dim configSettings As AppServerConfigSettings = AppServerConfig.GetSettings(si)
                Dim sourcefolder As String = BRApi.Utilities.GetFileShareFolder(si,FileShareFolderTypes.FileShareRoot,Nothing) & "\System\Contents\Export"
                Dim journalFilePath As String = sourcefolder & "\" & "Journal_Export.xlsx"
                Dim filePath As String = BRApi.Journals.Data.ExportJournalsToXlsx(si, journalFilePath, "Houston", "Actual", "T#2023M5", "", "All")

                Return Nothing
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
    End Class
End Namespace
```

Load Example

Use BRApi.Journals.Data.ImportAndProcessJournals and define the session, filepath, and journal tasks to complete upon loading the journal details.

'BRApi.Journals.Data.ImportAndProcessJournals(si, filePath, save, submit, approve, post, unpostAndOverwrite, throwOnError)

Loading Cell Detail

Cell Detail Excel Template

When loading Cell Detail via Excel, a specific template must be created to determine the Cell Detail Dimension Tokens and each Cell Detail line data.

OneStream reads this template using a specific Named Range which is explained later in this section. Ensure the following information is included in the Named Range.

Dimension Tokens

The first 19 rows of the Named Range in the Excel template must include the following token definitions:

CubeName	Houston
EntityName	Houston Heights
ParentName	
ConsName	USD
ScenarioName	ActualTest
TimeName	2011M3
AccountName	56100
FlowName	None
Origin	Forms
ICName	None
UD1Name	None
UD2Name	None
UD3Name	None
UD4Name	None
UD5Name	IFRS Adj
UD6Name	None
UD7Name	None
UD8Name	None
DetailViewType	Periodic

The Cube and each Dimension Member must be specified. All User Defined Members must be specified. If a specific User Defined Member is not used in the application, enter None.

Next, create the Dimension Tokens necessary to load each Cell Detail line. The specific tokens are as follows:

Dimension Tokens:		AMT#	LIT#	AW#	CL#	LB#
Data Rows:		700.00	Default	1.00		Description or Detail

AMT#

Amount

LIT#

Line Item Type

AW#

Aggregation Weight

CL#

Classification Type

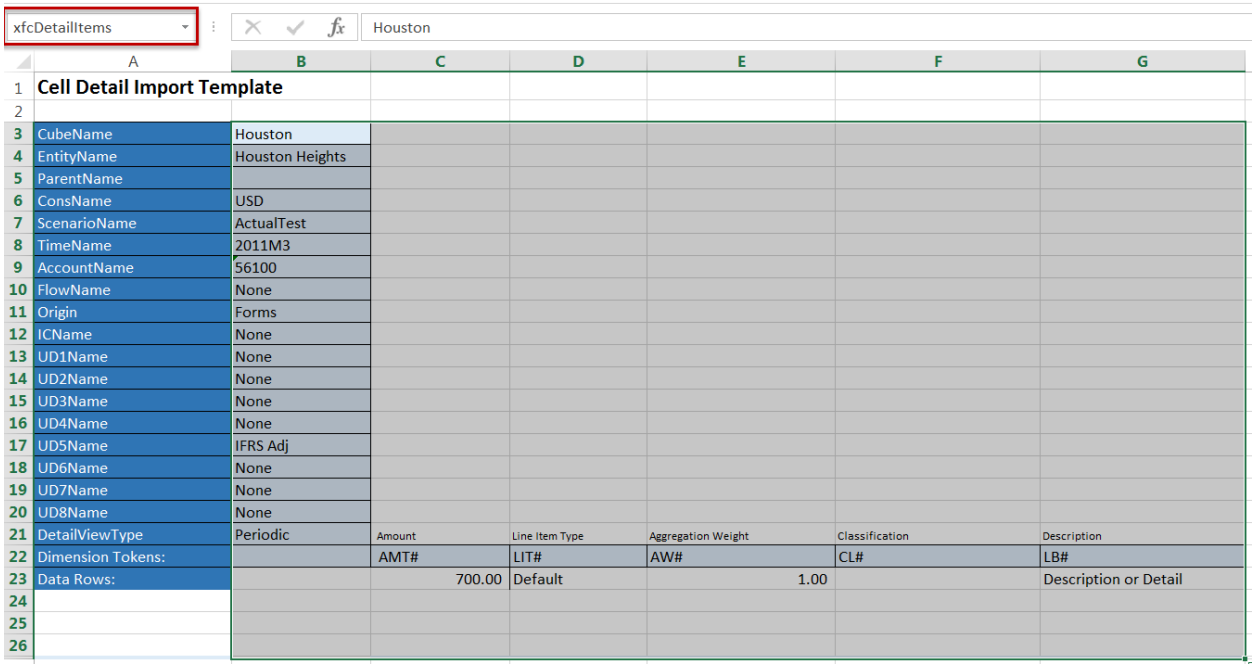
LB#

Label allowing users to add a description or additional detail.

After the Dimension Tokens are configured, enter the data in the corresponding column. The Dimensions can be in any order.

CubeName	Houston					
EntityName	Houston Heights					
ParentName						
ConsName	USD					
ScenarioName	ActualTest					
TimeName	2011M3					
AccountName	56100					
FlowName	None					
Origin	Forms					
ICName	None					
UD1Name	None					
UD2Name	None					
UD3Name	None					
UD4Name	None					
UD5Name	IFRS Adj					
UD6Name	None					
UD7Name	None					
UD8Name	None					
DetailViewType	Periodic	Amount	Line Item Type	Aggregation Weight	Classification	Description
Dimension Tokens:		AMT#	LIT#	AW#	CL#	LB#
Data Rows:		700.00	Default	1.00		Description or Detail

The final step is to create a Named Range beginning with XFC making sure to include the definition of Dimension token, and the data rows. The Named Range must begin with XFC for OneStream to read and load the Cell Detail correctly. Multiple XFC Named Ranges can be used across multiple tabs.

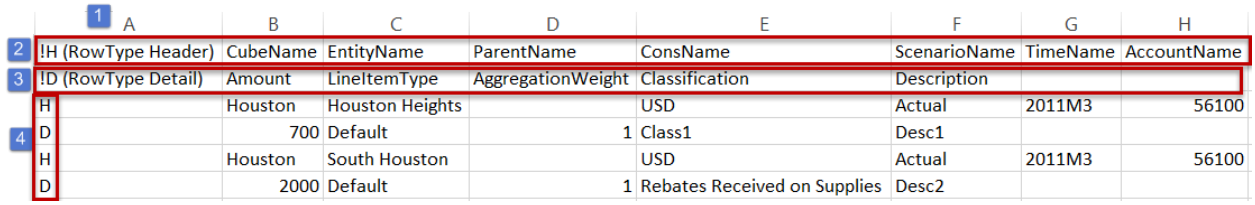


xfcDetailItems | Houston

	A	B	C	D	E	F	G
1	Cell Detail Import Template						
2							
3	CubeName	Houston					
4	EntityName	Houston Heights					
5	ParentName						
6	ConsName	USD					
7	ScenarioName	ActualTest					
8	TimeName	2011M3					
9	AccountName	56100					
10	FlowName	None					
11	Origin	Forms					
12	ICName	None					
13	UD1Name	None					
14	UD2Name	None					
15	UD3Name	None					
16	UD4Name	None					
17	UD5Name	IFRS Adj					
18	UD6Name	None					
19	UD7Name	None					
20	UD8Name	None					
21	DetailViewType	Periodic	Amount	Line Item Type	Aggregation Weight	Classification	Description
22	Dimension Tokens:		AMT#	LIT#	AW#	CL#	LB#
23	Data Rows:		700.00	Default	1.00		Description or Detail
24							
25							
26							

Cell Detail CSV Template

To set up a CSV template for Cell Detail, the Header and Detail values must be specified.



	1 A	B	C	D	E	F	G	H
2	!H (RowType Header)	CubeName	EntityName	ParentName	ConsName	ScenarioName	TimeName	AccountName
3	!D (RowType Detail)	Amount	LineItemType	AggregationWeight	Classification	Description		
4	H	Houston	Houston Heights		USD	Actual	2011M3	56100
	D	700	Default	1	Class1	Desc1		
	H	Houston	South Houston		USD	Actual	2011M3	56100
	D	2000	Default	1	Rebates Received on Supplies	Desc2		

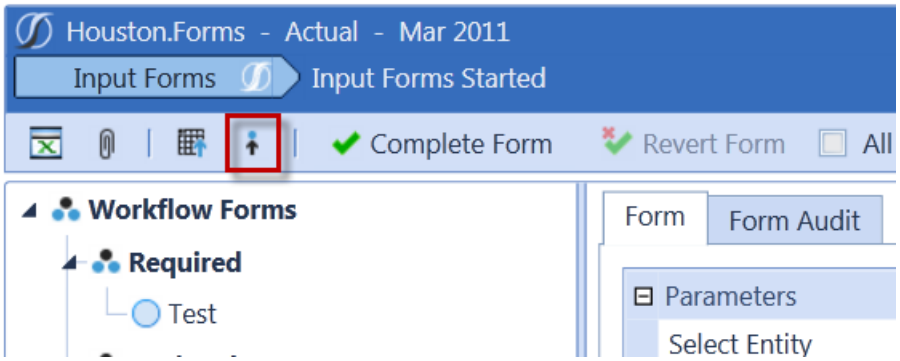
1. Column A Specifies Row Type

   In the first two rows of Column A, create two Row Type Parameters specifying the Header and the Detail. In the example above, !RowType (H=Header) and !RowType (D=Detail) are used to tag the corresponding rows with H or D identifying where the Header and Detail information is located in the CSV file.

2. Row One Specifies the Headers
   After the Header Parameter is configured, enter the Cell Detail column headers. The required Cell Detail Headers specify the Cube and all 18 Dimension Members. For more details on these, refer to Cell Detail Excel Template.
3. Row Two Specifies the Details
   After the Detail Parameter is configured, enter the Cell Detail detail headers. The required Detail Headers are Amount, LineItemType, AggregationWeight, Classification, and Description. For more details on these, refer to Cell Detail Excel Template.
4. Header and Detail Tags
   The Cell Detail's data is driven by how each row in column A is tagged. Any rows tagged with H load as the Headers and any row tagged with D load as the details.

Loading Cell Detail via Workflow

After the template is configured, users can load it directly into OneStream during the Workflow process. While the Form Input type is selected, click the icon in the Form toolbar. This allows the user to select the desired Excel or CSV template and load it into OneStream. After the file is loaded, it has been successfully stored to the Cube.



Extracting/Loading Cell Detail via Business Rules

Users can export Cell Details into a CSV file and load Cell Details from a CSV file by configuring an Extensibility Business Rule. This allows Cell Detail to be extracted from one application and loaded into another. Users can also extract the Cell Detail, make changes, and re-load.

Extract
Use BRApi.Finance.Data.ExportCellDetailtoCSV and define the session, filepath, the Entity Dimension, the Entity Member Filter, the Scenario and the Time Member Filter.
BRApi.Finance.Data.ExportCellDetailToCsv(si, filePath, entityDimensionName, entityMemberFilter, scenarioName, timeMemberFilter)

Load Example
Use BRApi.Finance.Data.ImportCellDetail and define the session and filepath.
BRApi.Finance.Data.ImportCellDetail(si, filePath, throwOnError)

Exporting Data Attachments

Exporting Data Attachments via Business Rule

You can export data attachments to perform variety of actions such as retrieve the file name, read the contents, store them in a File Explorer folder, or import them into another application. This process is configured in an Extensibility Business Rule using the following API.

Example:

```
Dim objDataAttachmentList As DataAttachmentList = BRApi.Finance.Data.
GetDataAttachments(si, memberScript, includeFileBytes)
```

Exporting Data Attachment Text

You can export the Data Attachment Text field to a CSV file using an Extensibility Business Rule. For example, use BRApi.Finance.Data.ExportCellTextToCSV and define the session, filepath, the Entity Dimension, the Entity Member Filter, the Scenario and the Time Member Filter.

```
BRApi.Finance.Data.ExportCellTextToCsv(si, cellTextFilePath,
entityDimensionName, entityMemberFilter, scenarioName, timeMemberFilter)
```

Loading Excel Templates to Custom Tables

OneStream Solutions typically have related SQL Server tables. Other custom solutions may also include adding custom SQL Server tables. This allows users to load data to these custom tables using an Excel template. The mechanism for loading these tables could be through the user interface of a OneStream Solution or through OneStream Extensibility Rules.

OneStream reads this template using a specific Named Range which is explained later in this section. Ensure the following information is included in the Named Range.

In the first three rows of the Named Range in Column A, specify the following:

Database Location
Application or System specifies which database contains the custom tables.

Table Name
Custom tables only; enter the Table name

Load Method
The load method determines the action and any additional criteria for the action.

The syntax is:
Action:[Where Clause Criteria] (Where Clause Criteria is optional)

Load Method Definitions

Merge

If there are no criteria, Merge updates the data if it finds a matching key, otherwise it inserts it

Merge Where Clause Criteria Example

First, this will clear the values for emp1 and then Merge

Merge:[EmployeeID = ‘emp1’]

Merge Where Clause Criteria with Substitution Variable Example

Substitution Variables can be used in the Where Clause Criteria

Merge:[WFProfileName = ‘|WFProfile|’]

Replace

If there are no criteria, Replace clears everything first. By default, instead of merging, it clears the entire table. This will perform better for high volume because it does not try to match rows from the file to the table. An error will occur if it finds a match.

Replace Where Clause Criteria Example

This does not try to locate, it only does inserts and appends.

Replace:[EmployeeID = ‘emp1’]

Replace Where Clause Criteria with Substitution Variable Example

Replace:[WFProfileName = ‘|WFProfile|’]

Next, define the Field Types and Field Names beginning in Column A Row 4 and spanning as many columns as necessary.

The column definition syntax is:

```
FieldType#:[FieldName]:StaticValue(optional):DefaultValue(optional)
```

Field Type

This relates to the column name in the table.

xfGuid
Unique identifier [SQL = uniqueidentifier]

xfText
Text defined column in the table [SQL = nvarchar, nchar, ntext]

xfInt
Short integer (4 byte integer) [SQL = int]

xfBit
0,1 (True, False) [SQL = bit]

xfDec
Decimal [SQL = Decimal (28,9)]

xfDbl
Floating point number (8 byte floating) [SQL = Float]

xfDateTime
Date [SQL = datetime]

Field Name
This is specific to the SQL table to be loaded.

StaticValue
Whatever is specified as the Static Value will override every row for that column regardless if it is blank or not.

StaticValueExample
This example will override all rows and enter 50,000 as the Static Value.
xfDec#:[Salary]:50,000

DefaultValue
This only applies to blank rows.

> **NOTE:** If something is specified in the Static Value, it will ignore whatever is in the DefaultValue.

Default Value Example

This example will enter a New Guid for all blank rows in the column.

xfGuid#:[EmployeeID]::NewGuid

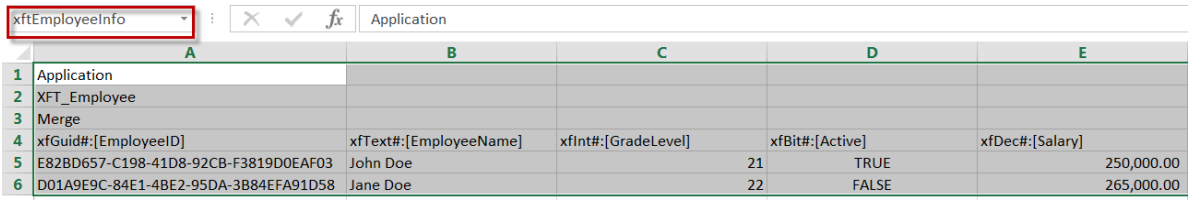
Substitution Variable Example

Substitution Variables can be used in both StaticValue and DefaultValue.

xfText#:[EmployeeName]::|Username|

	A	B	C	D	E
1	Application				
2	XFT_Employee				
3	Merge				
4	xfGuid#:[EmployeeID]	xfText#:[EmployeeName]	xfInt#:[GradeLevel]	xfBit#:[Active]	xfDec#:[Salary]
5	E82BD657-C198-41D8-92CB-F3819D0EAF03	John Doe	21	TRUE	250,000.00
6	D01A9E9C-84E1-4BE2-95DA-3B84EFA91D58	Jane Doe	22	FALSE	265,000.00

Finally, create a Named Range beginning with XFT making sure to include the entire template.



	A	B	C	D	E
1	Application				
2	XFT_Employee				
3	Merge				
4	xfGuid#:[EmployeeID]	xfText#:[EmployeeName]	xfInt#:[GradeLevel]	xfBit#:[Active]	xfDec#:[Salary]
5	E82BD657-C198-41D8-92CB-F3819D0EAF03	John Doe	21	TRUE	250,000.00
6	D01A9E9C-84E1-4BE2-95DA-3B84EFA91D58	Jane Doe	22	FALSE	265,000.00

After the template is complete, it is ready to be loaded into the custom table. If this is being used in conjunction with a OneStream Solution, refer to the Solution for further instructions on how to load the template to the table. If this is being loaded via an Extensibility Business Rule, refer to the following example.

Example

```
Dim fieldTokens As New List(Of String)
fieldTokens.Add("xfGuid#:[EmployeeID]::NewGuid")
'fieldTokens.Add("xfGuid#:[EmployeeID]")
```

```
fieldTokens.Add("xfText#:[EmployeeName]")
'fieldTokens.Add("xfText#:[EmployeeName]::|Username|")
fieldTokens.Add("xfInt#:[GradeLevel]")
fieldTokens.Add("xfBit#:[Active]")
fieldTokens.Add("xfDec#:[Salary]")
fieldTokens.Add("xfDbl#:[VacationDays]")
fieldTokens.Add("xfDateTime#:[HireDate]")
BRApi.Utilities.LoadCustomTableUsingDelimitedFile(si, SourceDataOriginTypes.FromFileShare,
filePath,
       Nothing, ",", dbLocation, tableName, loadMethod, fieldTokens, True)
```

Importing Text-based Data

To map cell text comments from a file into OneStream, here are a few tips to get started:

1. On the Cube's Integration Tab, ensure that the TextValue field is enabled for the desired Scenario Type. This is needed to import the actual text.
2. Create a Data Source to import data for the Scenario Type.

   > **NOTE:** TextValue is one of the mappable Dimensions.

3. Hardcode the View Dimension to import to the Annotation Member, or one of the other View Dimension Comment Members such as VarianceExplanation.
4. Map the comment text to the TextValue Dimension.
5. There may not be an Amount to bring in but select a column that has a decimal value in each row and a comment and link the Data Source to that column. These numbers will come into the Stage but will not end up in the Cube because they are mapped to an Annotation-type View Member.

Long Running Requests

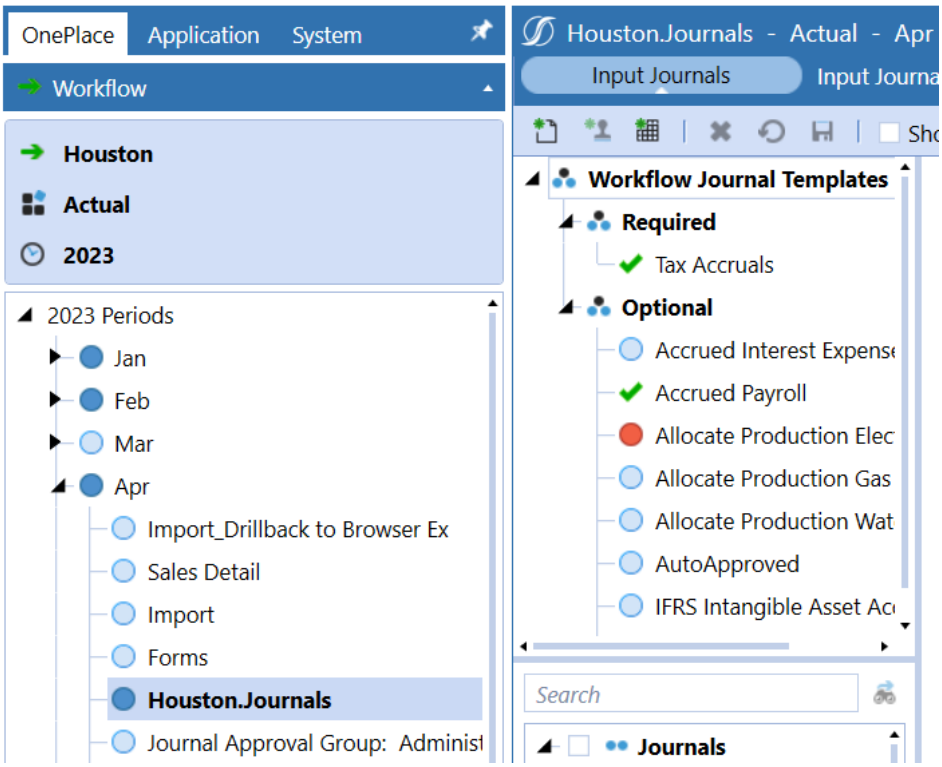
Tasks that timeout can be resolved by OneStream Support. To contact OneStream Support, register https://www.onestream.com/support/.

For additional information, see OneStream Web API Endpoints.

Journals Overview

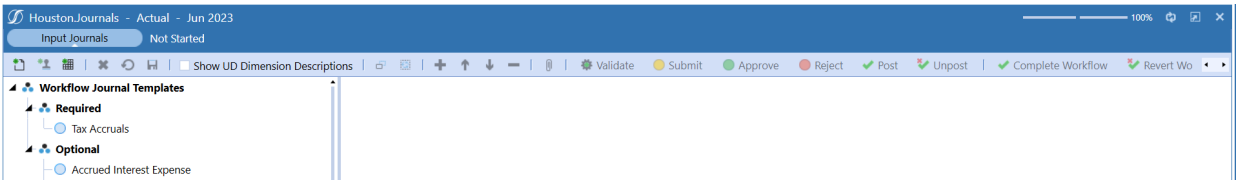
Journals are used to record business transactions in the accounting records of a business from an enterprise resource planning perspective. Journals are typically used in financial consolidation projects to record closing adjustments. Journals display account amounts and indicate whether the accounts are balanced. In the OneStream app, journals are written to the AdjInput member of the origin dimension.

You can open or create journals from the OnePlace Workflow tab. Select a Workflow Profile, Scenario, Time Period, and then click the journal you want to open.



Journals Toolbar

The Journals toolbar contains many tools for working with journals. This section describes each tool in detail.



Input Journals

Once in this step, there will be two options under Workflow Journal Templates: Required and Optional. The required Journals will have to be completed before you can move on from this step.

Create Journal

Click this to create a Journal if a template is not available. If this button is disabled, it is because the workflow profile is configured to only allow you to create journals based on existing templates.

Create Journal Using Template

Use this to create a Journal using the selected template. Manually enter the required Journal data and Click Save.

Create Journal Using Excel or Comma Separated Values File

Click this to load a journal entry via Excel template or CSV template. See Loading Journal Data in Data Collection Guides for more details on using these templates.

Copy Selected Journals

Copy existing journals into other time periods and scenarios. This is useful when you are creating similar journals, managing recurring journals, or copying Actual data into budget or what-if scenarios.

1. Select the journals you want to copy and click **Copy Selected Journals**. You can select journals in any status but when copied, they will be set to Working status in the new time period.
2. Click the drop-down arrow in the Target Scenario field.
3. Make a selection. The selection applies to all journals in that single-batch copy.
4. For each journal, select one or more time periods.
5. Type a new journal name and description.
6. You can include attachments by placing a check in the Include Attachments checkbox.

Administrators and members of the Journal Process Group can copy journals into a specified Adjustment Workflow Profile.

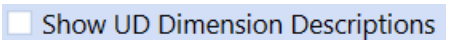
> **NOTE:** After journals are copied, WfProfile_WfUnit_Scenario_TimePeriod_Index is appended to the journal name.

Delete Selected Item

Delete one or more journals simultaneously. Journals in Working or Rejected status can be deleted. This saves time when you need to remove multiple journal entries quickly and efficiently. Select the boxes next to the journals you want to remove then click this button and confirm.

Reapply Template Settings to Journal

This will clear the data from the selected Journal and return it to the original template.



Show UD Descriptions

Check this checkbox to display descriptions for the eight UD Dimension Types. They will display next to the Dimension Type in the Journal POV and the line item headers. Descriptions are defined in the Application Properties, and can provide clarity on the purpose of the dimensions.

> **NOTE:** This feature applies to the Journal page only. Reports, BRApis and Excel/CSV templates related to journals remain unchanged.

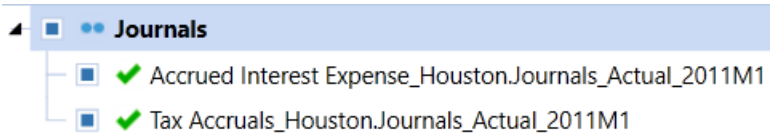
Export

Export journals from within the journals page. This is useful when you are more comfortable using a spreadsheet tool to edit journal data. You can export data from one area in OneStream and import it into another area or application.

1. Click **Export** to open the **Select Journals to Export** dialog. Workflow profile and scenario are locked.
2. Type a **Time Filter** using the T# syntax. For example, T#2023M1, or for multiple time periods, T#2023M1,T#2023M2.
3. Type a **Journal Status**. Select a single journal status or select **All** to include all statuses.
4. Type a **Journal Name**. This optional step will find all journals that contain the text you type.
5. Click **OK** to generate an XLSX file with journals that match the selected criteria.
6. In the **File Explorer** dialog, browse to where you want to save the XLSX file. Data from all journals that match the selection criteria is saved.

Journal Entry Checkboxes

Select multiple journal entries in the same state (e.g., completed, posted, approved, etc.) in order to Submit, Post, Approve, or Quick Post them at the same time. Select multiple journal entries in either the Working or Rejected status to delete them at the same time.



Validate

You can ensure that dimension members selected in the journal are valid cube intersections based on constraints configured in the dimension library. If validation errors exist, they will display in a Validation pane allowing you to view them as you make edits to the journal. This helps you save time by validating entries earlier in the process.

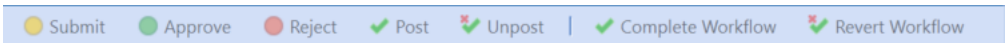
You must have the Journal Processing security role to use this feature and dimension member constraints must first be configured in the dimension library.

To use this feature:

1. Open a journal that is in Working status.
2. Click **Validate**.

If invalid data is found, the Validation pane will display messages that detail each entry error.

Once a Journal is completed, select Quick Post or Post. When all required Journals and any optional Journals are finished, click Complete Workflow and a green check will appear for the Journals Workflow Step.



Depending on the security configuration, there are multiple options for Journals. If full security is in place, the end user will be able to create and submit. The approver will approve or reject, then the end user can post and complete the Workflow.

Journal Line Items

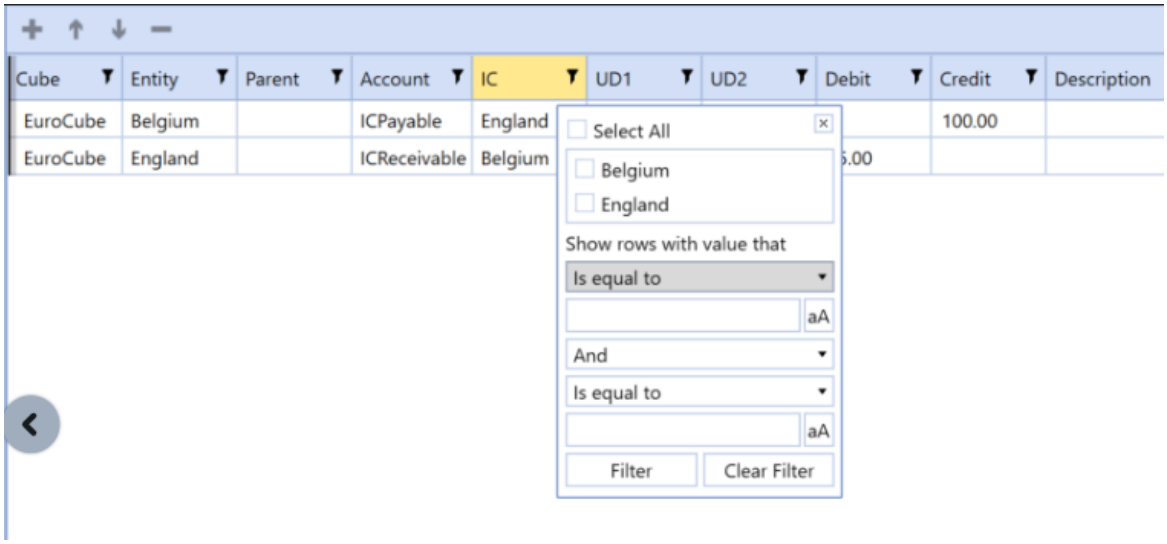
Journal data is entered into line items. You can add line items to reflect the Dimensions they are adjusting. Any dimensions not in line items must be configured in the Journal POV.

To select a dimension member in journal line items:

1. Double-click the Entity, Account, Flow IC, or UD1-UD8 field.
2. If you know the member name, type it into the text field. Repeat for all fields as needed.
3. If you do not know the member name, use the Select Member dialog to locate the member.

> **NOTE:** If the dimension members have a description, the members are displayed in the journal line items. For each dimension, you can set a preference to hide or show the description in journal line items. These preferences are saved on your local machine in OneStream user settings.

You can filter journal line items by clicking the filter icon in the column headings. A drop-down selection box appears where you can choose from various filtering options.



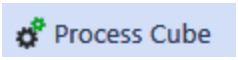
This is useful when you are working with journals that contain a large amount of line items, making it easier to locate a specific subset of line items.

There are four line item actions buttons that sit just above the line item entries that aid in editing the journal line items:

- **Add**: Creates a new journal line item. If an existing line is already selected, Add copies the selection. If no line is selected, the new line item is blank.
- **Up**: Moves the selected item up in the list.
- **Down**: Moves the selected item down in the list.
- **Remove**: Deletes the selected line item.

Process

Once in this step, click Process Cube in order to process the loaded data.



This icon performs No Calculate and all the standard Calculate, Translate, and Consolidate options using Calculation Definitions. Each added line item can be filtered by entity for reviewer level processes. Once the Cube is processed, the Process task will change from blue to green and move to the Confirm task. No Calculate can be used to allow the assignment of Data Management Sequences.

Confirm

This step runs the Confirmation Rules defined for this particular Workflow. This will immediately inform users if they have passed or failed the data quality rules. The two types of statuses for this step are Warning or Error. Warning means the user is outside of the threshold, but it will not stop the process. Error means the user is outside of the threshold and this will stop the process and turn this step to red. If anything has failed, revisit one or many of the previous steps to make sure the data is accurate and complete. Once the data has passed all quality rules, the Confirm task will change from blue to green and move to the Certify task.

Certify

This is typically the final step in the Workflow process. This certifies and completes the phase of the Workflow.

Some questions may need to be answered regarding the processes. Click each set of questions under the Questionnaires area. Answer the questions by clicking in the response cell and selecting the correct answer. Comments can also be added in order to explain the answers. The status will be displayed on the right. When this is completed, click on the Set Questionnaire Status icon and select Completed and then OK.

Set Certification Status

After each group of questions is completed, the Set Certification Status icon will be enabled. Click this and select Certify in order to certify the data as complete and accurate. This will give the final green check for the month being processed and the data can now be trusted as complete and accurate by any stakeholder that is analyzing this information.

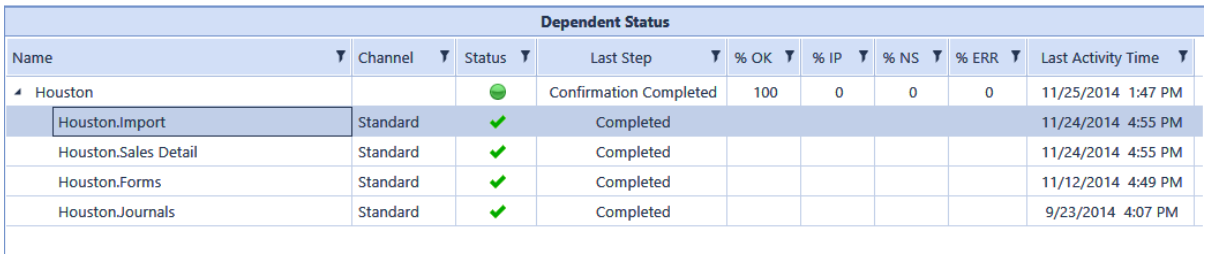
Quick Certification

This is a one click option to expedite the Certify process. No questions need to be answered. This will give the final green check for the month being processed and the data can now be trusted as complete and accurate by any stakeholder that is analyzing this information.

This data can now be used for Consolidations by users or managers responsible for this workflow. They can look at data as it moves up and perform their own top side adjustments, confirmations, and certifications at as many levels as is appropriate for the organization.

Dependent Status

Click on Dependent Status to see the status of each required Workflow task to ensure they are all completed. This will display the Workflow Profile name and all input types, the Workflow Channel, the status of each input type, the last step completed for each input type, the percentage of each step that is OK, In Process, Not Started, and steps with errors, and a record of when the last activity took place for each step. See Right-Click Options for details on these right-click functions.



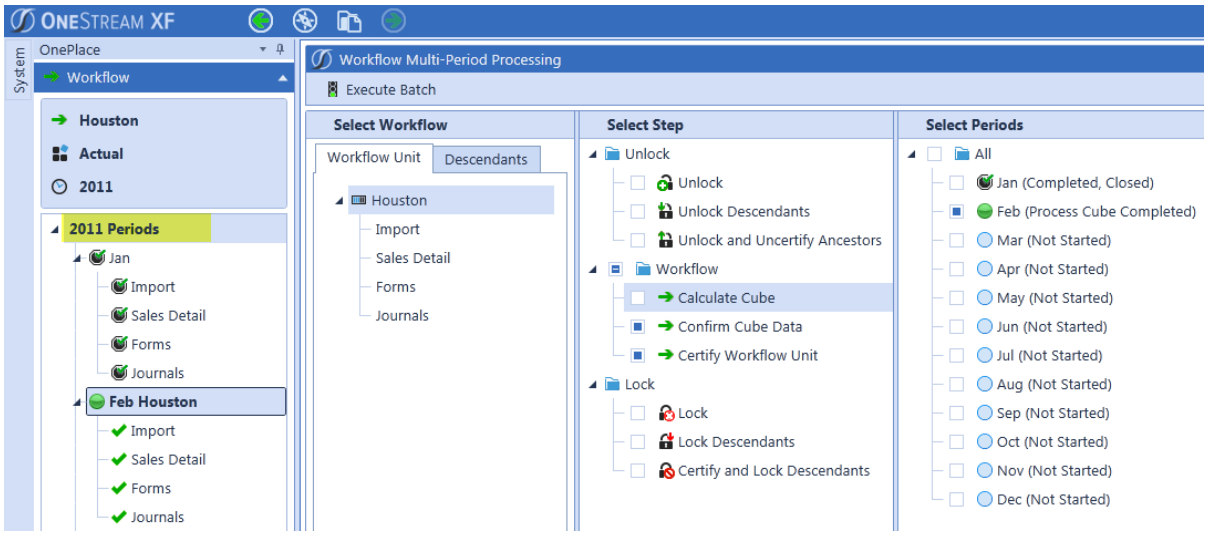
Dependent Status

Name	Channel	Status	Last Step	% OK	% IP	% NS	% ERR	Last Activity Time
Houston			Confirmation Completed	100	0	0	0	11/25/2014 1:47 PM
Houston.Import	Standard	✔	Completed					11/24/2014 4:55 PM
Houston.Sales Detail	Standard	✔	Completed					11/24/2014 4:55 PM
Houston.Forms	Standard	✔	Completed					11/12/2014 4:49 PM
Houston.Journals	Standard	✔	Completed					9/23/2014 4:07 PM

For more details on Workflow, see Workflow in Workflow.

Multi-Period Processing

Click on the Year in the Navigation Pane to enact Multi-Period Processing options:



From here, perform multiple workflow tasks for one to many time periods, as shown above.

Analysis Pane

In each workflow profile for a defined period there will be an Analysis Pane under the Status Pane. For example: under Process, Confirm, and Certify tasks.

Cube Views and Dashboards for Analysis

This is where data can be viewed and analyzed in cube views and dashboards. If there is a grid in any dashboard or cube view, these are available for further drill down or annotation by right clicking the cell. See Using OnePlace Cube Views and Using OnePlace Dashboards for more details.

To view the Cube Calculation Status, click on Cube Views| Calculation Status to show a data grid presenting the Calculation Status of the current active Workflow POV. This will be available at the total Monthly (not an Origin process) Review, Process, or Certify tasks. See Calculation Status in About the Financial Model for more details on this feature.

Intercompany Matching

IC Matching which will show any Intercompany discrepancies. If the button is red in the status column, click the item to see the details. Each Intercompany counterparty will be visible along with counterparties with an Intercompany variance. By clicking on the counterparty, details including the Reporting Currency, Entity Currency, and Partner Currency will be visible.

Select the Intercompany Partner to see the status of the Intercompany issue and any annotations the partner may have made. Select the Difference row to see both parties' statuses and annotations.

A red triangle indicator displays in the cell when data is dynamically derived. This generally occurs when the intercompany matching report performs currency translations.

Right click on the Entity row to perform the following functions:

Set IC Transaction Status

This allows the user to update the status by selecting Not Started, Loaded, Adjusting, Disputed, Finalized. The user may also include comments for the counterparty to see.

Show Partner Workflow Status

This allows the user to see the Workflow status of the counterparty.

Show/Hide Dimension Details

This allows the user to see the Dimension details for the Intercompany accounts.

Drill Down

This will allow the user to drill down on the Dimensions in order to get more information about the Intercompany data.

See Drill Down in Using OnePlace Cube Views for more details on this feature.

See Intercompany Eliminations in About the Financial Model for more details on Intercompany.

Create a New Journal

There are three ways to create a journal as shown in the image below.

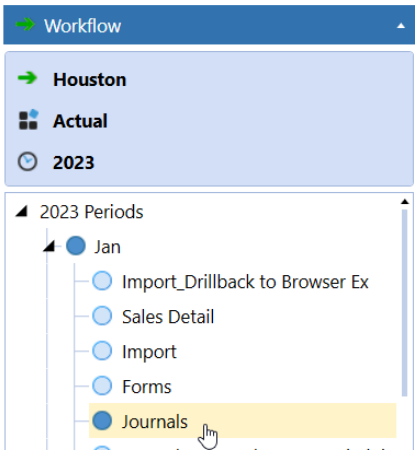
1. **Create Journal**: Create a new blank journal.
2. **Create Journal Using Selected Template**: Create a new journal based on the selected template.
3. **Create Journal Using File**: Create a new journal using data from either an Excel file or a CSV file.

In the following sections, you will create journals using the three methods listed above.

Create Journal

To display the Journals toolbar and create a new blank journal:

1. In the OneStream app, click the **OnePlace** tab.
2. Click **Workflow** to expand.
3. Click the existing **Workflow Profile** to change it if necessary.
    a. In the **Select Workflow View** dialog, click the **Workflow Profile** tab and make a selection.
    b. Click the **Scenario** tab and make a selection.
    c. Click the **Year** tab and make a selection.
    d. Click **OK**.
4. Expand the period tree, locate the time period, and click **Journals**.



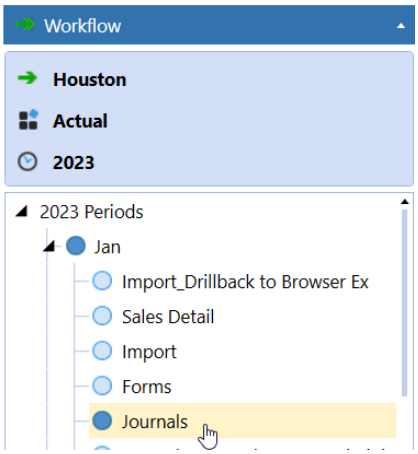
5. On the Journals toolbar, click **Create Journal**.
6. In the Journal Properties editor, under **General**, type a name for the new journal.
7. Under **Journal**, set properties as needed.
8. Under **Point of View**, set properties as needed.

9. Click **Save**. The new journal is added to the **Journals** pane to the right of the journal properties section.

Create Journal Using Selected Template

Templates can be built for journals to accelerate repetitive journal creation. They allow for control over which properties are configured by the user. They can be used to create new journals.

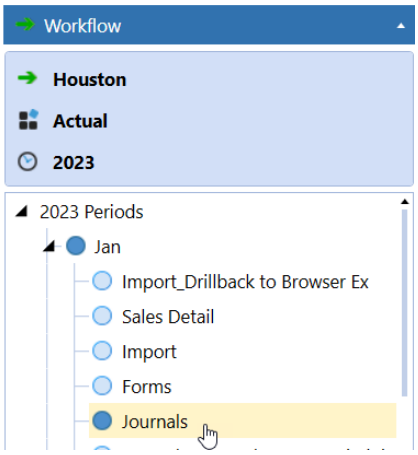
1. In the OneStream app, click the **OnePlace** tab.
2. Click **Workflow** to expand.
3. Click the existing **Workflow Profile** to change it if necessary.
    a. In the **Select Workflow View** dialog, click the **Workflow Profile** tab and make a selection.
    b. Click the **Scenario** tab and make a selection.
    c. Click the **Year** tab and make a selection.
    d. Click **OK**.
4. Expand the period tree, locate the time period, and click **Journals**.



5. In the **Workflow Journal Templates** pane, click a journal template to select. This will be the journal on which you will base the template.
6. On the Journals toolbar, click **Create Journal Using Selected Template**. It will automatically be given a name but you can change this if necessary.
7. Under **Journal Template**, set properties as needed.
8. Under **Journal**, set properties as needed.
9. Under **Point of View**, set properties as needed.
10. Click **Save**. The new journal is added to the **Journals** pane to the right of the journal properties section.

Create Journal Using File

1. In the OneStream app, click the **OnePlace** tab.
2. Click **Workflow** to expand.
3. Click the existing **Workflow Profile** to change it if necessary.
    a. In the **Select Workflow View** dialog, click the **Workflow Profile** tab and make a selection.
    b. Click the **Scenario** tab and make a selection.
    c. Click the **Year** tab and make a selection.
    d. Click **OK**.
4. Expand the period tree, locate the time period, and click **Journals**.



5. On the Journals toolbar, click **Create Journal Using Excel (xlsx) or Comma Separated Values (csv) File**.
6. In the **Open** dialog, browse to the file and click **Open**.
7. Under **Journal Template**, set properties as needed.
8. Under **Journal**, set properties as needed.
9. Under **Point of View**, set properties as needed.
10. Click **Save**. The new journal is added to the **Journals** pane to the right of the journal properties section.

Open a Journal

You can open any journals you have created from the **Journals** pane in the OneStream app.

1. In the OneStream app, click the **OnePlace** tab.
2. Click **Workflow** to expand.
3. Click the existing **Workflow Profile** to change it if necessary.

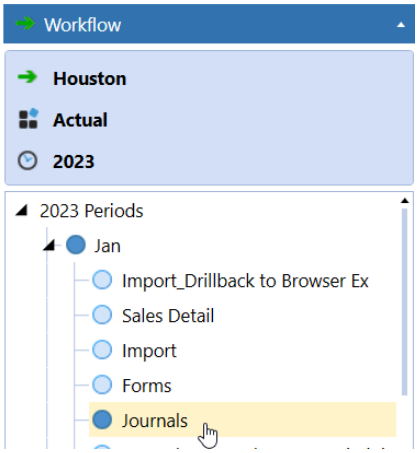
a. In the **Select Workflow View** dialog, click the **Workflow Profile** tab and make a selection.

b. Click the **Scenario** tab and make a selection.

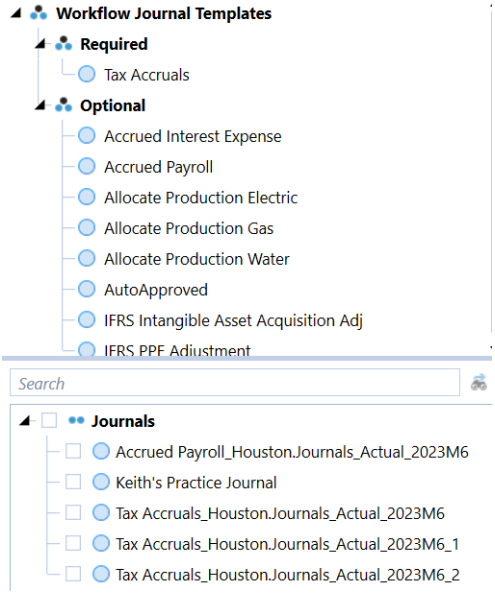
c. Click the **Year** tab and make a selection.

d. Click **OK**.

4. Expand the period tree, locate the time period, and click **Journals**.



The **Workflow Journal Templates** pane and the **Journals** pane display.



5. In the **Journals** pane, click the journal you would like to view. You can new set journal properties as needed and enter line items into the journal.

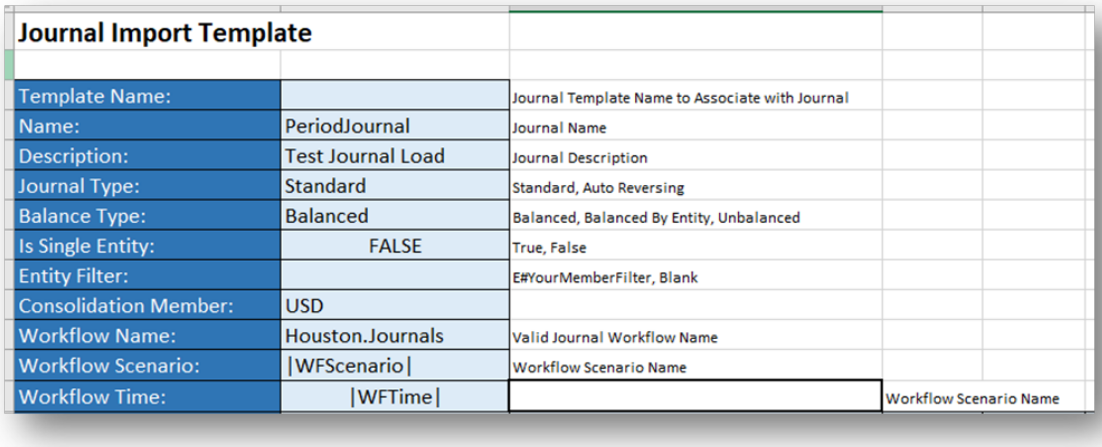
Loading Journal Data

Journal Excel Template

When loading journal entries via Excel a specific template must be created to determine the journal properties, the Dimensions to which the entries are being made, and the debit and credit amounts.

OneStream reads this template using a specific Named Range which is explained later in this section. Ensure the following information is included in the Named Range.

Property Tokens
The first eleven rows of the Named Range in the Excel template must include the following token definitions:



Journal Import Template			
Template Name:		Journal Template Name to Associate with Journal	
Name:	PeriodJournal	Journal Name	
Description:	Test Journal Load	Journal Description	
Journal Type:	Standard	Standard, Auto Reversing	
Balance Type:	Balanced	Balanced, Balanced By Entity, Unbalanced	
Is Single Entity:	FALSE	True, False	
Entity Filter:		E#YourMemberFilter, Blank	
Consolidation Member:	USD		
Workflow Name:	Houston.Journals	Valid Journal Workflow Name	
Workflow Scenario:	|WFScenario|	Workflow Scenario Name	
Workflow Time:	|WFTime|		Workflow Scenario Name

Template Name

Enter a template name is applicable. If this is a free form journal, leave this blank.

Name

Enter the name of the journal.

Description

If desired, enter a description for the journal.

Journal Type*

Standard or Auto-reversing. Allocation is not supported for Excel or CSV import.

Balance Type*

Balanced, Balanced by Entity, or Unbalanced

Is Single Entity*

True or False

Entity Filter*

Use a Member Filter to specify the Entities used with this journal.

Consolidation Member

Enter the specific currency or Local Member of Consolidation.

Workflow Name

Enter the Journal Workflow name. For example, if the name of the Workflow Profile is Houston, and the Adj input type is named Journals, enter Houston.Journals.

Workflow Scenario

Enter the Workflow Scenario or make the template dynamic by entering the |WFScenario| Substitution Variable.

Workflow Time

Enter the Workflow Time or make the template dynamic by entering the |WFTime| Substitution Variable. Workflow Time support two fields. The available cell immediately to the right is option representing the CubeTimeName. This field can be used when the Scenario's Workflow Tracking Frequency is Yearly, and the Input Frequency is Monthly. For example, the Workflow Time would be |WFTime| or 2019 and the CubeTimeName would be the period to post, 2019M7.

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	|WFTime|	

When Tracking Frequency is Yearly and Input Frequency is Monthly:

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	2019M7	

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	|WFTime|	2019M7

Workflow Scenario:	|WFScenario|	Workflow Scenario Name
Workflow Time:	|WFTime|	|WFYear|M7

*See Journal Templates Data Collection for details on these Journal properties.

Dimension Tokens

Next, create the Dimension Tokens necessary to load the journal to the correct Dimensions in OneStream. The Dimension tokens need to be the column header for each data row. The standard tokens used determine the Cube, Entity, Parent, Account, Flow, Intercompany, the User Defined Members, and a Label if needed. Refer to Loading Stage Data for the syntax. The journal specific tokens are as follows:

AMTDR#

This indicates the debited amount.

AMTCR#

This indicates the credited amount.

Header Abbreviations

Static Value

Use :[] to fix a specific Member to the entire column creating a Static Value for the specified Source Dimension. For example, F#:[None] imports the None Flow Member for every Flow row within the Named Range. This syntax applies to all Dimension Tokens.

After the Dimension Tokens are setup, enter the data in the corresponding column.

Template Example:

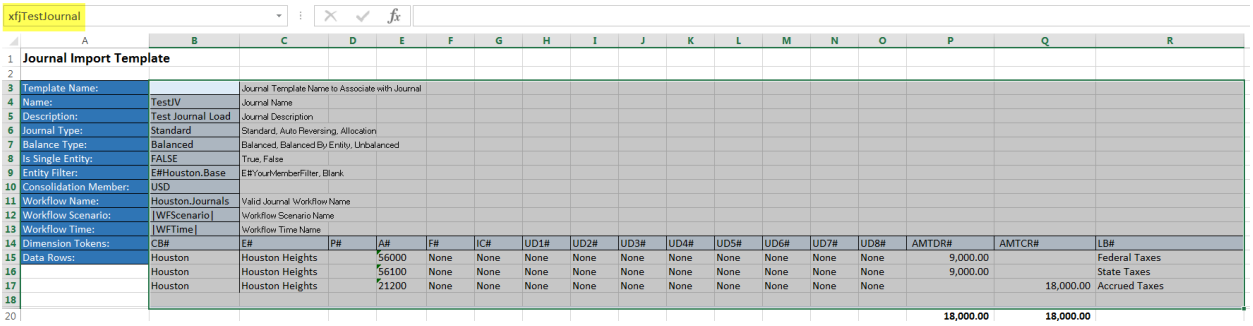
Journal Import Template

Template Name:		Journal Template Name to Associate with Journal															
Name:	TestJV	Journal Name															
Description:	Test Journal Load	Journal Description															
Journal Type:	Standard	Standard, Auto Reversing, Allocation															
Balance Type:	Balanced	Balanced, Balanced By Entity, Unbalanced															
Is Single Entity:	FALSE	True, False															
Entity Filter:	E#Houston.Base	E#YourMemberFilter, Blank															
Consolidation Member:	USD																
Workflow Name:	Houston.Journals	Valid Journal Workflow Name															
Workflow Scenario:	|WFScenario|	Workflow Scenario Name															
Workflow Time:	|WFTime|	Workflow Time Name															
Dimension Tokens:	CB#	E#	P#	A#	F#	IC#	UD1#	UD2#	UD3#	UD4#	UD5#	UD6#	UD7#	UD8#	AMTDR#	AMTCR#	LB#
Data Rows:	Houston	Houston Heights		56000	None	None	None	None	None	None	None	None	None	None	9,000.00		Federal Taxes
	Houston	Houston Heights		56100	None	None	None	None	None	None	None	None	None	None	9,000.00		State Taxes
	Houston	Houston Heights		21200	None	None	None	None	None	None	None	None	None	None		18,000.00	Accrued Taxes
															18,000.00	**18,000.00**	

The final step is to create a XFJ Named Range making sure to include the definition of each property, the Dimension tokens, and the data rows. The Named Range must begin with XFJ for OneStream to read and load it correctly. Multiple XFJ Named Ranges can be used within the template over multiple tabs.

> **NOTE:** Loading of Journal Templates or previously exported Journal data only requires a Parent (P#) column value to be populated if the target Consolidation dimension member being updated is OwnerPreAdj or OwnerPostAdj. Otherwise, this entry can be left blank.

Named Range Example:



xfjTestJournal

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	**Journal Import Template**																	
2																		
3	Template Name:		Journal Template Name to Associate with Journal															
4	Name:	TestJV	Journal Name															
5	Description:	Test Journal Load	Journal Description															
6	Journal Type:	Standard	Standard, Auto Reversing, Allocation															
7	Balance Type:	Balanced	Balanced, Balanced By Entity, Unbalanced															
8	Is Single Entity:	FALSE	True, False															
9	Entity Filter:	E#Houston.Base	E#YourMemberFilter, Blank															
10	Consolidation Member:	USD																
11	Workflow Name:	Houston.Journals	Valid Journal Workflow Name															
12	Workflow Scenario:	|WFScenario|	Workflow Scenario Name															
13	Workflow Time:	|WFTime|	Workflow Time Name															
14	Dimension Tokens:	CB#	E#	P#	A#	F#	IC#	UD1#	UD2#	UD3#	UD4#	UD5#	UD6#	UD7#	UD8#	AMTDR#	AMTCR#	LB#
15	Data Rows:	Houston	Houston Heights		56000	None	None	None	None	None	None	None	None	None	None	9,000.00		Federal Taxes
16		Houston	Houston Heights		56100	None	None	None	None	None	None	None	None	None	None	9,000.00		State Taxes
17		Houston	Houston Heights		21200	None	None	None	None	None	None	None	None	None	None		18,000.00	Accrued Taxes
18																		
20																18,000.00	18,000.00	

> **NOTE:** The Named Range only covers each property definition.

Journal CSV Template

To set up a CSV template for a Journal, the Header and Detail values must be specified.

	A	B	C	D	E	F
2	!RowType (H=Header)	JournalName	OriginatingTemplateName	JournalDescription	JournalType	JournalBalanceType
3	!RowType (D=Detail)	JournalName	CubeName	EntityName	ParentName	AccountName
	H	Test Tax Accruals_Houston.Journals_Actual_2011M3	Tax Accruals	Test Journal Load	Standard	Balanced
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3	Houston	Houston Heights		56000
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3	Houston	Houston Heights		56100
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3	Houston	Houston Heights		21200
4	H	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Tax Accruals		Standard	Balanced
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	Houston Heights		56000
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	Houston Heights		56100
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	Houston Heights		21200
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	South Houston		56000
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	South Houston		56100
	D	Test Tax Accruals_Houston.Journals_Actual_2011M3_1	Houston	South Houston		21200

1. Column A Specifies Row Type

   In the first two rows of Column A, create two Row Type Parameters specifying the Header and the Detail. In the example above, !RowType (H=Header) and !RowType (D=Detail) are used to tag the corresponding rows with H or D identifying where the Header and Detail information is located in the CSV file.

2. Row One Specifies the Headers

   After the Header Parameter is configured, enter the Journal Column Headers. The required Journal Headers are JournalName, OrigniatingTemplateName, JournalDescription,JournalType, JournalBalanceType, IsSingleEntity, EntityMemberFilter, ConsName, WFProfileName, WFScenarioName, WFTimeName and CubeTimeName. For more details on these, refer to the Journal Excel Template.

3. Row Two Specifies the Details

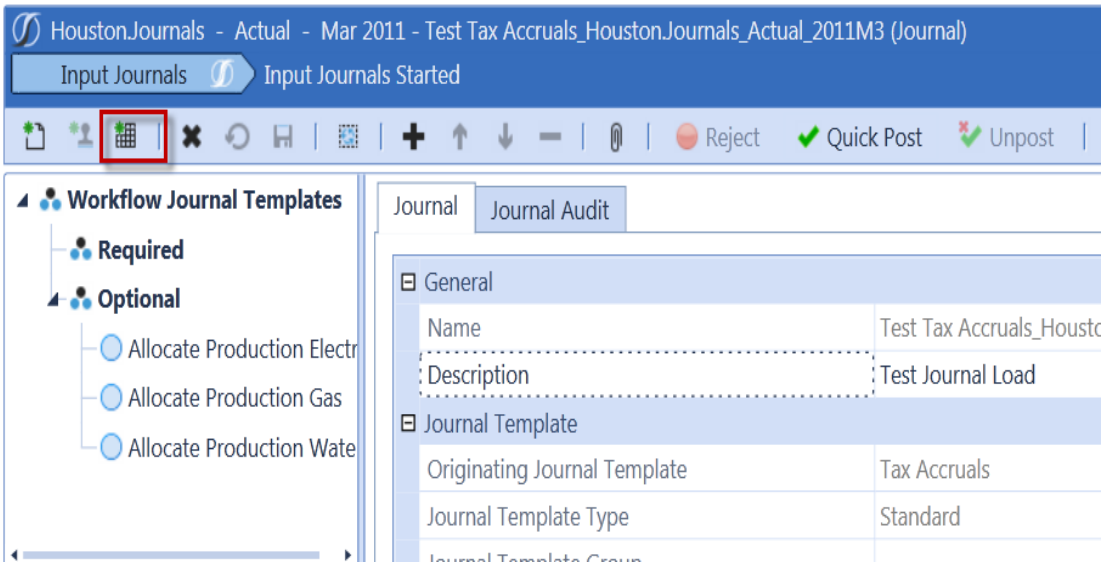
   After the Detail Parameter is configured, enter the Journal Detail Headers. The required Journal Detail Headers are JournalName, CubeName, EntityName, ParentName, AccountName, FlowName, ICName, all UDNames, DebitAmount, CreditAmount, and LineDescription. For more details on these, refer to Journal Excel Template.

4. Header and Detail Tags

   The Journal data is driven by how each row in column A is tagged. Any rows tagged with H load as the Headers and any row tagged with D load as the details.

Loading Journal Details via Workflow

After the template is configured, users can load it directly into OneStream during the Workflow process. While the Journal Input type is selected, click the icon in the Journal toolbar. This lets you select the desired Excel or CSV template and load it into OneStream. After the file is loaded, the journal line items will appear in the journal and the user can save it to the Cube.



Extracting/Loading Journal Details via BRApi

Users can export journal details into a CSV or XSLX file and load journals from a CSV or XSLX file by configuring an Extensibility Business Rule. This allows journal details to be extracted from one application and loaded into another. Users can also extract the journal, make changes, and re-load.

Extract Journals to CSV

Use BRApi.Journals.Data.ExportJournalstoCSV and define the session, filepath, Workflow Profile, Scenario, Time Filter, and Journal Status.

BRApi.Journals.Data.ExportJournalsToCsv(si, filePath, "Houston", "Actual", "T#|WFYear|.Base", "Posted")

Extract Journals to XSLX

Export journals associated with defined parameters to an XSLX file using the ExportJournalsToXSLX BRApi. It is sometimes easier to read and modify journal data in an XSLX format as compared to CSV files that are generated by the ExportJournalsToCSV BRApi. Use the parameters listed in the following table.

Parameter	Description
si	Session information
serverFilePath	Name and location to save the file
wfProfileName	Name of the Workflow Profile
wfScenarioName	Name of the Scenario

Parameter	Description
wfTimeMemberFilter	Filters workflow profile times. For example, if the scenario's Workflow Tracking Frequency is set to Quarterly, and Input Frequency is set to Monthly, the filter searches for journals associated with the Workflow Tracking Frequency. Time must be prefixed with T#. You can type a comma separated list of times such as T#2022H1, T#2022H2.
journalName	Filters journals that contain the specified text
journalStatus	Status of the journal. You can search by status or use All to return all statuses. You cannot type a comma separated list.

The following code example uses the ExportJournalsToXslx BRApi:

```
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.Extender.ExportJournalstoXSLX
    Public Class MainClass
        Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Object, ByVal args As ExtenderArgs) As Object
            Try
                Dim configSettings As AppServerConfigSettings = AppServerConfig.GetSettings(si)
                Dim sourcefolder As String = BRApi.Utilities.GetFileShareFolder(si,FileShareFolderTypes.FileShareRoot,Nothing) & "\System\Contents\Export"
                Dim journalFilePath As String = sourcefolder & "\" & "Journal_Export.xlsx"
                Dim filePath As String = BRApi.Journals.Data.ExportJournalsToXlsx(si, journalFilePath, "Houston", "Actual", "T#2023M5", "", "All")

                Return Nothing
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
    End Class
End Namespace
```

Load Example

Use BRApi.Journals.Data.ImportAndProcessJournals and define the session, filepath, and journal tasks to complete upon loading the journal details.

'BRApi.Journals.Data.ImportAndProcessJournals(si, filePath, save, submit, approve, post, unpostAndOverwrite, throwOnError)

Delete a Journal

You can delete journals that you no longer use or need.

1. Navigate to the journal you would like to delete:
    a. In the OneStream app, click the **OnePlace** tab.
    b. Click **Workflow** to expand.
    c. Click the existing **Workflow Profile** to change it if necessary.

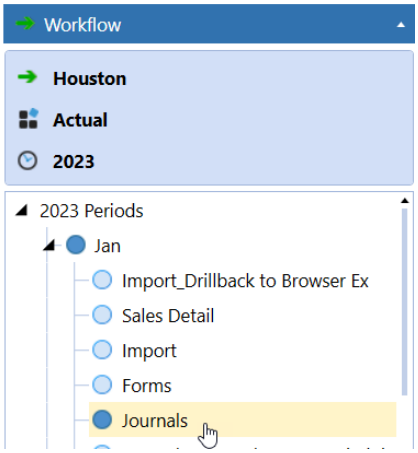
i. In the **Select Workflow View** dialog, click the **Workflow Profile** tab and make a selection.

ii. Click the **Scenario** tab and make a selection.

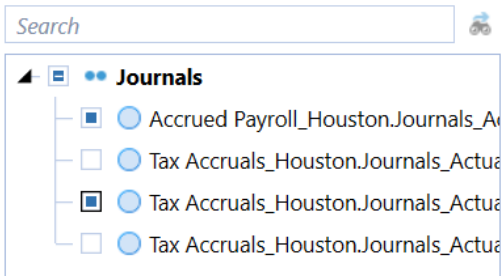
iii. Click the **Year** tab and make a selection.

iv. Click **OK**.

d. Expand the period tree, locate the time period, and click **Journals**.



2. In the **Journals** pane, place a tick into the box next to the journal you want to delete. You can delete more than one journal at a time by ticking all of the necessary boxes.



3. Click **Delete Selected Item**.

4. In the confirmation dialog, click **Yes**.

Process a Journal

After you have created or opened a journal and entered data into the ledger, there are several actions you can perform on the journal.

1. **Validate a journal**: Ensure that data is correct and that cube intersections are valid.
2. **Reject a journal**: Discard the journal data.
3. **Quick post a journal**: Apply the journal data to the associated cube.
4. **Unpost**: Remove the journal data from the associated cube.

In the following sections, you will move a journal through the above processes.

Validate a Journal

You can ensure that dimension members selected in the journal are valid cube intersections based on constraints configured in the dimension library. If validation errors exist, they will display in a Validation pane so that you can view them and make edits to the journal. This helps you save time by validating entries earlier in the process.

> **NOTE:** You must have the Journal Processing security role to use this feature and dimension member constraints must first be configured in the dimension library.

To validate a journal:

1. Open the journal that you would like to validate.
2. Click **Validate**.

Journal data will be evaluated and if invalid data is found, the **Validation** pane will display a message that details each entry error.

Reject a Journal

You can reject a journal after it has been submitted for approval.

To reject a journal:

1. Open the journal that you would like to reject.
2. Click **Reject**.

The journal status changes to Rejected.

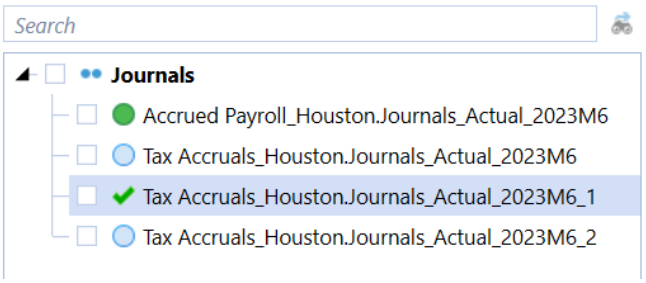
Post a Journal

Posting a journal in the OneStream app means adding the ledger data into the associated cube, similar to moving transaction data into a general ledger.

To post journal data:

1. Open the journal that you would like to post.
2. Enter line item journal data as necessary.
3. Optional. If you would like to post more than one journal at a time, select the journals in the **Journals** pane by placing a tick in the selection box next to the journals.
4. Click **Quick Post**. A green checkmark next to the journal in the **Journals** pane indicates the

journal has been posted.



Journal data is moved into the associated cube.

Unpost a Journal

Unposting a journal in the OneStream app means removing the ledger data from the associated cube.

To unpost journal data:

1. Open the journal that you would like to unpost.
2. Optional. If you would like to unpost more than one journal at a time, select the journals in the **Journals** pane by placing a tick in the selection box next to the journals.
3. Click **Unpost**. The green checkmark next to the journal in the **Journals** pane is removed, indicating the journal has been unposted.

Journal data is removed from the associated cube.

Journal Templates

Create pre-set Journal Templates in Journal Template Groups. After the Journal Template Groups are created, they are organized into Journal Template Profiles and then assigned to Workflow Profiles. For more details on Profiles, see Form Template Profiles.

Journal details can also be loaded via an Excel or CSV template. See "Loading Journal Data" in Data Collection Guides for more information.

Journal Template Properties

This section includes various journal template properties.

General

Name
Name of the journal template.

Description
A detailed description of the journal template.

Journal Template

Journal Template Type
A journal template can either be standard or auto approved. If auto approved, a user can create a journal from the template with limited editing ability meaning permission to change name, description, and so forth. After pressing Save, the line items are validated, and the journal skips the approval process and becomes approved.

If the journal needs to be edited, it is handled like any other journal from that point forward. Someone in the Approve Journals group must reject it for the original user to edit it. Then, the regular Submit, Approve steps will occur. This approach is needed because the person who created the auto approved template only pre-approved certain numbers to be put in the system.

The user does not need to be in the ApproveJournals security group to do this. They only need to be in the ProcessJournals security group which is normally the group for end users to save or submit journals.

Journal Template Group
This displays the journal template group that was created under the new template.

Journal Requirement Level
This setting works together with Frequency and determines when a journal is required. The options available are Not Used (anymore), Optional, or Required.

Journal Frequency

All Time Periods
This allows the journal to display every period.

Monthly
This allows the journal to display every month. If this is for a weekly application, it will display the last week of each month.

Quarterly
This allows the journal to display every quarter, or four times a year.

Half Yearly
This allows the journal to display two times a year; once in June and December.

Yearly
This allows the journal to display once a year in December.

Member Filter
This turns on the Frequency Member filter and then the filters can be defined in that section. For example, if this journal only needs to be completed in September, use the filter T#WFYearM9.

Frequency Member Filter
This becomes available when Member Filter is chosen in the Forms Frequency options above, otherwise this is gray for all the others. The purpose of this option is to allow the ability to filter by time.

The following settings control what the user can modify when creating a journal. Settings are True or False.

Can Change Journal Settings
Can Change Journal POV
Can change Journal Line Items
Can change Journal Amounts

Journal

Journal Type
The options are Standard, Auto Reversing, or Allocation. When posting an Auto Reversing journal, the auto reversal journal is automatically created in the next time period and set to the Approved state. The Auto Reversal journal has all the debits and credits reversed.

When un-posting an Auto Reversing journal, check to make sure the auto reversal is not posted first. Ifnot, delete the Auto Reversal journal from the next time period and un-post.

An Allocation journal can be set up to perform simple or more intricate allocations, such as creating the weighting of the allocation, previewing the actual allocation entries, and un-posting them if they need to be run again.

Journal Balance Type

Balanced
The entire set of journal lines must balance between the debits and credits.

Balance by Entity
Debits and credits in a multi-Entity journal must balance for each entity.

Unbalanced
Balance check will not be performed. This is normally used for one-sided journals.

Is Single Entity
If True, the entity name is entered in the Journal POV and all journal lines relate to this one entity. If False, the Cube, Entity and Parent columns must be filled out for every line in the journal instance.

Entity Member Filter
This Member Filter will help limit the list of entities presented to the user in the Journal POV and journal lines.

Point of View
In order to limit the amount of setup for every journal line, the items that remain constant (e.g. Flow = None) can be set in the Journal POV instead of in every line Item.

> **TIP:** A Journal template can be repeated on a regular basis if values are placed in the journal lines and journal's settings require repeating upon a certain frequency.

Journal Template Group Properties

This section lists the general group properties.

General (Journal Template Group)

Name
The name of the Journal Template group.

Description
A short description of the Template group such as how or where it is used.

Security

Access Group
Members of this group have access to the Journal Template group

Maintenance Group

Members of this group have the authority to maintain the Journal Template group

> **NOTE:** Click to use the Security page to modify users and groups before assigning them. Click and begin typing the name of the group. As the first few letters are typed, groups are filtered making it easier to find and select the desired group. Select the group, press CTRL and then double-click.

Presenting Data With Books, Cube Views and Dashboards

You can present data using report books, cube views, and dashboards. Report Books allow you to combine a variety of reports and files in a way that fits your reporting needs. Cube views allow you to query cube data and present it in a variety of ways. Dashboards present data by combining a variety of sources such as components, data adapters, and other files. in this section you will learn how to present data using these methods.

Presenting Data Using Report Books

Report Books let you combine a variety of reports and files into a single document. They are commonly used to create financial statements and management report packages. Anyone who needs a snapshot view of their company financials can run report books, but this is done most often by managers and individuals who report to management.

There are pre-built reports that can be customized to fit your reporting needs, such as:

- Cube views (which can be exported to an Excel file)
- Dashboard reports
- Charts
- Extensible documents

After creating a report book, it can be viewed and used in different contexts. For example, they can be added to other books, added to dashboards using either the Book Viewer or File Viewer dashboard component, or emailed via the OneStream Parcel Service OneStream Solution. You can also generate and view report books by running a data management sequence.

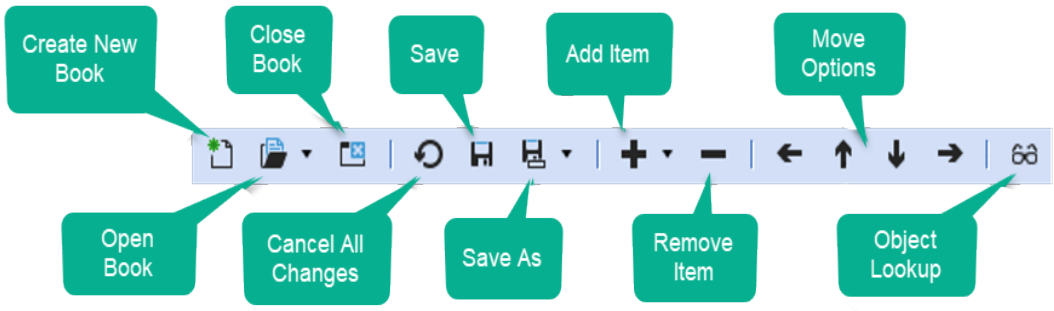
Books can be stored in locations such as a fileshare, your computer desktop, or a dashboard file. When saving a report book it is important to follow pre-defined naming conventions depending on the format of the book, such as bookname.xfDoc.pdfBook (PDF books), bookname.xfDoc.zipBook (zipped books), and bookname.xfDoc.xlBook (Excel books).

> **NOTE:** Embedding fonts in a PDF Report book could increase the size of the PDF file. Use the PDF Embedded Fonts to Remove property in the Application Server Configuration File to specify which fonts should not be embedded. This will reduce the size of PDF files and control the resolution during report book PDF generation. The default font-stack setting is: Arial; Calibri; Segoe UI; Tahoma; Times New Roman; Verdana.

Book Designer Toolbar

When designing a book, you will work in the two tabs of the book designer to build and preview your book. The book designer is where you use the book toolbar icon to create a book in a hierarchical format. The OneStream Book Designer includes tools and methods that help you customize data when creating reports.

The following image displays the Book Designer toolbar.



The names of the tools describe their functions, but there are options to choose from when you click **Save As** and **Add Item**.

Save As lets you specify the kind of book you are creating and the location.

- **Save as File in OneStream File System**: Saves the book in the File Explorer. It can be saved in a public folder or a local folder on your system.
- **Save as File on Local Folder**: Saves the book in a local folder on your system.
- **Save as Application Workspace File**: Saves the book in the File section of an Application Workspace Maintenance Unit.
- **Save as System Workspace File**: Saves the book in the File section of a System Workspace Maintenance Unit.

You can create three types of books: Excel books, PDF books, and Zip File books. Excel books produce an Excel Workbook with each piece of content on a new tab. PDF books compile all content in a single PDF file. Zip books produce a zip file where each piece of content exists individually. You can create each type of book by saving it with the appropriate file suffix.

You can specify the book type when you save the file using any of the following extensions:

- Excel books: ReportBookName.xfDoc.**xlBook**
- PDF books: ReportBookName.xfDoc.**pdfBook**
- Zip File books: ReportBookName.xfDoc.**zipBook**

Add Item lets you specify the kind of item to add to the book. You can add:

- Files
- Excel Export Items
- Reports
- Loops
- Condition statements (If, Else If, and Else)
- Item to Change Parameters

Book Properties

When you are creating a new book, regardless of type, there are two book properties to consider:

- Determine Parameters from Content
- Required Input Parameters

Parameters let you determine what data should be displayed in the resulting report book. They act like filters in that only certain data is selected and displayed in the report book. Each of these items is explained in further detail below.

Determine Parameters from Content

This item can be set to True or False. If set to True, it automatically determines the required input parameters from the book's content items. For large report books, setting this to False and manually specifying the required parameters will result in a performance gain.

Required Input Parameters

This field requires a comma-separated list of parameter names. You can leave this field blank if Determine Parameters from Content is set to True.

Using an Item to Change Parameters

Change parameters are used to turn off the prompt that the parameter would normally generate to define the desired result of the book. The Loop definition populates the items instead.

For example, imagine a cube view that would normally prompt you for the time member to be added to the report book. In this case, you might not want the user to make a selection, and instead, you want the report book to run for the workflow time. Rather than copying or altering the cube view, you would add an Item to Change Parameters.

File Type

Part of the process of creating new report books is selecting a file source type. You select a file source type whenever a new file is added to a book.

You can select the following source types for your files:

- **URL**: Specify a file from an internal or external web page. You must use the full URL.
- **Application Workspace File**: Select a file stored in the Application Workspace Maintenance Unit File Section.
- **System Workspace File**: Select a file stored in the System Workspace Maintenance Unit File Section.
- **Application Database File**: Select a file stored in the Application Database Share.
- **System Database File**: Select a file stored in the System Database Share.

- **File Share File**: Select a file from the File Share. Many file types can be added such as Word, PDF, CSV, and XLSX.

In addition to specifying the file source type, you must also set the following:

- **URL or Full File Name**: Select the URL or name of the file you want to use. Type the full URL or click the ellipsis to browse to a file.
- **Output Name**: Type a name for the file when creating a Zip book. If the .pdf extension is included for a Word or Excel file, the file will be converted into a PDF. This property is optional and does not apply to PDF books.

Excel Export Item

When creating an XLbook, you will use Excel Export Items. You can specify a cube view to export to an Excel file. Each cube view will display on a separate worksheet in the book. These are useful when added to XLBooks rather than PDFBooks.

> **NOTE:** Report and File items are not supported by Excel books and are ignored if added.

To create an Excel Export Item, set the following properties:

- **Cube View Name**: Click the ellipsis and select a cube view to add to the Excel Book.
- **Output Name**: This field is optional. Type a name for the cube view to display on the Excel worksheet. The character limit is 31.

Report

When you add a report to a report book, you must specify a report type. This usually depends on how you have saved the book.

You can select from the following report types:

- **Cube View**: The report is based on a cube view.
- **Dashboard Chart**: The report is based on an Application Dashboard Chart.
- **Dashboard Report**: The report is based on an Application Dashboard Report.
- **System Dashboard Chart**: The report is based on a System Dashboard Chart.
- **System Dashboard Report**: The report is based on a System Dashboard Report.

> **NOTE:** The chart and report items listed above are application dashboard chart components, rather than dashboard components.

In addition to selecting a report type, you must also set the following:

- **Cube View or Component Name**: Name of the Cube View, Dashboard Report or Dashboard Chart that you are using in the book. Click the ellipsis and browse to the source for a report.
- **Output Name**: Type a name for the report when it is a Zip book. This field is optional and does not apply to PDF books.
- **Include Report Margins/Report Header/Page Header/Report Footer/Page Footer**: Select True to keep the original margins, report or page headers, and report or page footers. Select False to remove the original formatting.

Loop

A loop is a sequence of instructions that will continually run a process as many times as is defined in the loop definition. For example, a book can be set up to loop through all the base entities under a particular hierarchy and generate an instance of the same cube view report for each entity.

Loop Type

There are three types of Loops to use when building your book:

Comma Separated List: Select this option to enter values separated by a comma to be referenced later in the book hierarchy.

Dashboard Parameter: Select this option to use a pre-configured Parameter found in the Application Dashboards page.

For example, a ParamSalesRegions parameter returns a list of all Sales Regions within the application resulting in the report book's loop variables using the same list.

Member Filter: Select this option to utilize the member filter builder to build a member filter based on dimension members.

This loop will run each of the report book items for the specified members.

Loop Definition

When using the loop type **Comma Separated List**, enter a comma-separated list of text values to loop over. Use brackets [ ] when names have spaces.

For example:
(Houston, Clubs, [Houston Heights])

When using the loop type **Dashboard Parameter Loop Type Used**, enter the name of a dashboard parameter to create a list based on an existing parameter.

For example, a ParamSalesRegions parameter returns a list of all Sales Regions within the application resulting in the report book's loop variables using the same list.

When using the loop type **Member Filter Loop Type Used**, enter a member filter to supply a list of members to use in the loop.

For example:
E#Frankfurt, E#Houston, E#Montreal
This loops over each entity and performs the process three times.

E#[NA Clubs].Base
This loops over each base entity under NA Clubs and performs the process for however many base entities there are.

Dimension (only available when Member Filter is the Loop Type)
The name of the specific dimension used by the member filter. Click the ellipsis and select the correct dimension.

Loop Variables

Create loop variables by selecting a variable from the drop down menu.

|Loop1-4Variable|
Allows all Report Book items located in the loop's hierarchy to reference the loop definition's values by name.

Up to four loop variables can be referenced and all must be enclosed in |pipes|.

For example
|Loop1Variable|.

Use |Loop2Variable| through |Loop4Variable| to create nested loops within a loop.

|LoopDisplay1-4Variable|
Allows all report book items in the loop hierarchy to reference the loop definition's values by description.

For example
|Loop1Display|

|Use |Loop2Display| through |Loop4Display| to create nested loops within a loop.

|Loop1-4Index|
Assigns a number to the values in the loop definition beginning with number one, which can be referenced in the report book items in the hierarchy.

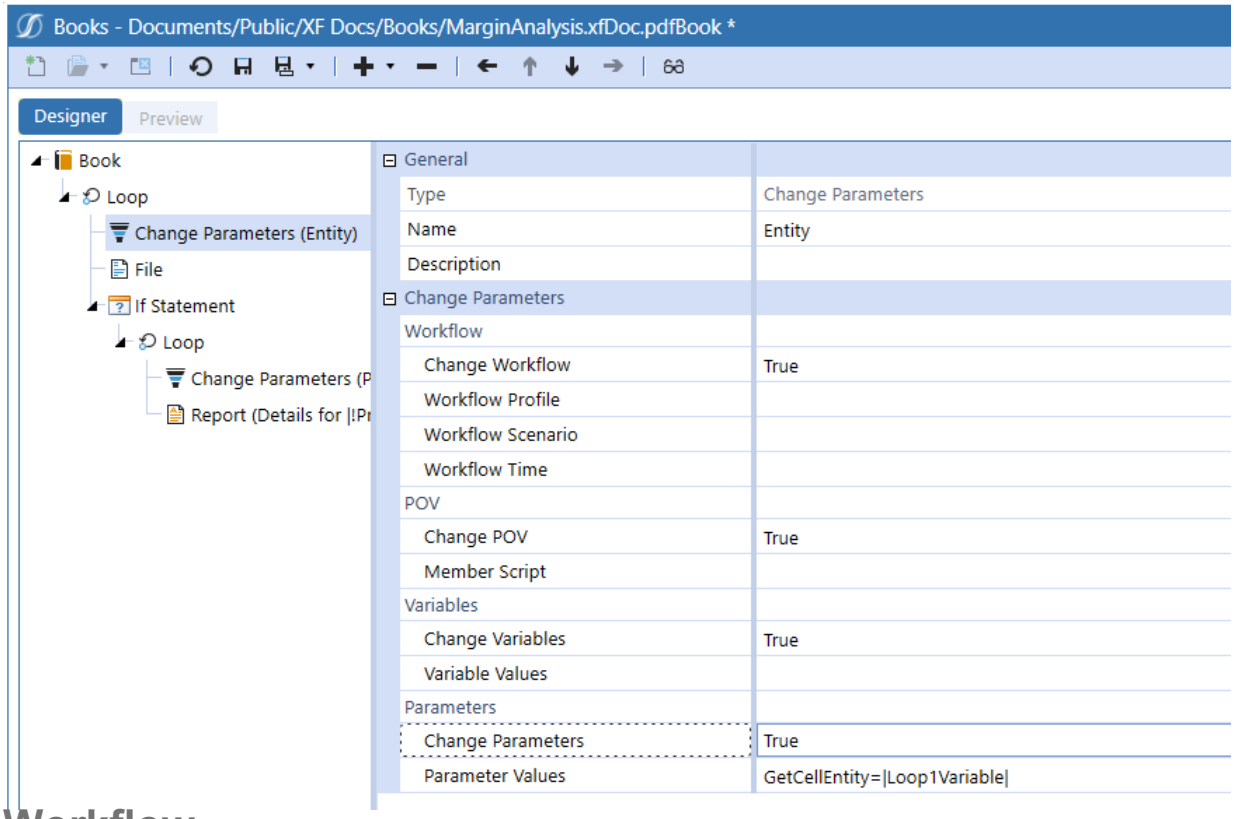
Use |Loop2Index| through |Loop4Index| to create nested loops in a loop.

Change Parameters

Change parameters are used to enhance the data in report books. Setting these parameters customizes the report book output without altering the input from the source.

Loops require Change parameters to run, but Change parameters are not exclusive to loops. When a Change Parameter type is encountered in a loop, the Loop variable is updated to use the next Loop variable.

For example, a loop may be used to loop through a list of entities and run a report for each. The Change parameters book Item should be added here to pass along the appropriate Loop variable (such as |Loop2Variable|) which applies to each entity included in the loop.



Workflow

Change Workflow: Set to True to change settings for cube views or reports driven by Workflow POV information.

Workflow Profile: Specify the Workflow Profile name to replace the original Workflow Profile referenced in the cube view or report.

Workflow Scenario: Specify the Workflow Profile Scenario Type to replace the original Workflow Profile Scenario Type referenced in the cube view or report.

Workflow Time: Specify the Workflow Profile Time to replace the original Workflow Profile Time referenced in the cube view or report.

POV

POV: Set to True to change the book's POV without having to change the actual cube view or report.

> **NOTE:** To use this feature, the POV tab in the cube view should not have any members selected.

Member Script: Click the ellipsis button to launch the Member Filter Builder and enter a Member Script to change the POV. The example below changes the POV for both the entity and account.
E#[|Loop1Variable|]:A#Sales

Variables

Change Variables: This serves as a placeholder that can store up to ten variables. This is valuable when If statements are used.

Variable Values: Enter a comma-separated list of name-value pairs to change the values of the predefined variables named Variable 1-10.

```
Example: Variable1=Red, Variable2=Large, Variable3=[(Loop1Variable)]
```

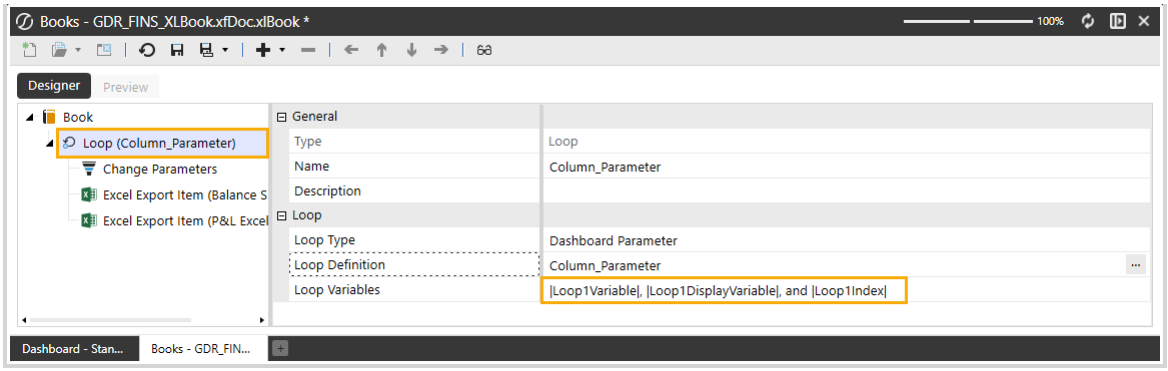
Parameters

Change Parameters: Use to specify a value for a dashboard parameter found in a cube view.

Parameter Values: Click to select a parameter. Enter a comma-separated list of name-value pairs to override the custom parameter's values.

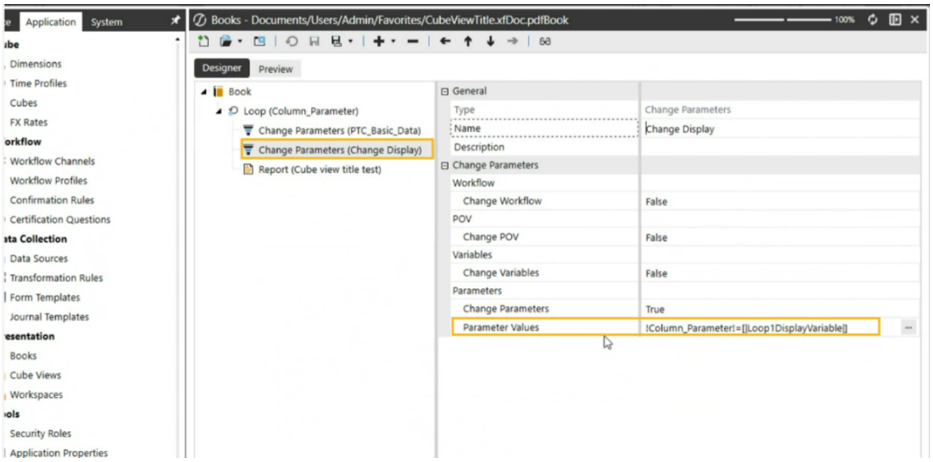
```
Example: MyParam=Red, MyOtherParam=[|Loop1Variable|]
```

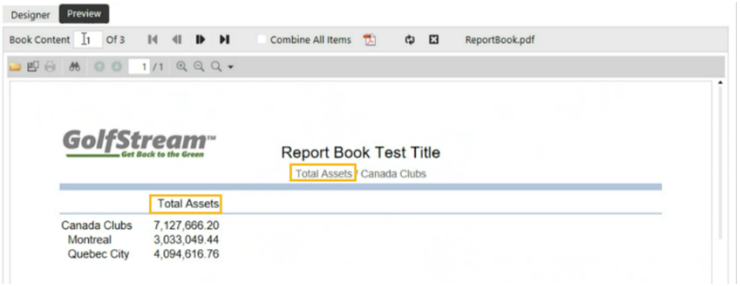
> **NOTE:** Use !! to display the Display Items in a report header. Use ! to display the Value items in a report header.



The Loop definition has a single "Change Parameters" definition. Within that definition, it changes to a single variable, "Loop1Variable". The "Loop1DisplayVariable" and "Loop1Index" variables never change.

To change the Report Book title based on the Cube View column results, add a second "Change Parameters" definition changing the "Loop1DisplayVariable" variable.





If Statement

If Statements can be used to provide conditional logic to your report books. They are commonly used in tandem with loops to create a more fine-tuned output but they can also stand on their own. Additional logic can also be added through the use of Else If and Else statements. If Statements determine how book items within the hierarchy are processed.

Statement: Enter a conditional statement using parameters to determine whether the book items will be processed.

Example of Using Variables in If Statements

```
(|Loop1Variable| = [Frankfurt])
```

This is an example of an If Statement that would have existing with a loop. If “Frankfurt” is found within the Loop Definition for the Loop1Variables, the book items in this hierarchy will process for Frankfurt.

Example of Using Parameters in If Statements

```
(|!UserName!| = Administrator)
```

If the person running the book has a user name of Administrator, the book items in this hierarchy will be included in the book.

Example of Combining Statements

```
(|!UserName!| = Administrator) Or (|!UserName!| = JSmith)
```

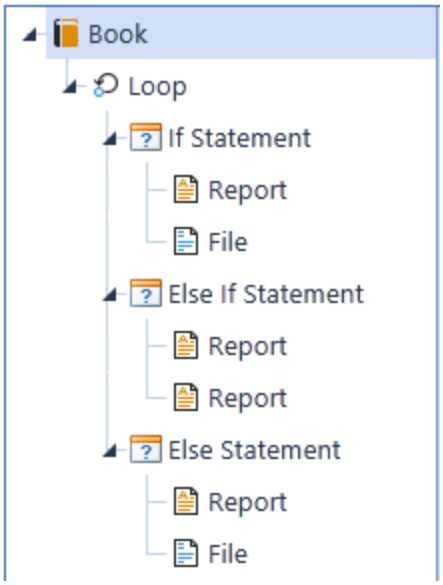
If the person running the book has a user name of Administrator or JSmith, the book items in this hierarchy will be included in the book.

Else/Else If Statement

Else and Else If Statements, like If Statements, may or may not be used within loops to provide additional logic to the report book output. Else and Else If statements determine how book items within the hierarchy are processed. An If statement is needed to use an Else or Else If statement.

Statement: Enter a conditional statement using parameters to determine whether the child items will be processed. See examples in If Statement.

Example:




```
Loop1Variable: E#[Frankfurt], E#[Europe Clubs], E#[Clubs]
```

If Statement: (|Loop1Variable| = [Frankfurt])
If Loop1Variable is Frankfurt, run the report and file under the If statement.

Else If Statement: (|Loop1Variable| = [Europe Clubs])
Else If (or otherwise if) the Loop1Variable is Europe Clubs, run the report under the Else If statement.

Else: Else (in all other cases) run the report under the Else statement.

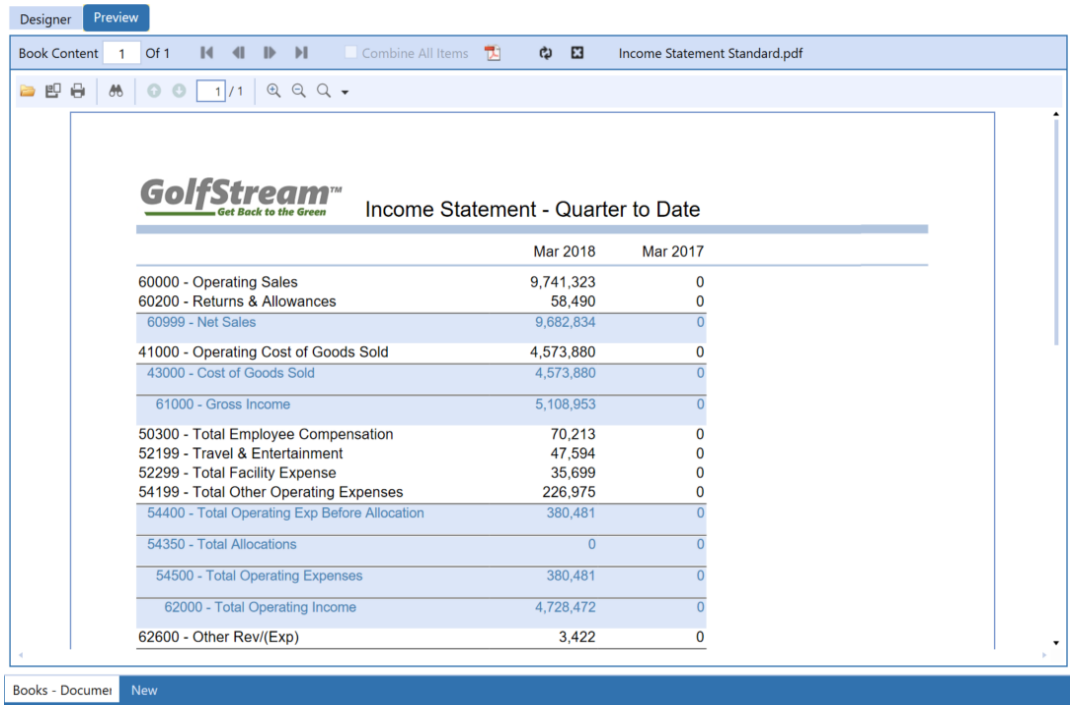
Create a Report Book

This procedure will guide you through the process of creating a basic report book.

> **NOTE:** When creating books, you can drag and drop book items. This makes it easier to move items up and down the heirarchy.

1. Open OneStream.
2. On the **Application** tab, under **Presentation**, click **Books**. The Book Designer opens.
3. Click **Create New Book** on the toolbar.
4. Set the Determine Parameters from Content field to either **True** or **False**. See Book Properties.
5. If you set the field to False in Step 4, type the appropriate parameters into the **Required Input Parameters** field.

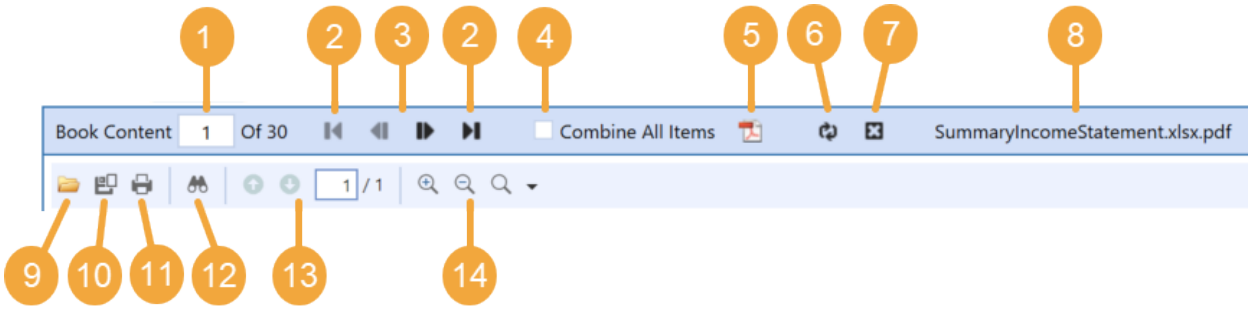
   If you set the field to True, leave the Required Input Parameters field blank.
6. Click **Add Item** on the toolbar and select the items you want to add to the book:
   - **Add File**: See File Type.
   - **Excel Export Item**: See Excel Export Item
   - **Report**: See Report
   - **Loop**: See Loop
   - **Conditional Statement**: See If Statement and Else/Else If Statement.
   - **Add Item to Change Parameters**: See Parameters
7. If you need to remove items from the Book Designer, click **Remove Item** on the toolbar.
8. Click **Save As** and select a location.
9. Click the **Preview** tab to see the new book. A book similar to the following image is displayed.



10. Click **Close Book** on the toolbar when you are done. You can open another book by clicking **Open Book** and selecting the kind of book.

Book Preview Toolbar

The Book preview toolbar allows you to interact and navigate in the book being previewed.

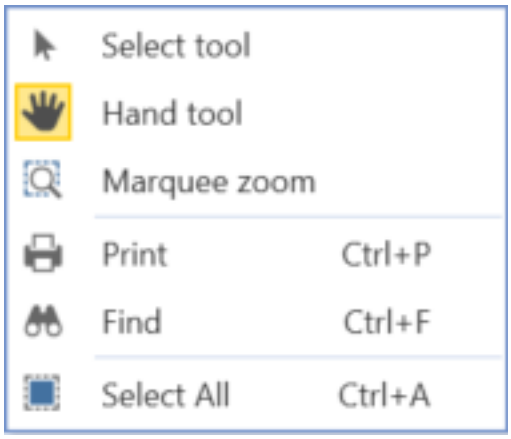


1. **Page Navigation**:This displays what page is currently being previewed in the report book. To navigate to a specific page, enter the page number and press **Enter**.
2. **First/Last**: Use these buttons to navigate to the first or last page of the report book.
3. **Previous/Next**: Use these buttons to navigate forward or backward one page.
4. **Combine All Items**: This combines the report book's pages and treats them as one content item. Use this feature in conjunction with saving or printing an entire report book. If selected, you can save the entire report book as one PDF file, print the entire book, and navigate page by page using the blue navigation arrows. If this box is cleared, only the current page will save or print, and black navigation icons will be enabled.
5. **Download Combined PDF File**: This combines the entire report book into a PDF file using the standard Adobe rendering process and uses less memory when attempting to display large combined PDF files. This does not require enabling Combine All Items. This is the suggested method when downloading any report book to PDF.
6. **Refresh**: Use this button to refresh the report book and select new Parameter values.
7. **Close**: Use this to close the report book in the Preview screen.
8. **File name**: Shows the name of the file currently being previewed.
9. **Open**: Use this to open a report book from a computer desktop or folder.
10. **Save**: Use this to save the current report book page. To save the entire report book as one PDF file, ensure that the Combine All Items checkbox is selected.
11. **Print**: Use this to open the Print dialogue box. To print the entire report book, ensure that the Combine All Items checkbox is selected.
12. **Find**: Use this to find specific keywords in a report book.

13. **Previous Page/Next Page**: Use these icons to navigate pages of a single report book content item or to navigate a report book's pages when the Combine All Items checkbox is selected. If a particular content item, such as a Cube View Report, is more than one page, use these buttons to navigate that report.
14. **Zoom**: Use these buttons to zoom in and out of a report book.

Right-Click Options

Right-click anywhere within the report to select these options.



Select tool: Use this to select portions of the report book. This feature allows you to copy and paste. Highlight a portion of the report book, press **Ctrl+C** then **Ctrl+V**.

Hand tool: Use this button to scroll through a report book by clicking anywhere on the screen and moving the mouse.

Marquee Zoom: Use this tool to select an area of interest on which to zoom.

> **TIP:** Use **Alt+Left** to return to the previous view or **Ctrl+0** to return to a page level view.

Print: Use this to open the Print dialogue box.

Find: Use this to find specific keywords in a report book.

Select All: Use this to select all the content in a report book.

Build Reports Through Cube Views

Cube views submit and present cube data. You can format cube views to display final reports or forms for data entry, both of which can be exported to a PDF or Excel spreadsheet. You create and maintain cube views on the Application tab in the Presentation section, but can access and run cube views from:

- Application Cube Views page
- Workflow Task (forms)
- Workflow Analysis section of a Workflow
- Cube Views section in the OnePlace tab
- Spreadsheet/Excel Add-In

You can incorporate cube views in report books, dashboards, form templates, spreadsheets, and extensible documents. That is why they are often called the "building blocks of reporting". You can also link cube views to support a more focused, granular analysis. See:

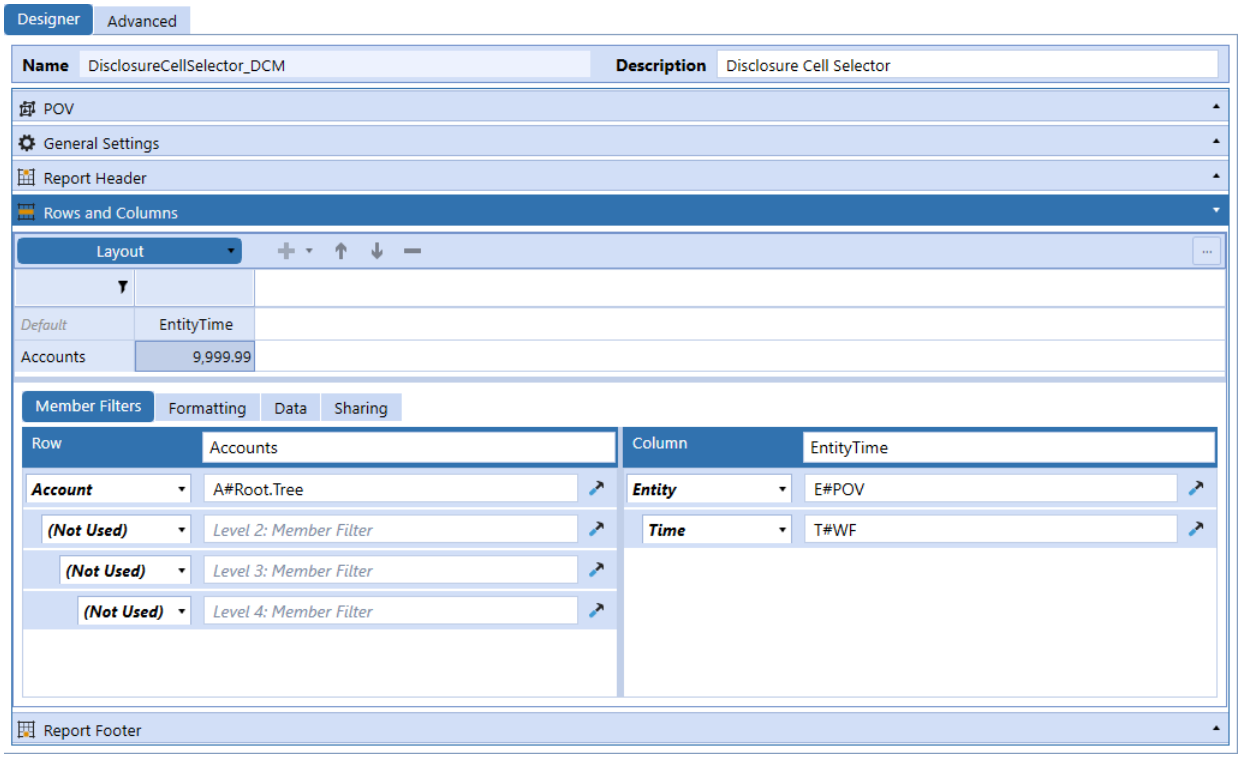
- Build a Simple Cube View
- Member Filter Builder
- Cube View Performance
- Link Cube Views
- Analyze Data Using Passed Point of View Selections

Build a Simple Cube View

A cube view can query and submit data into a cube. They are flexible in how they present data, but this section focuses on how to build a simple cube view.

Cube View Properties

Use the following features to manage cube views. In OneStream, go to **Application** > **Presentation** > **Cube Views** to access the Cube Views page. Then, under **Cube View Groups**, expand a cube view group, and select a cube view to access these properties.



- **Designer** and **Advanced**: Each tab has the same properties, but they are organized differently. You can use either tab.
- **POV**: Make selections for the cube and all dimensions used in the cube view. You do not need to make selections for any dimensions used in rows or columns. Dimensions can also be selected in the Cube POV.
- **General Settings**: Apply row and column templates, control common settings, configure headings and formatting, and set navigation links using bound parameters.
- **Report Header**: Control what displays in the header when a cube view is run as a data explorer report (PDF style).
- **Rows and Columns**: Determine the content for the rows and columns.
- **Member Filters**: Select the dimension members, parameters, variables, expansions, expressions, and business rules.
- **Formatting**: Set the header and cell formatting for the selected row or column or for the default cell for the entire cube view.
- **Report Footer**: Control what displays in the footer when a cube view is run as a data explorer report (PDF style).

Cube View Toolbar

Use the following toolbar items to manage cube views.

 Create Group: Organize cube views.

Create Profile: Organize cube view groups.

Manage Profile Members: Assign cube view groups to cube view profiles.

Create Cube View: Create a new cube view in a cube view group.

Delete Selected Item: Delete a selected item, such as a cube view or cube view group.

Rename Selected Item: Rename a selected item.

Cancel All Changes Since Last Save: Cancel unsaved changes.

Save: Save changes.

Copy Selected Cube View: Copy the selected cube view to use it as a template for another cube view.

Search: Search for cube views.

Open Data Explorer: Run a cube view and see it in a data explorer report.

Show Objects That Reference The Selected Item: View other areas where the selected item is used. For example, see where a cube view is being used in an application to know the impact of a change. This icon is available on other pages as well.

Object Lookup: Find and select an object to copy.

Manage Rows and Columns

Access these additional items on the Advanced tab and the Rows and Columns slider to manage rows and columns.

Add Row or Column: Add rows or columns to a cube view.

Move Up: Move rows or columns up.

Move Down: Move rows or columns down.

Remove Selected Row or Column: Remove a row or column.

Create Cube View Groups and Profiles

Groups organize the artifacts that users are building. Cube views are organized into cube view groups. A cube view group needs to be created before a cube view, and it can include one or more cube views. Profiles further structure the artifacts and enable them to be used in other places in the application and accessed by users. Cube view groups are organized into cube view profiles that are assigned to different areas of the application, such as a workflow profile. Cube view profiles can include one or more cube view groups.

1. On the **Cube Views** page, click **Create Group**.
2. Type a name for the new group.
3. Click **Save**.
4. Click **Create Profile**.
5. Type a name for the new profile.

   > **TIP:** Use the **Visibility** drop-down menu to specify where cube views can be viewed in the application. The default is Always, which displays the cube views in all listed locations. If you select a specific location, the cube views will display in that location. If you select Never, the cube views will only display in the Cube Views page in the Application tab.

6. Click **Save**.

Cube View Group Properties

Cube view group properties apply to general and security settings.

General (Cube View Group)

Name
The name of the cube view group.

Description
A short description of how the group is used, or what it contains.

Workspace
The currently selected workspace.

Maintenance Unit
The currently selected maintenance unit.

Security

Access
Members of this group can access the cube views in the cube biew group.

Maintenance
Members of this group have the authority to maintain the cube views in the cube view group.

> **NOTE:** Click 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, click CTRL and double-click. This will enter the correct name into the appropriate field.

Assign Cube View Groups to Cube View Profiles

Assign cube view groups to cube view profiles to control how the cube views in the group are accessed, using the visibility cube view profile property.

1. On the **Cube Views** page, under **Cube View Profiles**, select a profile.
2. Click **Manage Profile Members**.
3. Under **Available Groups**, select the group and then click the arrow to move it to the right.
4. Click **OK**.

> **NOTE:** You can assign a cube view group to multiple cube view profiles.

Create a New Cube View

1. On the **Cube Views** page, under **Cube View Groups**, select a group.
2. Click **Create Cube View**.
3. Type a name and description for the cube view.
4. Click **Save** and specify General Settings.

Copy a Cube View

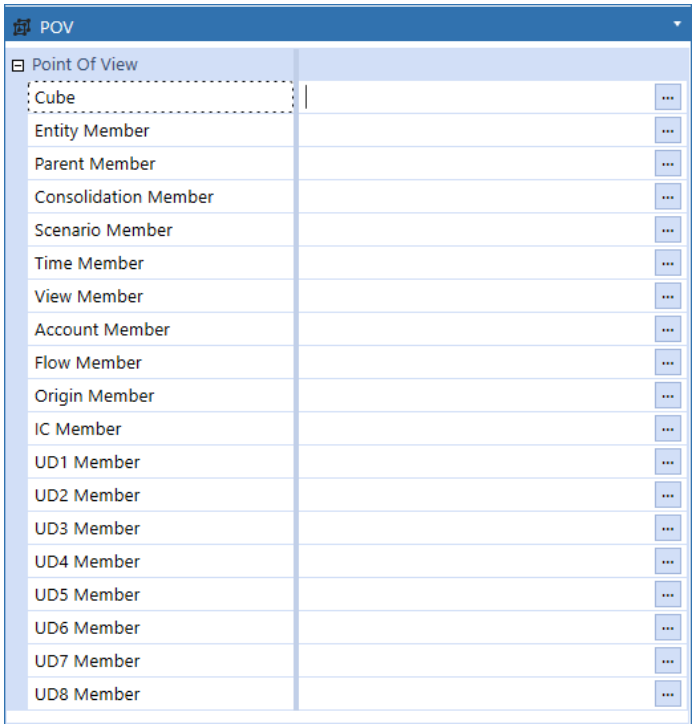
For efficiency, you can copy a cube view to use its formatting settings rather than creating a new cube view. A copy of a cube view is not linked to another cube view.

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Click **Copy Selected Cube View**.
3. Select the cube view group from the drop-down menu.
4. Enter a unique name, not used by another cube view, and click **OK**.

Set the Cube View POV

Every cube view needs a POV setting to retrieve data. To query data in a cube view, set your Cube View POV, rows, and columns. The Cube View POV will set the cube and all of the dimensions used to query the various cells of the cube views.

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Select the **POV** slider. Enter the information individually by typing in the fields or clicking the ellipsis for each member, or move the entire Cube POV.



To move the entire Cube POV, on the **POV Pane**:

a. Right-click **Cube POV**.

b. Click **Copy Cube POV**.

c. Right-click the light blue **Point Of View** header.

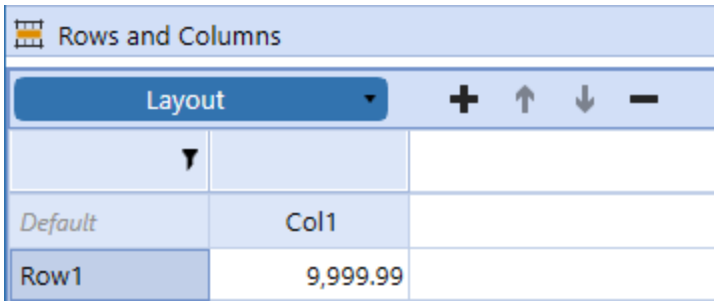
d. Click **Paste POV**.

> **TIP:** To move an entire cube POV, you can also drag and drop the Cube POV from the POV pane into the Cube View POV.

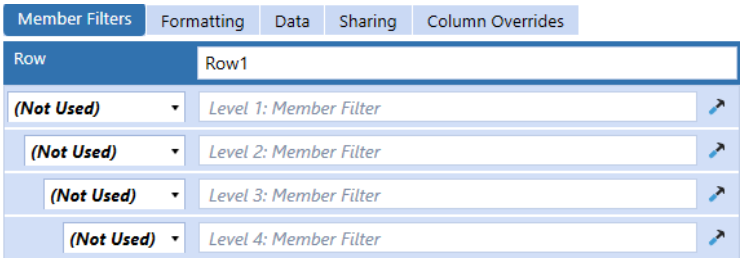
> **NOTE:** The information in the POV slider takes priority over the information in the POV pane.

Create Cube View Rows and Columns

1. To set the cube view rows and columns, select the **Rows and Columns** slider. The preview grid for the new cube view displays default row and column names: **Row1** and **Col1**. You can edit these names to make them more descriptive.



2. Click **+** to add rows or columns, and click **-** to remove rows or columns.
3. Select a row, column, or intersection in the preview grid, and then set the display options by using the settings tabs. The member filters, formatting, data suppression options, and overrides can be applied at the row and column level.



4. Click **Save**.

> **NOTE:** See Member Filter Builder.

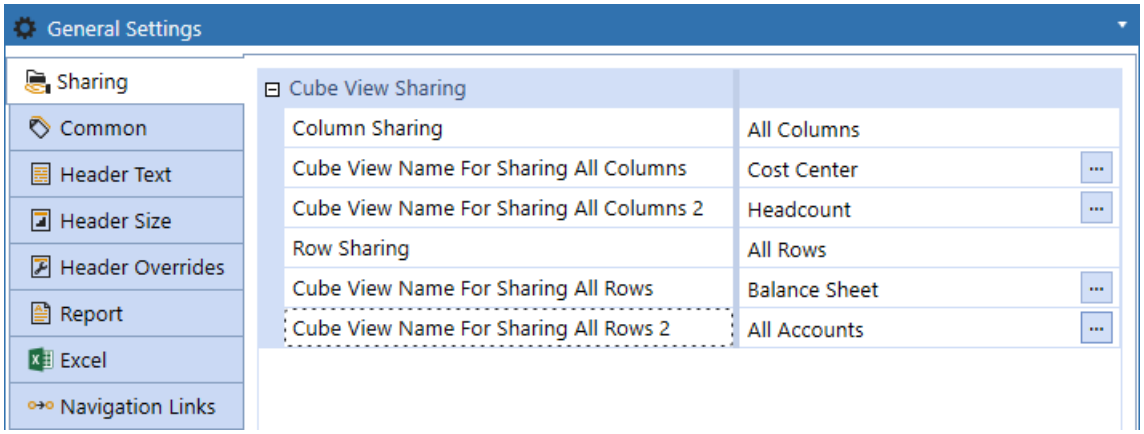
Create a Cube View Using Shared Rows and Columns

An administrator can share all rows and columns or selected rows and columns from another cube view. Sharing is more efficient when building and managing cube views.

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Go to **General Settings** > **Sharing**.
3. In the rows for the **Row Sharing** and **Column Sharing**, click in the fields to the right to access the drop-down menus. Select the option to share all or specified rows or columns.
    - **All Rows/Columns**: Enter the source cube view name of the rows and columns to be shared.
    - **Specified Rows/Columns**: Reference a single row or column from the source cube view when designing rows and columns.

> **TIP:** You can share rows and columns from two cube views. If you share rows or columns with another cube view, updates are automatically linked. This can make building a cube view more efficient.

4. If you selected the option to share all rows or columns, click the ellipsis and then select the cube view. Click **OK** to confirm.



5. Click **Save**.

Use a Cube View Template

Creating and using cube views as templates can increase efficiency and consistency for cube views that have similar formatting and properties for the rows and columns. If you plan to use templates, create a cube view group specifically for this purpose. For example, create a cube view group with the name template (Templates_Columns or Templates_Rows) and save cube views to use as templates in this cube view group. Then, other users can find and use the templates.

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view to use as a template.
2. Click **Copy Selected Cube View**.
3. Select the cube view group from the drop-down menu and type a name for the new cube view.

4. Click **OK**.

> **NOTE:** Each cube view must have a unique name.

Cube View Shortcuts

Use Cube View shortcuts to launch other cube views. This is helpful when an administrator wants to use the same cube view but does not want to use the same parameters.

For example, a user can launch a cube view for an Income Statement and be prompted with a ParamView parameter. The ParamView has two values of YTD or Periodic, meaning an Income Statement can be launched to show the data with a Year to Date view or a Periodic view (such as Month to Date).

In this case, a shortcut can be used for save two versions of the same report without prompting the user or having to maintain two cube views. In each case, the shortcut cube view Name would be the same (such as Income Statement), but the literal parameter value would be different. The YTD version would be ParamView = [YTD] and the Periodic version would be ParamView = [Periodic]. Each would open the Income Statement cube view to the proper view settings without prompting the user.

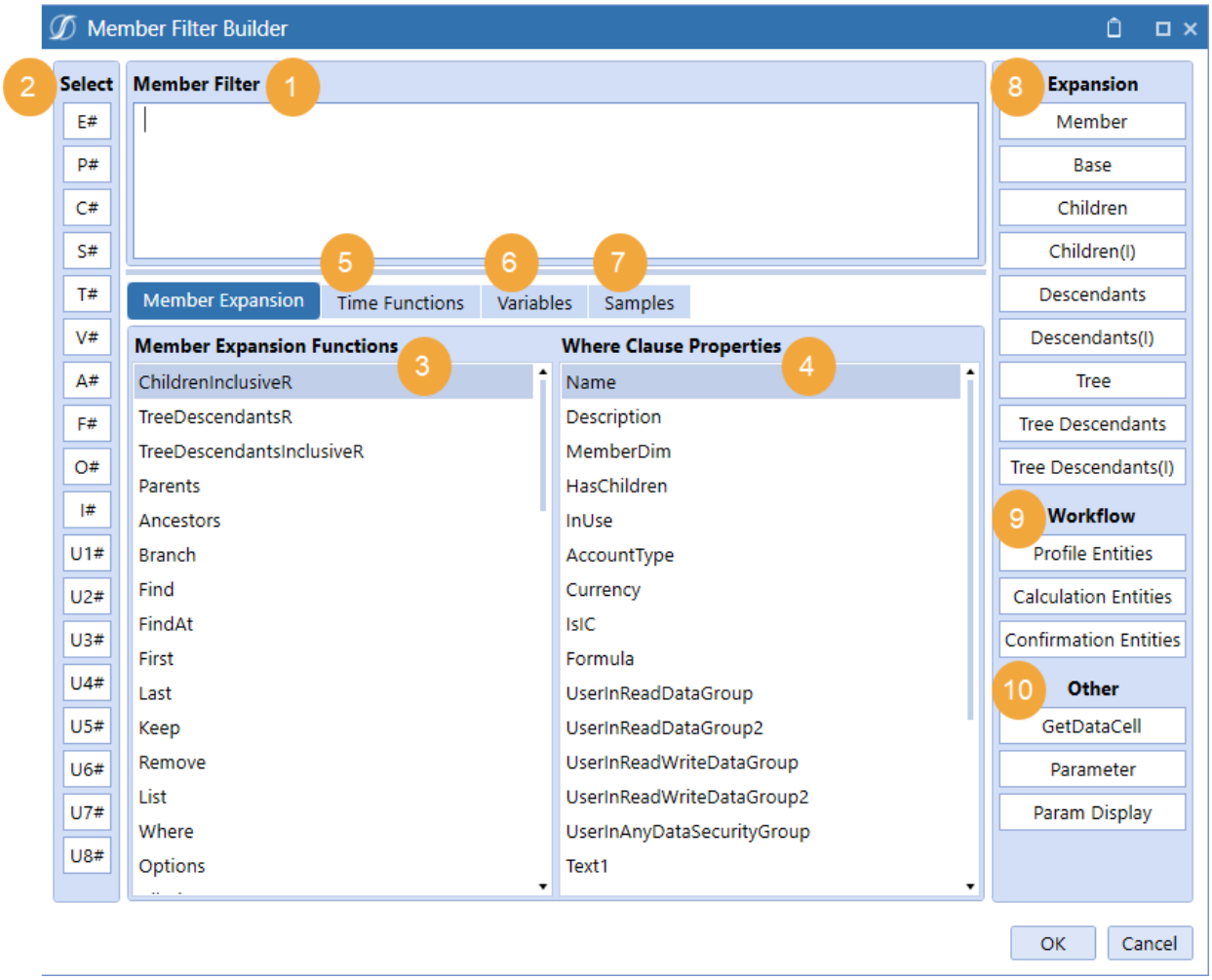
Member Filter Builder

The Member Filter Builder dialog box makes it simple to build complex member filters without having to remember or look up the proper syntax.

On the **Cube Views** page, select a cube view. Select **Rows and Columns** and then select a row or column header. Click the **Member Filters** tab and then **Member Filter Builder** to launch the dialog box.

Navigation

Hover over the items in the Member Filter Builder dialog box for additional information.



1. **Member Filter**: Location where the member filter is built. Type in this section or use the other items in the dialog box to enter information into the field.
2. **Dimension Tokens**: There is a button for each dimension that launches the appropriate selection dialog box.

3. **Member Expansion Functions**: Double-click a member expansion to add it to the member filter.
4. **Member Expansion Where Clause**: If you select Member Expansion Where, use the where clause properties to complete the expression.
   Example: UD2#AllProducts.Children.Where(Name Contains Clubs)
5. **Time Functions**: Only applies to the Time dimension, such as T#POVPrior1. Double-click a time function to add it.
6. **Substitution Variables**: Double-click a system-wide substitution variable to add it.
7. **Samples**: Refer to this tab for example syntax for complex queries and calculations, including how to build member expansions, where clause expressions, time expressions, calculations (GetDataCell and column/row expressions), custom member list expressions, and XFBR, XFCell, and XFGetMemberProperty.
8. **Expansion**: Commonly used member expansions added to the end of filters. Double-click the expansion to add it.
9. **Workflow**: Commonly used workflow member expansions used in cube views that point to a report, form, or dashboard and are affiliated with a specific workflow profile.
10. **Other**: Commonly used member filter functions that enable you to create calculated rows and columns or use a custom parameter to store member lists.

Member Expansion Tab

To reduce cube view maintenance, use simple member expansions and connect your cube views to the metadata when complex expansions are needed. Avoid referencing individual members.

- **Reverse order expansions**: These selections include ChildrenInclusiveR, TreeDescendantsR, and TreeDescendantsInclusiveR.
- **Where clauses**: Enable further qualification of the results and can pull members based on properties such as "name contains" or text properties.
- **Remove functions**: Removes some of the members from the results.
- **Parents**: Returns the direct parent of the member. There may be multiple parents if there are alternate hierarchies.
- **Ancestors**: Returns all members up the chain from the original member.
- **Options**: Advanced option if you are a report builder familiar with the extensibility design in your application. This expansion is commonly used to focus on one portion of your extensible dimensionality.

This is not an exhaustive list of all the member expansions. Each expansion can also be found in the Samples tab. Hover over each to see the purpose and a sample of the expected syntax. Double-click the member expansion to place the sample syntax in the Member Filter field.

Use the member filter to filter data by creating a list of restricted members. Member filters can contain multiple member scripts. A member script queries a defined set of dimensional members. Members can be specified for any or all dimensions and the primary dimension can also specify a member expansion formula (for example, Descendants).

For example, a simple member script that returns the year 2022: T#2022.

For example, a member script with a member expansion that returns all the income statement accounts: A#[Income Statement].Descendants.

If one or a few dimensions in the member script have a dimension token, the remaining dimensions are pulled from the Cube View POV, the Global POV, or Workflow, Time, and Scenario. Separate each dimension that has a token in the member script with a colon as in the following example:

Cb#GolfStream:E#Houston:P#Texas:C#Local:S#Budget:T#2022M3:V#YTD:A#60000:F#None: O#Forms:IC#None: U1#Sales: U2#HybridXL: U3#Northeast: U4#SportsCo: U5#None: U6#None: U7#None: U8#None

For example, a member filter that contains three different member scripts that returns the Actual, Budget, and Forecast scenarios: S#Actual, S#Budget, S#Forecast.

Time Functions Tab

Time is a fixed dimension and is based on the time dimension type associated with the application. OneStream software has custom time dimensions (for example, weekly and monthly), which affect time functions and expansions. Time functions pivot the member, and time expansions extend the member.

- Time function example to display the prior period of the time member in a Cube POV: T#POVPrior1
- Time expansion example to display all months in the year 2022: T#2022.Months

Add a time function from the Time Functions tab:

1. In the **Member Filter Builder** dialog box, click the **Time Functions** tab.
2. Select the type of time function to display the examples available: POV, WF, Global, or General. You can also select All.
3. Double-click a function to populate it in the **Member Filter** field.

> **NOTE:** Click the **Samples** tab to view more time functions.

The following time function example demonstrates a style of syntax that provides flexibility. You can choose from a variety of time functions, but this one enables you to pivot easily on the year and period separately:

T#YearPrior1(|PovTime|)PeriodNext1 (|PovTime|)

- Prior year from the POV time: **T#YearPrior1(|PovTime|)**
- Next period from the POV time: **PeriodNext1 (|PovTime|)**

Variables Tab

Substitution variables are short scripts that use pipe characters to include a predefined substitution variable. For example, |WFProfile| would refresh to display the current workflow profile name. They come with every installation of OneStream and cannot be edited, so you do not need to create or maintain them.

Substitution variables can be used throughout the application. For cube views, you can use them for:

- Headers and footers
- Rows and columns
- Cube view page captions

Substitution variables are always referenced with pipes (for example, |POVTime|). The Member Filter Builder dialog box includes substitution variables in the following categories:

- POV (Cube POV)
- WF (Workflow POV)
- Global (Global POV)
- CV (Cube View POV)
- MF (Member Filter)
- General (items not related to a POV)

When using a substitution variable to return a member name, the prefix indicates where the value of the variable is pulled from. Choose if it should refresh based on the Cube View POV (CV), Workflow POV (WF), Cube POV (POV), Member Filter (MF), or Global POV (Global).

Choose if you want to display the member description (usually in headers and footers) by adding Desc as the suffix. Time has an added short description option that is set through the time profiles. For example, |WFTimeDesc| returns the description Feb 2022, and |WFTimeShortDesc| returns the short description Feb.

Common substitution variables:

- Username that ran the report: |UserName|
- Cube view name: |CVName|
- Members text properties: |Text1|
- Today's date: |DateDDMMYYYY|

On the Variables tab, you can double-click a substitution variable to add it to a member filter. You can also copy a substitution variable from the Object Lookup dialog box:

1. On the **Cube Views** page, click **Object Lookup** to open the dialog box.
2. Under **Object Type**, select **Substitution Variables**.
3. Select a substitution variable from the list. Use the Filter field to find a specific option.
4. Click **Copy to Clipboard** and then paste the substitution variable where needed.

Samples Tab

The Samples tab holds example syntax for more complex queries and calculations in the following categories:

- Member Expansions
- Where Clause Expressions
- Time Expressions
- Calculations (GetDataCell Expressions and Column/Row Expressions)
- Custom Member List Expressions
- XFBR, XFCell, and XFGetMemberProperty

Add an expression from the Samples tab:

1. In the **Member Filter Builder** dialog box, click the **Samples** tab.
2. Double-click a sample to populate it in the **Member Filter** field.

Where Clauses

Where clauses are commonly used in reporting to create a more flexible query. The where clause can pull members based on properties:

- Text properties
- Portions of the description or name (for example, starts with or contains)
- Security
- Account types (account dimension only)
- Intercompany (entity and account dimension only)
- Specific currency (entity dimension only)
- In use property
- Has children

> **NOTE:** Where(HasChildren = True) can only be used for the dimensions assigned to the cube on the default Scenario Type.

- Has a member formula

> **NOTE:** Click the **Samples** tab to view common where clauses.

Add a where clause from the Samples tab:

1. In the **Member Filter Builder** dialog box, click the **Samples** tab.
2. Expand the **Where Clause Expressions** list.
3. Double-click a sample to populate it in the **Member Filter** field.

Cube View Performance

Cube view performance includes the following information:

- Database Sparsity
- Row and Column Suppression
- Suppressed Members
- Sparse Row Suppression
- Cube View Paging

Database Sparsity

Sparsity is the ratio of data record volumes in the cube compared to the dimensions modeled in the cube. We see sparsity when the data unit (combination of cube, entity, parent, consolidation, scenario, and time) has sparsely populated data intersections across the account-type dimensions (account, intercompany, flow, and user-defined). The absence of data records can affect reporting performance because it is difficult to render reports if data is sparse.

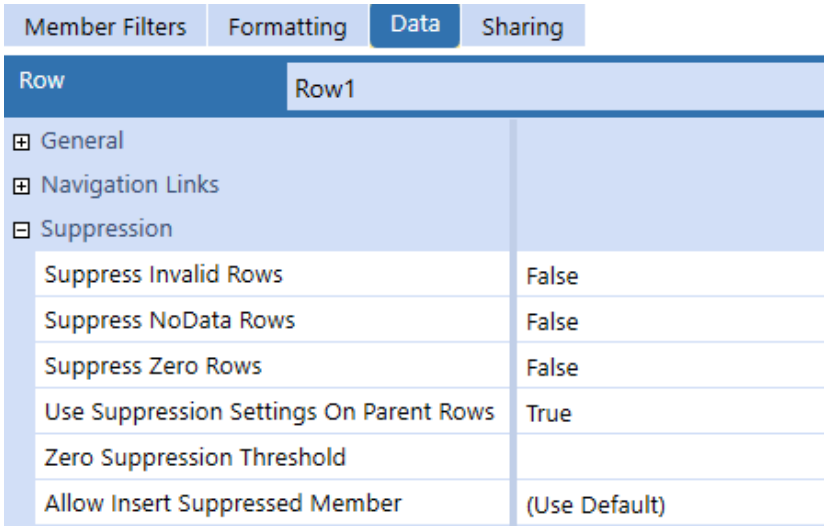
Avoid sparsity in your application design when possible. Use analytic blend, extensibility, or other design frameworks to ensure the Dimension Library is designed for your reporting structure and to minimize sparsity. However, even with an optimal design, sparsity can still occur if a large report pulls in a lot of dimensions. Suppression settings can help improve reporting performance if this is the case.

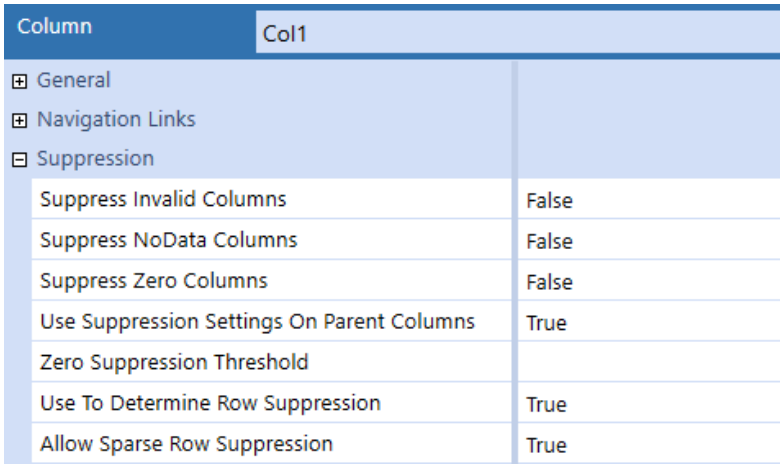
Row and Column Suppression

Use row and column suppression in larger reports to make them easier to read while still enabling the report builder to create a low maintenance report based on the metadata design.

To adjust the suppression settings:

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Select **Rows and Columns** to expand the slider.
3. Select the row or column and then click the **Data** tab. The following settings are available.





- **Suppress Invalid Rows/Columns**: Set to True to suppress any invalid cells.
- **Suppress NoData Rows/Columns**: Set to True to suppress any cells without data. When data entry is required, the recommended setting is False.
- **Suppress Zero Rows/Columns**: Set to True to suppress any cells containing zeros.

- **Use Suppression Settings on Parent Rows/Columns**: This property relates to the prior properties and controls whether the parent members in this member filter use the same settings.
- **Zero Suppression Threshold**: When Suppress Zero Rows/Columns is set to True, enter a value to suppress all numbers below it. All numbers below the specified number are recognized as zeros. For example, entering 499.99 results in every number lower than that value being recognized as zero and therefore suppressed.
- **Allow Insert Suppressed Member** (rows only): Use this setting to access a member currently suppressed for data entry purposes. You can only use this with cube views and form templates.
    - **All**: Enables visibility to all cube view row expansions
    - **False**: All row expansions remain suppressed
    - **Nested**: Enables visibility of the row expansions two through four
    - **Innermost**: Enables visibility of the row expansion that is at the bottom level
- **User To Determine Row Suppression** (columns only): Set to True to improve performance on large cube views by enabling the designer to better define how to apply row suppression.
- **Allow Sparse Row Suppression** (columns only): Provides performance improvements for cube views that use multiple nested row dimensions and works in conjunction with a General Settings property. When set to True, sparse row suppression is applied to the entire cube view. It can be turned on and off for specific columns.

Suppressed Members

This section includes instructions for how to manage suppressed members.

Apply Modify Suppression Property

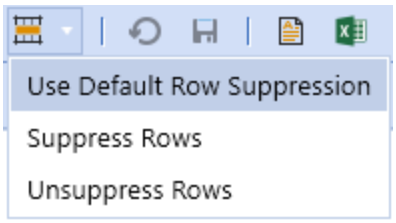
With this setting, you can choose whether rows are suppressed from the data explorer grid. This feature is useful for a small cube view that will not generate a large number of additional rows if the suppression is turned off.

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Go to **General Settings** > **Common**.
3. Under **Restrictions**, set the **Can Modify Suppression** property to **True**.
4. Click **Save**.

Modify Suppression

To modify suppression, you must first apply the correct property. See Apply Modify Suppression Property.

1. Open a cube view in Data Explorer view.
2. Select the **Row Suppression** drop-down menu.



3. Choose the option to apply.
    - **Use Default Row Suppression**: Apply the cube view suppression settings.
    - **Suppress Rows**: Suppress any rows with zeros or no data regardless of the cube view settings.

- **Unsuppress Rows**: Unsuppress all rows that were suppressed with zeros, no data, or invalid data regardless of the cube view settings.

Apply Property to Allow Insert Suppressed Members

You can insert suppressed members so that their data is entered in the cube view.

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Go to **Rows and Columns** > **Data** > **Suppression** and confirm that **Allow Insert Suppressed Member** is set to **All**.

   > **NOTE:** The Allow Insert Suppressed Member option is only available for rows.

3. Click **Save**.

Insert Suppressed Members

To insert suppressed members, you must first apply the correct property. See Apply Property to Allow Insert Suppressed Members.

1. Open a cube view in Data Explorer view.
2. Right-click in the account or entity.
3. Click **Insert Suppressed Member**.
4. To the right of the account or entity field, click **Select Member**.
5. Select an item and click the arrow to add it to the **Result List**.
6. Click **OK** and then click **OK** to confirm.
7. Enter data as needed in the white cells.
8. Click **Save** and then click **OK**.

Sparse Row Suppression

Designs for analytic reports typically have multiple dimensions nested in rows. The combination of members generated from the nested expansions can easily result in billions of potential expanded rows, many of which may not have data. In these designs, standard row and column suppression (invalid, no data, or zero) in the cube view would still require each of those billion rows to be inspected individually for data. So, we recommend sparse row suppression be enabled to enhance the performance of a large cube view when, due to widespread database sparsity, the report is likely to return many records without data and take a long time to run. Sparse row suppression evaluates the data records of the cube view intersections and filters records with no data (not zeros) before rendering the cube view.

> **IMPORTANT:** Sparse row suppression cannot be applied to dynamically calculated data through dynamically calculated members and cube view math. Avoid errors by correctly applying the settings on the dynamically calculated columns.

> **NOTE:** When sparse row suppression is applied, OneStream assesses all row data before displaying the cube view. Cells containing dynamic calculations are populated on-the-fly. When OneStream assesses the rows of data, the rows containing dynamically calculated data are omitted from the cube view because the data is not saved in the database.

To enable sparse row suppression in cube views:

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Go to **General Settings** > **Common**.
3. Under **Suppression**, set **Allow Sparse Row Suppression** to **True**.
4. Select **Rows and Columns** to expand the slider. Select the row and then the **Data** tab.

5. Ensure the additional row suppression properties you will use are also set to **True**.

> **NOTE:** Any row assigned a suppression setting will be enabled for sparse row suppression. If no suppression is applied, sparse row suppression will not be applied.

6. In the **Rows and Columns** slider, select the column and then the **Data** tab.
7. Under **Suppression**, set **Allow Sparse Row Suppression** to **True**.
8. Ensure the additional column suppression properties you will use are also set to **True**.

> **NOTE:** If columns contain dynamically calculated data, set **Use to Determine Row Suppression** and **Allow Sparse Row Suppression** to **True** to avoid an error.

Cube View Paging

Cube view paging is only applied to the data explorer view of a cube view and is used to enhance the performance of cube views containing more than 10,000 unsuppressed rows. The cube view will attempt to return up to 2000 unsuppressed rows within a maximum processing time of 20 seconds. The purpose of paging is to protect the server from large cube views that could affect application performance.

Evaluations are performed on the potential size of the rows and the processing requirements to determine if paging is enabled and the number of rows returned.

- Evaluation 1 – Enable Paging: An evaluation is performed on the entire cube view to determine the total number of possible unsuppressed rows that will be generated. If the total number of potential unsuppressed rows is less than 10,000, no paging will be enabled.
- Evaluation 2 – Paging Enabled: If the total number of unsuppressed rows is greater than or equal to 10,000, paging is enabled.

- Evaluation 3 – Paging: Once paging begins, the cube view evaluates the rows attempting to return a minimum of 20 to a maximum of 2000 unsuppressed rows. In the case of nested dimensions on rows, the evaluation starts on the left most dimension expansion, as defined in the cube view. After a maximum processing time of 20 seconds, the first page of the cube view will be returned for display containing only the rows that completed processing during the time constraint. For this reason, cube view pages are not a fixed number of rows. The rows are ultimately determined by their time to process. This also relates directly to the percentage display, because each page is generated by processing time requirements, and the last page is not known while the cube view is running. Therefore, this percentage is not intended to be a precise measurement.

When a row is defined with nested dimensions, the paging evaluation is performed on the left most dimension. For each expansion of the left most dimension, the paging will not progress to the next sibling until all the records are returned by all the other dimension expansions to be completed.

To set the properties, on the **Cube Views** page, under **Cube View Groups**, select a cube view. Go to **General Settings** > **Common**. The following settings are available.

- **Max Unsuppressed Rows Per Page**: Determines how many rows are written before the cube view starts paging (default is -1). The maximum value is 100,000.
- **Max Seconds To Process**: Determines how many seconds the cube view processes before it starts paging (default is -1). The maximum value is 600 seconds.

See Cube View Paging for more information on how to cancel a long-running cube view through Task Activity.

Advanced Cube Views

There are several advanced uses with cube views such as setting them to change by Point of View or Workflow. Using advanced settings let you create and maintain fewer cube views.

With Workflow

Set the POV, rows, and columns in the cube view so it is driven by the Workflow POV and the entities assigned to the workflow profile in use. By doing this, you can make forms, dashboards, cube views, and reports driven dynamically by the workflow profile.

Workflow Entities

Using an expression such as E#Root.WFProfileEntities from within the Rows or Columns Tab shows the entity or entities assigned to that particular Workflow Profile at run time. WFProfileEntities or similar expressions cannot be assigned to the POV because there can be more than one and the POV only requires a single member.

Workflow Scenario

Under the cube view Point of View Slider, select the WF Member for the Scenario dimension, or use WFScenario|, or a similar Substitution Variable in Rows and Columns.

Workflow Time

Under the cube view Point of View Slider, select the WF Member for the Time dimension, or use WFTime|, or a similar Substitution Variable in Rows and Columns.

With Dashboard and Form Parameters

When using dashboards and forms, use parameters to focus the data to what is needed. For example, when a Dashboard is launched, it can prompt for the specific entity or region needed. See Parameter Guides for details.

Within the cube view, refer to parameters in the Point of View and Rows and Columns Slider to restrict the query to just the data expected. Surround the parameter name in pipes and exclamation points (for example, |!ParameterName!|).

The following image shows an example of using the entity and region parameters in the cube view's Point of View:

Point Of View	
Cube	GolfStream
Entity Member	|!WorkflowEntityPrompt!|
Parent Member	
Consolidation Member	Local
Scenario Member	WF
Time Member	WF
View Member	YTD
Account Member	
Flow Member	None
Origin Member	Top
IC Member	Top
UD1 Member	Top
UD2 Member	Top
UD3 Member	|!Region Parameter (Workflow)!|
UD4 Member	Top

> **TIP:** Use dashboard parameters as a single repository for parameters that are used in dashboards or forms. If a parameter is referred to within a cube view (for example, |!ParameterName!|) and there is not a parameter by that name associated with the form, it will search through the application's dashboard parameters for one with that name and use it.

Changing Member Display Name

Change the name returned with a member script in a cube view by adding :Name("enter name here") at the end of the member filter. The double quotes around the Name() value are optional.

For example

```
A#60999:Name("Net Sales") or A#60999:Name(Net Sales)
```

Format a Cube View

This section includes the following information to format a cube view:

- Overview
- Basic Formatting
- Rename Rows and Columns
- Create Conditional Formatting
- General Settings
- Format a Report Page

Overview

Formatting settings and overrides combine to create the cell format. You can apply formatting and isolate it to nested expansion levels on rows or columns by using property filters found in the conditional formatting dialog box.

Apply property filters in this order:

1. Application properties standard report settings
2. Cube view default settings
3. Column settings
4. Row settings
5. Column row overrides
6. Row column overrides

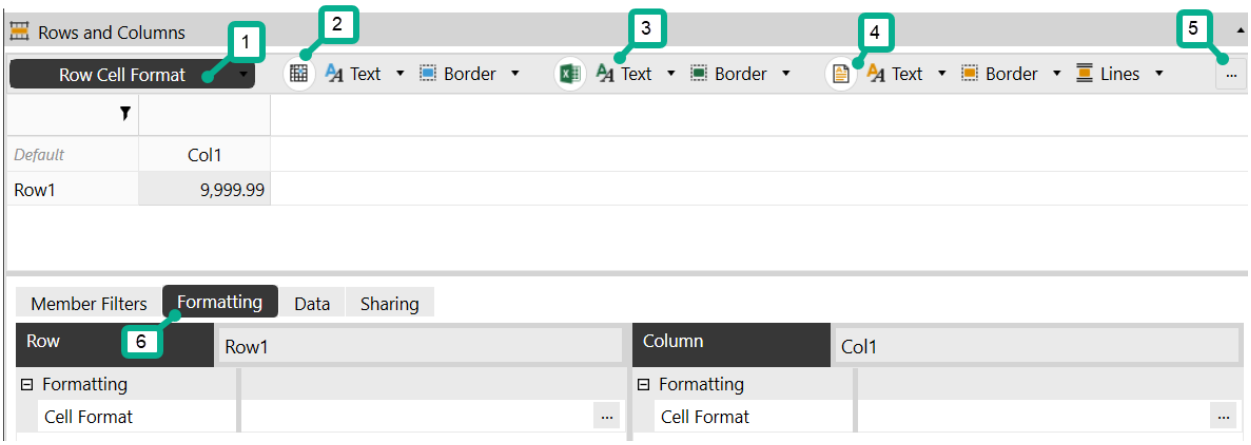
The primary output for a cube view is the data explorer grid. You can also export to Excel or use a report viewer.

Basic Formatting

Cube view formatting controls how cube views display. Most of the format settings transfer to Excel and can still be overridden with the standard Excel styles. The formatting also transfers to reports where the cube view is being used in a data adapter.

Cube view formatting can be selected for the entire cube view or specific rows, columns, headers, and individual cells. The formatting options enable the number formats, percentage signs, scaling, currency symbols, colors, fonts, and font size to be unique to the business needs. Label row and column sets to support cube view maintenance when editing, formatting, and applying calculations.

In the cube view, select the **Rows and Columns** slider. Then click an individual cell or header and then the **Formatting** tab to view and edit the toolbar options for data explorer, Excel, and report formats. The icons in the toolbar include the most common formatting properties. Each icon coincides with the cell selected in the Rows and Columns slider. For example, if a column header is selected, the formatting changes made will only affect the specific column header.



1. Selector
2. Data explorer formatting options
3. Excel export formatting options
4. Report formatting options
5. Advanced formatting options
6. Formatting tab

Set Up Basic Formatting for a Cube View

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Select **Rows and Columns** to expand the slider.
3. Select the row, column, or cell to edit.

   > **NOTE:** Click the **Default** cell to update the default settings for all headers and cells.

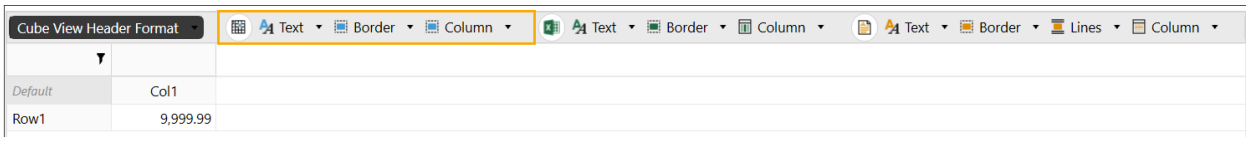
4. Click the **Formatting** tab.
5. Select the options to change the header and cell format settings for the data explorer, Excel, and report output types.

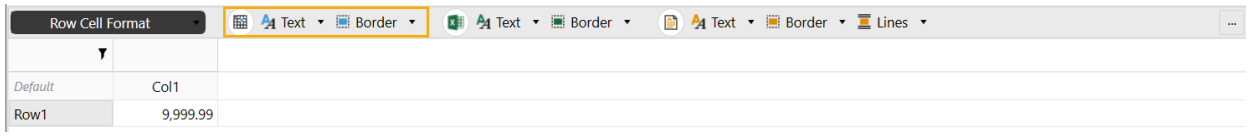
Data Explorer

The formatting settings available are organized in the following groups.

- **Text**: Font, color, size, bold, and italic settings. When you select a cell, the following formatting properties become available in the Text drop-down menu: number, zero offset, scale, flip sign, show percent sign, and show currency.
- **Border**: Background color and gridline color
- **Column**: Whether the column is visible and its width

There are different settings available for the headers and cells.



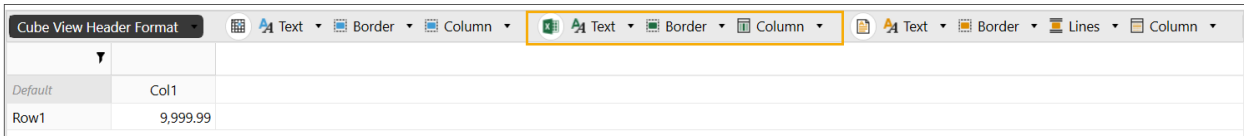


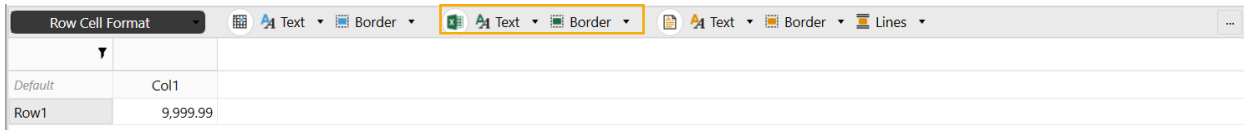
Excel

The formatting settings available are organized in the following groups.

- **Text**: Color, horizontal alignment, vertical alignment, indent level, and wrap settings. When you select a cell, the following formatting properties become available in the Text drop-down menu: number and use scale.
- **Border**: Background color and the color and line styles of the cell borders
- **Column**: Column width

There are different settings available for the headers and cells.



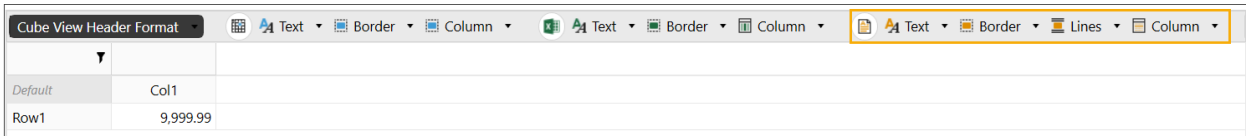


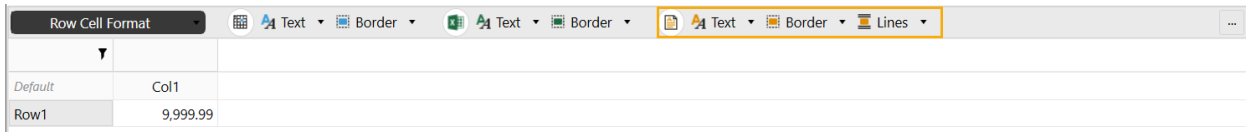
Report

The formatting settings available are organized in the following groups.

- **Text**: Color, alignment, size, and underline settings. When you select a cell, the following formatting properties become available in the Text drop-down menu: no data number, use numeric binding, and numeric binding.
- **Border**: Background color and cell borders
- **Lines**: Settings for the top and bottom lines for the first row, padding, color, and thickness
- **Column**: Column width

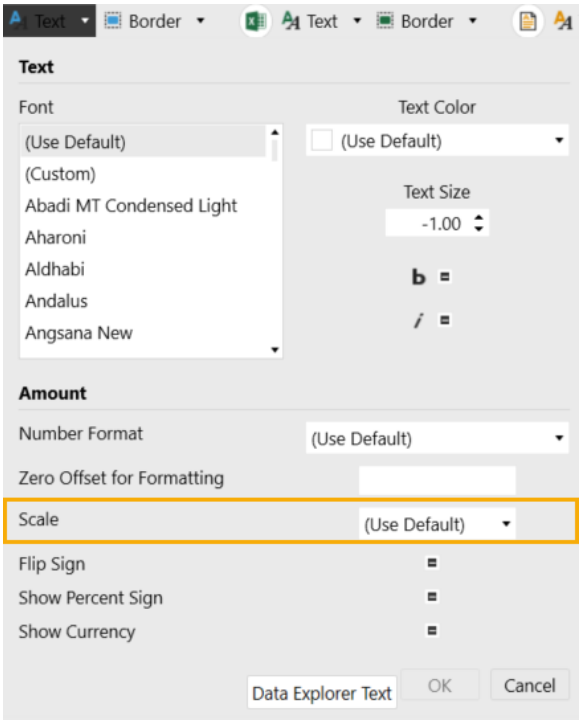
There are different settings available for the headers and cells.



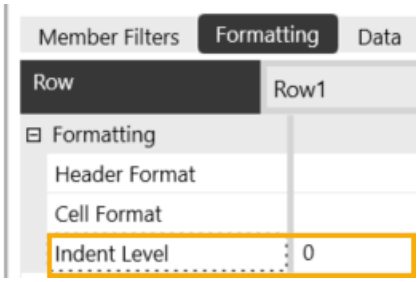


Apply Scaling, Indents, and Currencies

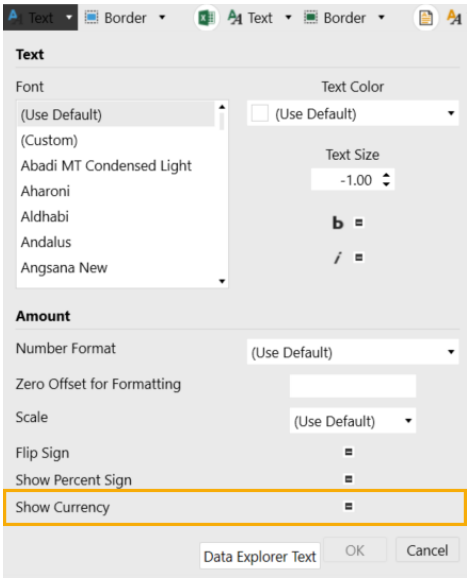
- **Scale**: Open the **Data Explorer Text** drop-down menu to view the **Scale** setting. You can also apply it to Excel and report formatting.



- **Indent level**: On the **Formatting** tab for a row or column header, view the **Indent Level** setting.



- **Show currency**: Open the **Data Explorer Text** drop-down menu to view the **Show Currency** setting.



Header Format Details

The formatting properties for a cube view header include default, row, and column headers. These properties are in the Rows and Columns slider. Not all properties are available for each header.

1. In the **Rows and Columns** slider, select the **Default** cell to apply the formatting settings to all of the cube view headers.

   > **NOTE:** You can select an individual row or column header instead if you want to edit the settings for a specific header.

2. In the **Formatting** tab, click the ellipsis to the right of **Header Format**.
3. Click **Format** to view the options. The Excel section only affects how the cube view data display in Excel.

- **Custom Parameters**: Click the ellipsis to select and assign a custom parameter such as a cube view style to the cube view header.
- **RowExpansionMode**: Control expansion of nested rows. This determines how rows are displayed in the data explorer grid. The default sets this property to False.
- **ShowDimensionImages**: Set to False to hide the dimension icons in the data explorer grid row and column headers. The default setting will display the dimension icons.
- **HeaderWrapText**: When False, column and row header text does not wrap. When set to True, column and row header text displays as wrapped. This applies to Default Cube Views and works with Header Overrides.

  > **NOTE:** **ColumnWidth** and Row Header Width must be set from the Header Size settings menu for row header text to wrap

- **IsColumnVisible**: Set to False to hide specific columns. Set this property at run time with a parameter to show and hide detail.
- **ColumnWidth**: Enter a numerical value for column width. If the header label exceeds the column width, the header will automatically wrap the text.
- **ColumnHeaderWrapText**:False, column header text does not wrap. When set to True, column header text displays as wrapped. This applies to individual column headers and works with Header Overrides. ColumnWidth must be set for Column Headers to wrap.

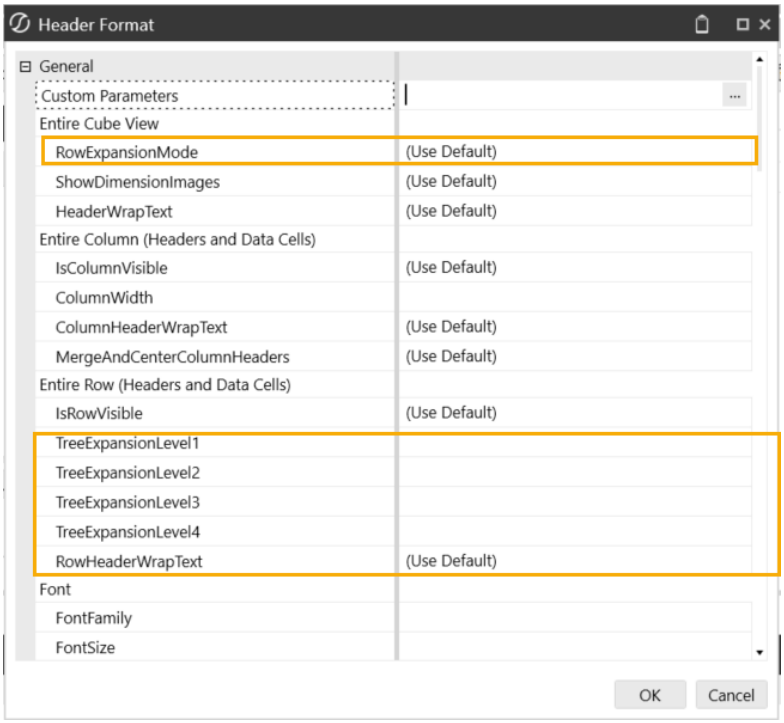
  > **NOTE:** **ColumnWidth** must be set for header text to wrap

- **MergeAndCenterColumnHeaders**: When set to False, column headers are left justified. When set to True, column headers are centered. This applies to Default Cube Views or individual column formatting in Data Explorer and Reports.

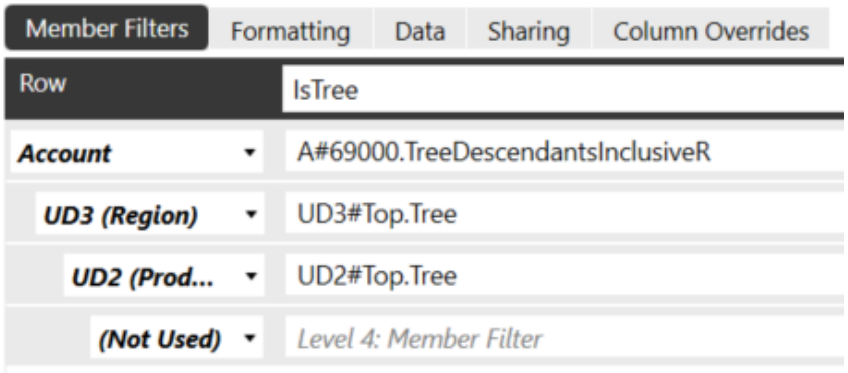
  > **NOTE:** MergeAndCenterColumnHeaders does not apply to Header Overrides.

- **IsRowVisible**: Set to False to hide specific rows. Set this property at run time with a parameter to show and hide detail.
- **TreeExpansionLevel1-4**: Automatically expand up to 4 nested rows in Data Explorer. Enter the number of rows to be expanded from 0 – 4 in the corresponding TreeExpansionLevel.

> **NOTE:** For TreeExpansionLevel to work, RowExpansionMode from the Default Header must be set to "Use Default". Using Collapse All or Expand All overrides this setting.

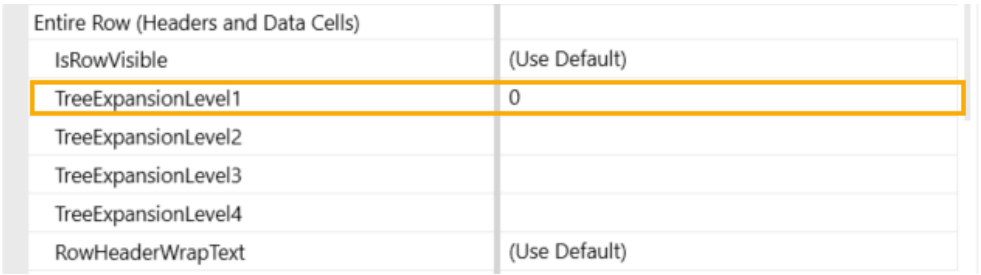


TreeExpansionLevels are tied to the Member Filter Dimension Types on the Designer tab and require a .Tree Member Expansion.

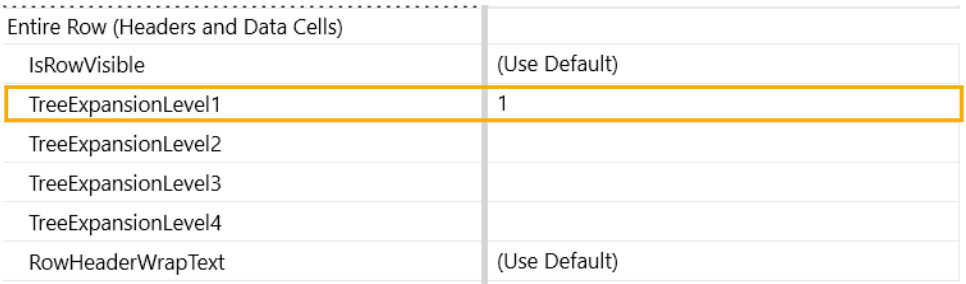


NOTE: Default settings are overwritten by Row Overrides when set.

Set the TreeExpansionLevel to 0 to collapse all nested rows.



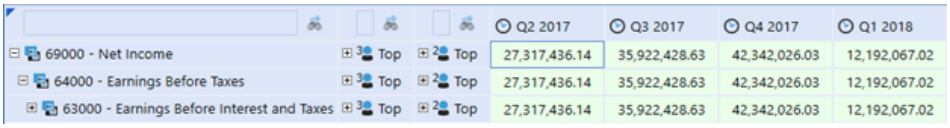
Set the TreeExpansionLevel to 1 to expand the first level of nested rows.



			Q2 2017	Q3 2017	Q4 2017	Q1 2018
69000 - Net Income	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02
64000 - Earnings Before Taxes	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02

Set the TreeExpansionLevel to 2 to expand the first two levels of nested rows.

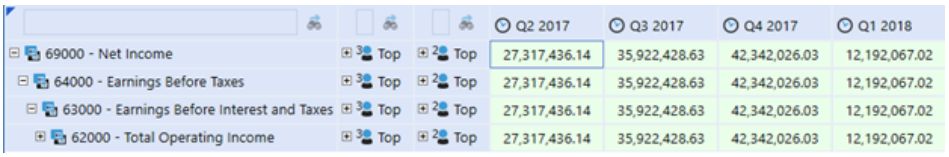
Entire Row (Headers and Data Cells)	
IsRowVisible	(Use Default)
TreeExpansionLevel1	2
TreeExpansionLevel2	
TreeExpansionLevel3	
TreeExpansionLevel4	
RowHeaderWrapText	(Use Default)



			Q2 2017	Q3 2017	Q4 2017	Q1 2018
69000 - Net Income	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02
64000 - Earnings Before Taxes	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02
63000 - Earnings Before Interest and Taxes	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02

Set the TreeExpansionLevel to 3 to expand the first 3 levels of nested rows.

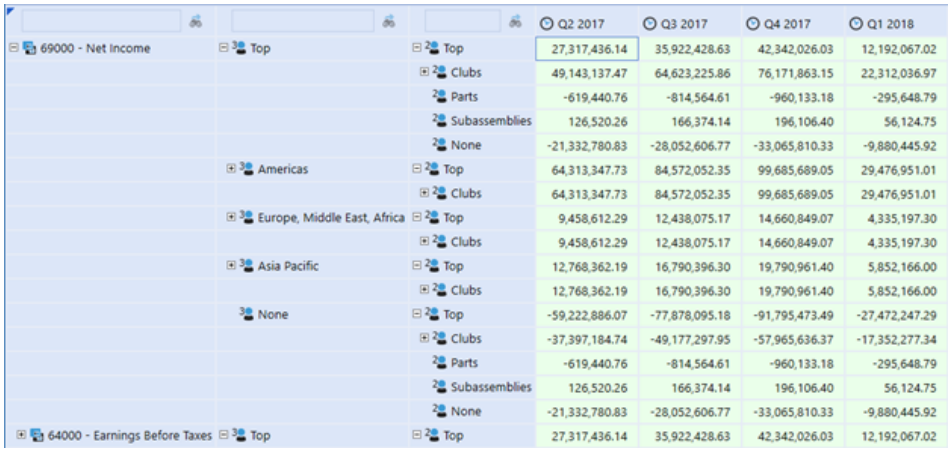
Entire Row (Headers and Data Cells)	
IsRowVisible	(Use Default)
TreeExpansionLevel1	3
TreeExpansionLevel2	
TreeExpansionLevel3	
TreeExpansionLevel4	
RowHeaderWrapText	(Use Default)



			Q2 2017	Q3 2017	Q4 2017	Q1 2018
69000 - Net Income	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02
64000 - Earnings Before Taxes	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02
63000 - Earnings Before Interest and Taxes	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02
62000 - Total Operating Income	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02

This example shows the TreeExpansionLevel 1, 2, and 3 expanded:

Entire Row (Headers and Data Cells)	
IsRowVisible	(Use Default)
TreeExpansionLevel1	1
TreeExpansionLevel2	1
TreeExpansionLevel3	1
TreeExpansionLevel4	
RowHeaderWrapText	(Use Default)



			Q2 2017	Q3 2017	Q4 2017	Q1 2018
69000 - Net Income	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02
		Clubs	49,143,137.47	64,623,225.86	76,171,863.15	22,312,036.97
		Parts	-619,440.76	-814,564.61	-960,133.18	-295,648.79
		Subassemblies	126,520.26	166,374.14	196,106.40	56,124.75
		None	-21,332,780.83	-28,052,606.77	-33,065,810.33	-9,880,445.92
	Americas	Top	64,313,347.73	84,572,052.35	99,685,689.05	29,476,951.01
		Clubs	64,313,347.73	84,572,052.35	99,685,689.05	29,476,951.01
	Europe, Middle East, Africa	Top	9,458,612.29	12,438,075.17	14,660,849.07	4,335,197.30
		Clubs	9,458,612.29	12,438,075.17	14,660,849.07	4,335,197.30
	Asia Pacific	Top	12,768,362.19	16,790,396.30	19,790,961.40	5,852,166.00
		Clubs	12,768,362.19	16,790,396.30	19,790,961.40	5,852,166.00
	None	Top	-59,222,886.07	-77,878,095.18	-91,795,473.49	-27,472,247.29
		Clubs	-37,397,184.74	-49,177,297.95	-57,965,636.37	-17,352,277.34
		Parts	-619,440.76	-814,564.61	-960,133.18	-295,648.79
		Subassemblies	126,520.26	166,374.14	196,106.40	56,124.75
		None	-21,332,780.83	-28,052,606.77	-33,065,810.33	-9,880,445.92
64000 - Earnings Before Taxes	Top	Top	27,317,436.14	35,922,428.63	42,342,026.03	12,192,067.02

- **RowHeaderWrapText**: When False, row header text does not wrap. When set to True, row header text displays as wrapped. This applies to individual row headers and works with Header Overrides.

  > **NOTE:** Row Header Width must be set from the Header Size settings menu for row header text to wrap.

- Font and color options are available. For the FontFamily, any .Net font installed on the Excel client operating system is included in the font family. The FontSize default is 11.

- **ExcelMaxOutlineLevelOnRows** and **ExcelMaxOutlineLevelOnCols**: Up to six outline levels can be used for the creation of collapsible and expandable Excel groups of rows or columns when exporting. Enter the number of outline levels needed. This relates to the Excel Indent Level setting that is applied in the header format on each row or column, so the number entered here should equal the highest setting applied on the rows or columns.
- **ExcelExpandedOutlineLevelOnRows** and **ExcelExpandedOutlineLevelOnCols**: When a cube view is exported to Excel, these settings control to which outline level the file is initially opened. The default is 1, which means each grouping is fully collapsed.
- **ExcelColumnWidth**: Default Excel column width in pixels unless overridden in columns.
- **ExcelOutlineLevelCol** and **ExcelOutlineLevel**: Default Excel outline level for rows and columns is 1 unless specified here or overridden in rows or columns.
- **ExcelHorizontalAlignment** and **ExcelVerticalAlignment**: Control how the data will be aligned horizontally and vertically using the Excel standard alignment options.
- **ExcelIndentLevel**: The number of characters to indent the text or value.
- **ExcelWrapText**: This determines if text like headers will be wrapped in the output.
- Color and border options are available.

- **ReportSeparateBandPerCubeViewRow**: This is a setting to optimize performance for large cube views that will be dynamically generating a data explorer report with many row definitions with unique formatting. Default setting is (Use Default), which is a conditional setting. This applies to when a data explorer report is run from a cube view and background temporary database tables in memory, one for each row name in the cube view are created. If there are more than 100 row definitions, fewer report bands (separate temporary in-memory tables) are created, combining rows where formatting is the same. Set to True to create a separate table per cube view row definition, which may affect performance with hundreds or thousands of row definitions. Set to False to combine row definitions of the same format into fewer temporary in-memory tables.
- Page and column header and table layout options are available.
- **ReportColumnWidth** and **ReportRowHeight**: Sets the width of a column or the height of a row by number of pixels.
- **ReportTopLinesOnFirstRowOnly** and **ReportBottomLinesOnLastRowOnly**: If the row definition results in multiple rows, a top line will be placed on the first line only or a bottom line will be placed on the last line only.
- **ReportBandHeightBeforeFirstRow** and **ReportBandHeightAfterLastRow**: Allows a number to be entered in pixels to add before the first row or after the last row of where a row definition starts.

  > **NOTE:** The top and bottom report bands used for lines and spacing are removed if there are no data rows or for suppressed rows.

- **ReportRowPageBreak**: Applies a page break where appropriate for this row.
- **ReportRowContentTop**: Specifies the vertical position of the row relative to the row above it. If it is set to zero, the top of the row immediately follows the row above it with no vertical spacing.

- **ReportRowContentHeight**: The height of the row in pixels.
- **ReportRowPaddingTop** and **ReportRowPaddingBottom**: Control the extra space (padding) above and below a row in a report.
- **ReportFontSize**: The point size of the font on the report if it differs from the font size displayed in data explorer.
- **ReportTextAllignment**: Controls how the data are aligned in a report.
- **ReportUnderline**: Creates a simple underline of report values.
- Color options are available for text and background.
- **Top Lines** and **Bottom Lines**: Used for underlines and overlines. The Line1 and Line2 options can be used together to create a double underline or double overline. For the lines to have a small gap between each column, use the ReportBottomLine1PaddingLeft property or one similar.
- **Borders**: Allows a line to be drawn around the border of a cell.

Cell Format Details

The format properties for a cell include font, colors, background, and grid lines.

1. In the **Rows and Columns** slider, select the **Default** cell to apply the formatting settings to all of the cube view cells.

   > **NOTE:** You can select an individual cell instead if you want to edit the settings for a specific cell.

2. In the **Formatting** tab, click the ellipsis to the right of **Cell Format**.
3. Click **Format** to view the options. The Excel section only affects how the cube view data display in Excel.

- **Custom Parameters**: Click the ellipsis to select and assign a custom parameter such as a cube view style to the cube view cell.
- **NumberFormat**: Uses the Microsoft .NET standard number format syntax, which allows different formats to be specified for positive and negative numbers separated by a semi-colon (not available in Excel). For example:
    - #,### ;(#,###);0 would show the number using a comma as the thousands separator, no degrees of precision, parenthesis around negative numbers, and a zero for null values. The pattern of this format choice is Positive;Negative;Null values with each separated by a semicolon. After the first #,### there is a space. This enables the numbers to line up with negative numbers due to the parenthesis.
    - #,###.#% ;(#,###.#%);" – " would show a percentage with a comma as the thousands separator, one degree of precision, negative percentages in parenthesis and a dash for null values.
    - N2 would show the data as a numeric value with two degrees of precision. Negative numbers are presented with a minus sign.
    - P1 would show a percentage with one degree of precision.
- **ZeroOffsetForFormatting**: Related to NegativeTextColor, which is determined by whether a number is less than zero. For example, if sales were less than 100, rather than 0, they could be displayed in red. A valid setting is any number other than zero.
- **Scale**: -12 to +12 are the valid values for the scale. For example, to show a number in thousands, the scale should equal 3, or for millions, the scale should equal 6.
- **FlipSign**: This will flip the display value between positive and negative. This is useful for reports where certain expense numbers are stored as positive or negative and need to be shown on the report.
- **ShowPercentageSign**: Determines whether a percentage sign is displayed.

- **ShowCurrency**: Shows the currency code (for example, EUR). This is not available in Excel.
- Font and color options are available. For the FontFamily, any .Net font installed on the Excel client operating system is included in the font family. The FontSize default is 11. NegativeTextColor, WritableBackgroundColor, and SelectedGridLinesColor can optionally override the format for negative numbers, writeable data cells, and selected data cells.
- **ExcelNumberFormat**: Apply to control how numbers display when exporting to Excel. A number format can have up to four sections of code separated by semicolons. These code sections define the format for positive numbers, negative numbers, zero values, and text, in that order.

  <POSITIVE>;<NEGATIVE>;<ZERO>;<TEXT>

  For example, use these code sections to create the following custom format. Note that this format can include the underscore to create a space after the trailing positive number and can control the color of the negative number to be red:

  #,##0.00_);[Red](#,##0.00);0.00

  > **NOTE:** ExcelNumberFormat settings [#,##0,,.0 ;[Red](#,##0,,.0)] on a cube view scales numbers when the cube view is exported to Excel.

- **ExcelUseScale**: Determines if Excel uses the scale property.
- Alignment, color, and border options are available.
- **ReportNoDataNumberFormat**: By default, NoData (null) cells display as empty in the data explorer and as zeros in a report. However, any .NET number format text can be specified to format those zero values differently in the report. For example, type NODATA in that setting. To display empty text, type # in that setting. If the number format setting of the cell already does something with null values, this property does not need to be filled.

- **ReportUseNumericBinding** and **ReportNumericBindingFormat**: These are related to the export of dashboard reports to Excel, so numbers can be rendered in the proper number format rather than appear in Excel as text. Set ReportUseNumericBinding to True to use numeric amounts instead of OneStream text-based formatting when generating a cube view report. The ReportNumericBindingFormat setting must follow a specific syntax. This number format is related to the report engine, so Excel standard number formats are not used. For example, to represent a number format for positive, negative, and no values:

  {0:#,# ;(#,#);" - "}

  This feature provides the ability to generate numbers instead of text when exporting a dashboard report to Excel. However, it cannot be used for calculation status and annotation data cells because those features cannot be represented as numbers. In those cases, use column overrides to display the values as text.

- Position, font, color, top lines, bottom lines, and borders are available. See Header Format Details.

Member Filters

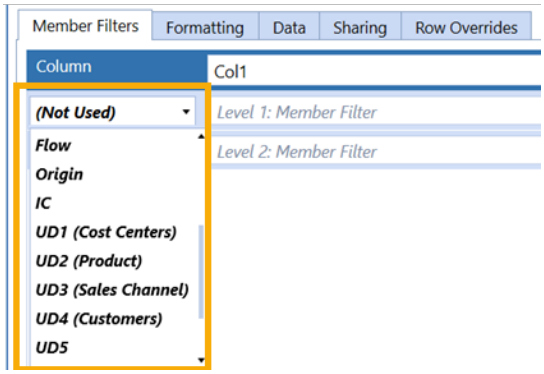
In a cube view, you can nest up to four different dimensions as rows and two different dimensions as columns.

> **NOTE:** When nesting cube view columns, if the top column header is longer, it will span the nested column headers when displayed on a cube view report.

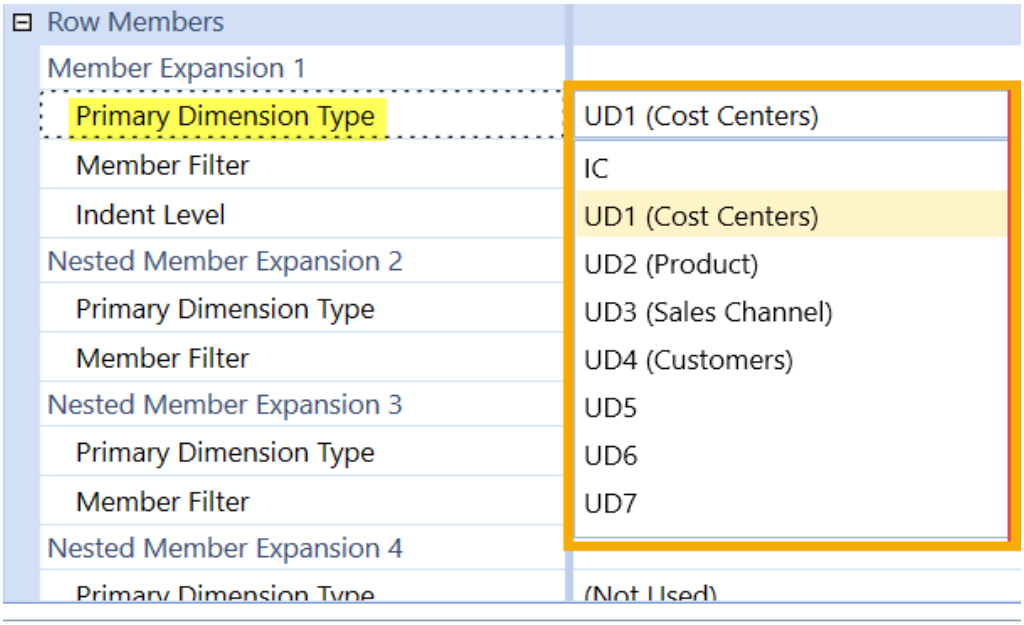
Dimension Type: Select a dimension type from the drop-down list. Indicates which dimension type will display on the selected row or column.

Member Filter: Enter a member filter to express the specific members needed from the dimension.

User Defined Description – Cube View: In the Cube View Editor, the custom user-defined descriptions will display in the Designer and Advanced tabs, in the Member Filters dimension type:



In the Advanced tab, Row and Column dimension type:



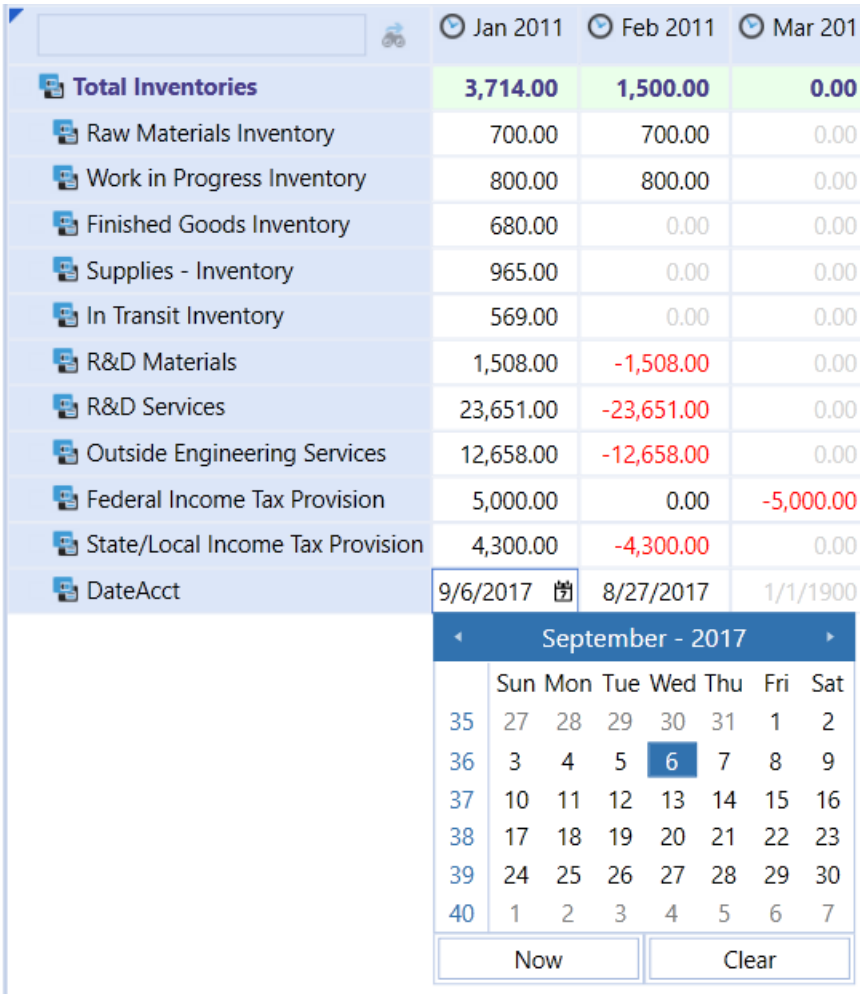
Data

Can Modify Data: If False, specific rows or columns are read-only. If True, the False setting will override it. However, if Can Modify Data is False, the entire cube view is read-only and this setting will not override if True.

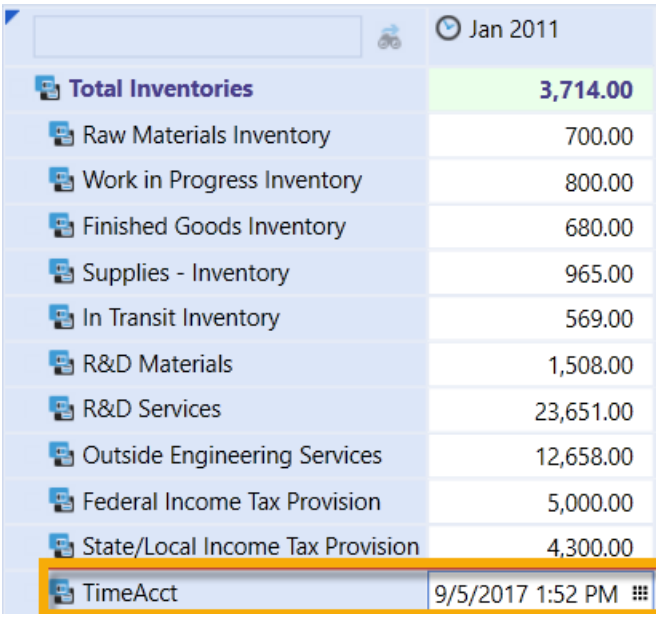
Text Box: Makes the cube view cell numerical. Only numerical values can be entered in the data cell at run time. This is the default setting.

Combo Box: Select this when a List Parameter is used in a cube view data cell. Enter the name of the List Parameter in the property below.

Date: Enables a calendar in the cube view cell and allows you to select a specific date in the data cell.



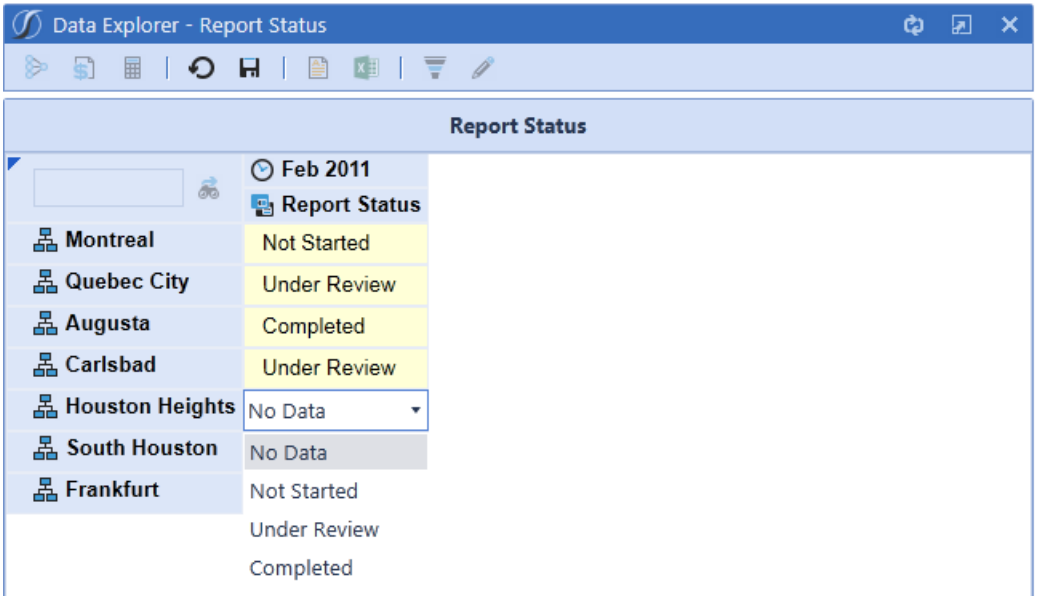
Date Time: Enables a calendar and a time display and allows you to select a specific date and time in a data cell.



	Jan 2011
Total Inventories	3,714.00
Raw Materials Inventory	700.00
Work in Progress Inventory	800.00
Finished Goods Inventory	680.00
Supplies - Inventory	965.00
In Transit Inventory	569.00
R&D Materials	1,508.00
R&D Services	23,651.00
Outside Engineering Services	12,658.00
Federal Income Tax Provision	5,000.00
State/Local Income Tax Provision	4,300.00
TimeAcct	9/5/2017 1:52 PM

NOTE: Selecting Now in the calendar enters the current date or current date and time.

List Parameter: Enter a parameter to create a drop-down list of choices in a cube view data cell.



Enable Report Navigation Link (Rows Only): Set to True to enable Navigation Link Drill Downs in dashboards. Enter a value in the Dashboard to Open in Dialog field. Set to False if this function is not used for the cube view. See Navigation Links.

Dashboard to Open in Dialog (Rows Only): Name of the dashboard or parameter used for the Report Navigation Link. Use a parameter for a cube view acting as a row template across multiple cube views.

Linked Cube Views: Enter a comma-separated list of cube views that will be available to open when viewing a cube view in the Data Explorer Grid. The cube views specified in this field apply to the selected row or column and are available when you right-click on any data cell in the specified row/column.

> **NOTE:** Row settings override column settings which override cube view settings.

Rename Rows and Columns

By default, a cube view has standard headers to represent rows and columns. All the header settings control the row and column headers presented when the cube view is run if the default is not used. See General Settings.

If the header name for a row or column must be different than the dimension name or description, use the rename function. The rename function is an expansion of the end of a member filter that enables you to enter what you want displayed. For example, A#CashBalance:Name(Cash Balance) will display Cash Balance as its header.

Another way to use the rename function is to include the XFMemberProperty, which returns member properties. For example:

A#6009.base:Name(XFMemberProperty(DimType=Account, Member=|MFAccount|, Property=AccountType))

1. Use the rename function to pull in the member property: **Name(XFMemberProperty**
2. Define the dimension type: **(DimType=Account,**
3. Determine what will be in the member filter: **Member=|MFAccount|,**
4. Pull in the property: **Property=AccountType))**

Create Conditional Formatting

Conditional formatting provides the ability to format headers or cells based on defined criteria. Conditional formatting in cube views supports data analysis by highlighting cells or ranges of cells, identifying key values, and using data bars, color scales, and icon sets that correspond to specific variations in the data. Conditional formatting is applied and visible in data explorer, Excel, and report views. The conditional formatting criteria are applied to cube view rows or columns on the formatting property within the cube view editor.

Conditional formatting is available on the formatting elements of a cube view.

- Cube view default formatting
- Row and column headers
- Row and column data cells
- Row and column overrides

Conditional formatting follows the cube view processing order of operations:

1. Number format as defined in application properties
2. Cube view default
3. Column formatting
4. Row formatting

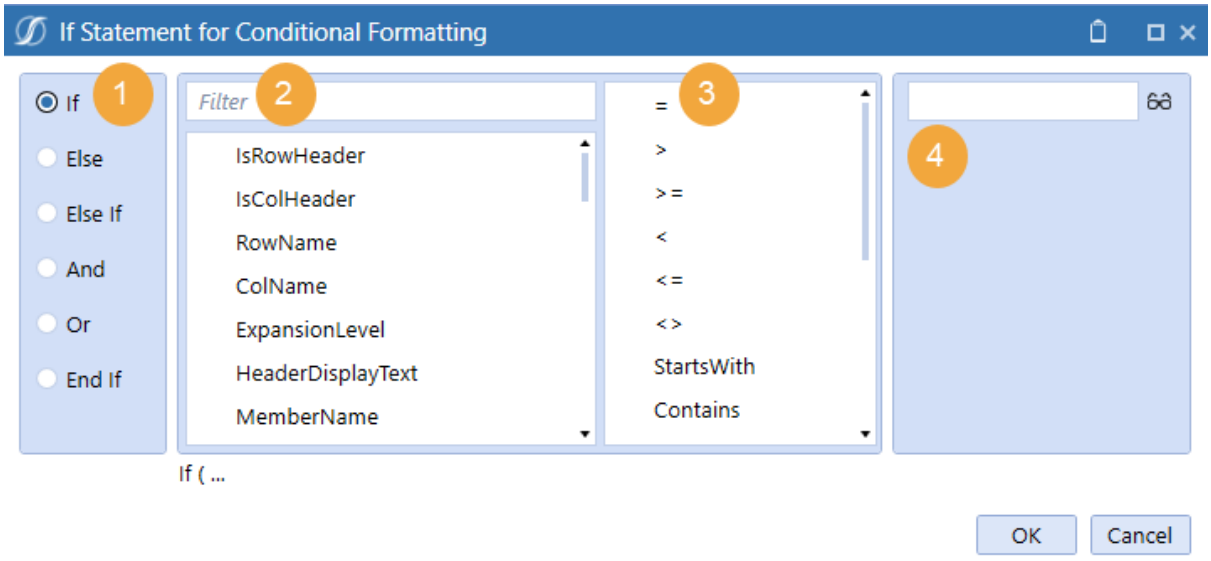
5. Column overrides
6. Row overrides

The combination of formats and overrides equals the format for the cell when rendered. Formatting can be applied and isolated to the nested expansion levels on rows and columns by using property filters in the dialog box. This feature is a cube view formatting option.

> **NOTE:** If there are any existing formats before applying conditional formatting, they will be retained if the range of cells containing the conditional formats does not meet the conditions of the rule. All styles from the cube view and the selection styles that had been applied to that range are overridden by conditional formatting.

Conditional Formatting Properties

General and conditional formatting can be applied in the Header Format and Cell Format dialog boxes. The Condition and Format buttons launch additional dialog boxes with available options. The dialog box for conditional formatting is divided into sections.



1. **Conditional statement**: Defines the options for conditional statements to apply constraints to formatting.
2. **Filter**: Applies the criteria that pertain to your statement.
3. **Operator**: Contains criteria filters to use in your conditional statement.
4. **Text/object**: Find the text or object to be tested.

Header Property Filters

Header Property Filter	Description
IsRowHeader	Boolean, determines if an object is a row header field (appropriate for Cube View Default).
IsColHeader	Boolean, determines if an object is a column header field (appropriate for Cube View Default).
RowName	Cube view row name
ColName	Cube view column name
ExpansionLevel	Determines the expansion level for rows (1-4) and columns (1-2). Related to isolating the row expansion headers.
HeaderDisplayText	Custom descriptions used with the :Name() function
MemberName	Metadata member labels
MemberDescription	Metadata member descriptions

Header Property Filter	Description
MemberShortDescription	Metadata member short descriptions (only Time dimension)
IndentLevel	Indentation level derived as a formatting setting or that is system-generated based on tree expansions

Expansion Specific Property Filters

Expansion Specific Property Filter	Option	Description
RowE expansion level - criteria	Row expansions 1–4	Identify nested or expansion rows with specified criteria
ColE expansion level - criteria	Column expansion 1 and 2	Identify nested or expansion columns with specified criteria

Cell Format Property Filters

Cell Format Property Filter	Description
IsNoData	Test for no data
IsRealData	Test for stored data, ignoring derived Zero-View data
IsDerivedData	Test for derived data, commonly resulting from Scenario Zero-View settings

Cell Format Property Filter	Description
IsRowNumberEven	Test for the row number as an expansion or on a fixed row
ExpandedRowNum	Test the count of expanded rows, Zero-based. This property filter is based on the total cube view count of rows generated from each row and its expansions.
CellAmount	Test cell data amount
CellStorageType	Test the method used to store data

Define Properties for Conditional Formatting

Conditional formatting can be applied based on a data point. Cell formatting conditions can occur based on member names or descriptions, indent levels, row and column names, and expanded and even numbered rows.

Conditional formatting can also be applied based on members. The ability to format based on the name or description is enhanced by using a standardized metadata naming convention. Summary level members with keywords such as “Total” or prefixes or suffixes such as “Tot” could be used in conditional formatting. Members can be formatted using dynamic criteria, such as StartsWith or EndsWith. HeaderDisplayText differs from the MemberName and MemberDescription property filter because it references the custom name parameter in a member filter. Dynamic criteria can be applied to the name and description property filters to apply the required formatting.

1. On the **Cube Views** page, under **Cube View Groups**, select a cube view.
2. Select **Rows and Columns** to expand the slider.

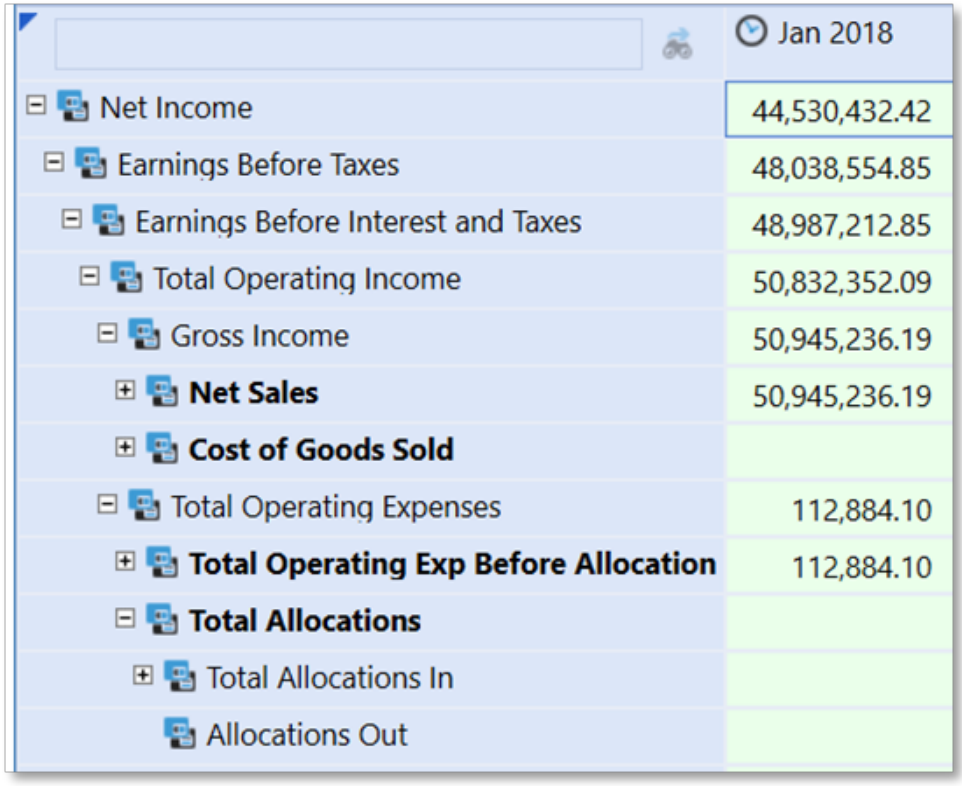
3. Select a row or column.
4. Click the **Formatting** tab.
5. Click the ellipsis to launch the dialog box.
6. Click **Condition**, set the requirements, and then click **OK**.
7. Click **Format**, set the requirements, and then click **OK**.
8. Click **Condition** and then **End If** to close the condition.
9. Click **OK** twice.

Examples Applying Conditional Formatting

Below are common usage examples of conditional formatting.

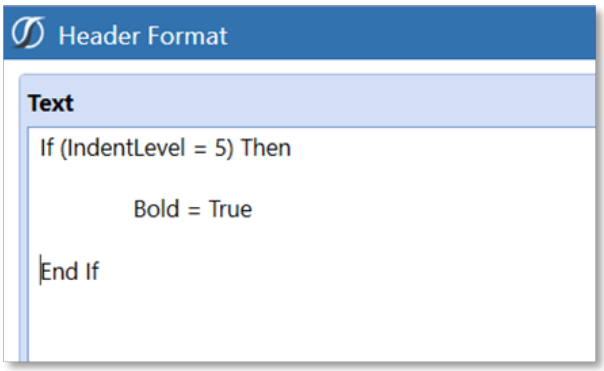
Indent Level

The IndentLevel Property filter will dynamically format from defined rows or expansions. Indentation is zero-based. The formatting can be applied to the default or to rows. This solution can speed formatting for summary level dimension members.



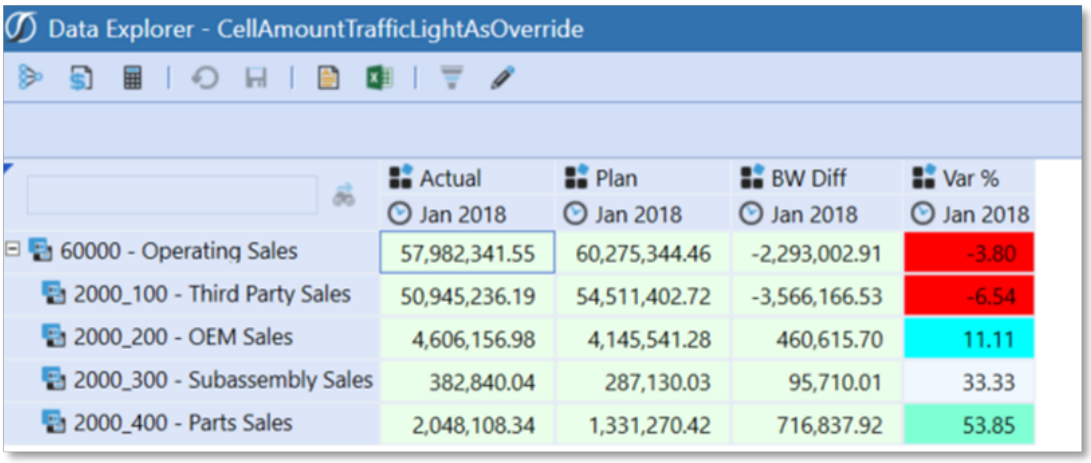
	Jan 2018
Net Income	44,530,432.42
Earnings Before Taxes	48,038,554.85
Earnings Before Interest and Taxes	48,987,212.85
Total Operating Income	50,832,352.09
Gross Income	50,945,236.19
Net Sales	50,945,236.19
Cost of Goods Sold	
Total Operating Expenses	112,884.10
Total Operating Exp Before Allocation	112,884.10
Total Allocations	
Total Allocations In	
Allocations Out	

Apply the formatting to the cube view's Header section or to specific rows.



Examples of Using Conditional Property Filters Conditional "Traffic-Lighting"

"Traffic-Lighting" is data related and therefore applied as a cell format. The designer has a choice to apply the conditional formatting to either a row or a column. The order of operations for formatting can impact the decision. Row overrides are the final layer of formatting applied to a cube view and would not be impacted by other more general formatting.



	Actual Jan 2018	Plan Jan 2018	BW Diff Jan 2018	Var % Jan 2018
60000 - Operating Sales	57,982,341.55	60,275,344.46	-2,293,002.91	-3.80
2000_100 - Third Party Sales	50,945,236.19	54,511,402.72	-3,566,166.53	-6.54
2000_200 - OEM Sales	4,606,156.98	4,145,541.28	460,615.70	11.11
2000_300 - Subassembly Sales	382,840.04	287,130.03	95,710.01	33.33
2000_400 - Parts Sales	2,048,108.34	1,331,270.42	716,837.92	53.85

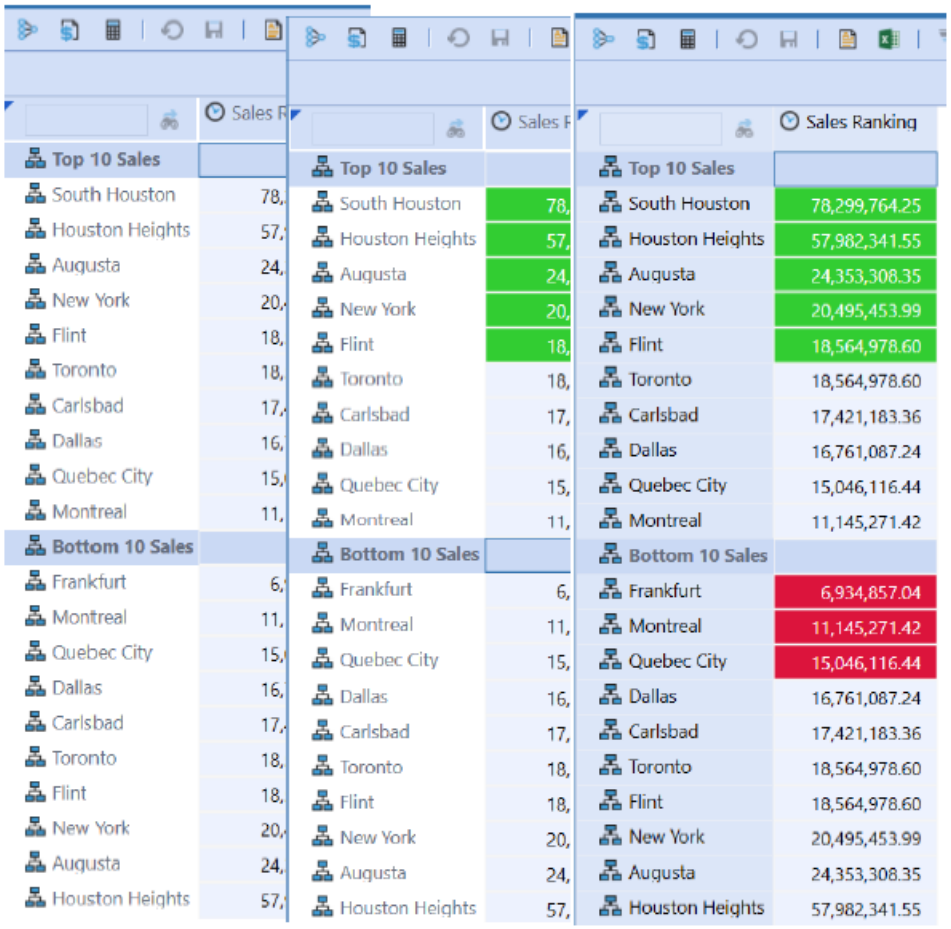
1. Conditional formatting can be applied to the rows or columns. A definition applied to the row Formatting tab would apply to all columns. A row override would isolate the formatting to a specified column(s).

2. The CellAmount filter is used within multiple If/ElseIf statements to define the various tests required for the report.

ExpandedRowNum Expansion Range

Conditional formatting provides the ability to format an expanded range of data cells using the ExpandedRowNum Property Filter. This is useful in formatting to support “Top 10” type reports. This filter best supports cube view designs that use known expansions, such as ranking business rules or member lists. If applied to specific rows, the formatting applies to the defined row, but the Expansion Number Reference relates to the entire cube view. Formatting defined on subsequent rows will be impacted if the expansion members change on previous rows. The ExpandedRowNum Property

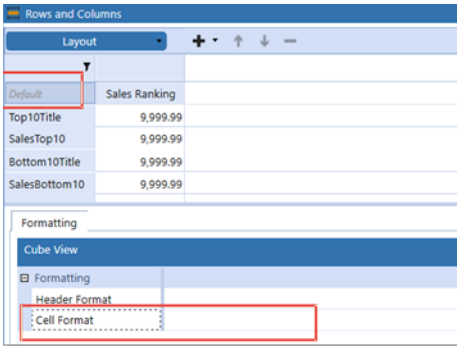
Filter can also be applied as a default cube view format, in which the definition will apply to all rows. This would require If/ElseIf type statements to support all rows.



Top 10 Sales	Sales R	Top 10 Sales	Sales R	Top 10 Sales	Sales Ranking
South Houston	78,	South Houston	78,	South Houston	78,299,764.25
Houston Heights	57,	Houston Heights	57,	Houston Heights	57,982,341.55
Augusta	24,	Augusta	24,	Augusta	24,353,308.35
New York	20,	New York	20,	New York	20,495,453.99
Flint	18,	Flint	18,	Flint	18,564,978.60
Toronto	18,	Toronto	18,	Toronto	18,564,978.60
Carlsbad	17,	Carlsbad	17,	Carlsbad	17,421,183.36
Dallas	16,	Dallas	16,	Dallas	16,761,087.24
Quebec City	15,	Quebec City	15,	Quebec City	15,046,116.44
Montreal	11,	Montreal	11,	Montreal	11,145,271.42
Bottom 10 Sales		**Bottom 10 Sales**		**Bottom 10 Sales**	
Frankfurt	6,	Frankfurt	6,	Frankfurt	6,934,857.04
Montreal	11,	Montreal	11,	Montreal	11,145,271.42
Quebec City	15,	Quebec City	15,	Quebec City	15,046,116.44
Dallas	16,	Dallas	16,	Dallas	16,761,087.24
Carlsbad	17,	Carlsbad	17,	Carlsbad	17,421,183.36
Toronto	18,	Toronto	18,	Toronto	18,564,978.60
Flint	18,	Flint	18,	Flint	18,564,978.60
New York	20,	New York	20,	New York	20,495,453.99
Augusta	24,	Augusta	24,	Augusta	24,353,308.35
Houston Heights	57,	Houston Heights	57,	Houston Heights	57,982,341.55

1. Design or open a report which supports formatting for ExpandedRowNum , such as a ranking report in the example.

2. Determine how the formatting should be applied, as a default, row or column. The example will use the cube view default formatting.



3. Cell format is applied for conditional formatting using the ExpandedRowNum filter. Being zero-based, and having to account for each row and its expansions:

a. Condition1 – Rows < 6 is defined because the text header row “Top10Title” initiates the count at zero.

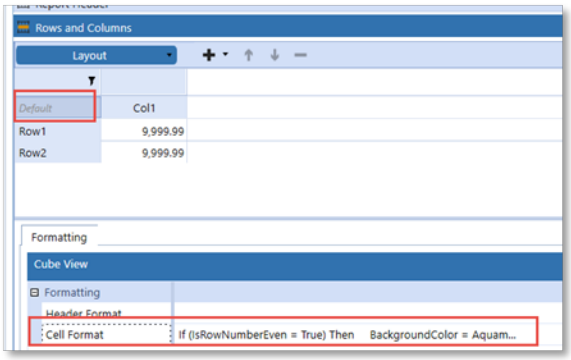
b. Condition2 – Rows > 11 and < 15 is defined because all the rows up to Row4, “SalesBottom10”, reflect rows 0-11 on the cube view. The conditional row references must reflect all the cube view expansions.

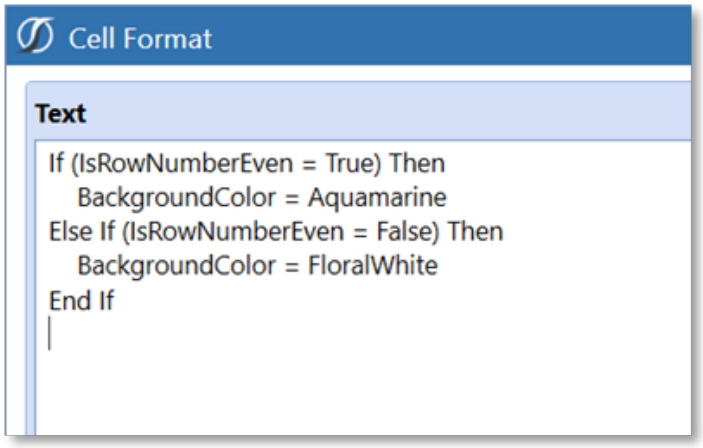
Test for Row IsRowNumberEven

The IsRowNumberEven property filter may be useful to vary formatting by the even/odd numeric expansion of rows on a cube view. For example, to replicate a “green-bar” style report the IsRowNumberEven would be a suitable filter.

	Jan 2018	Feb 2018	Mar 2018	Apr 2018	May 2018
Net Income	44,530,432.42	0.00	0.00	0.00	0.00
Earnings Before Taxes	48,038,554.85	0.00	0.00	0.00	0.00
Earnings Before Interest and Taxes	48,987,212.85	0.00	0.00	0.00	0.00
Total Operating Income	50,832,352.09	0.00	0.00	0.00	0.00
Gross Income	50,945,236.19	0.00	0.00	0.00	0.00
Net Sales	50,945,236.19	0.00	0.00	0.00	0.00
Operating Sales	50,945,236.19	0.00	0.00	0.00	0.00
Third Party Sales	50,945,236.19	0.00	0.00	0.00	0.00
Total Operating Expenses	112,884.10	0.00	0.00	0.00	0.00

In this example the formatting can be applied to the cube view default cell format since it will globally apply to all rows.

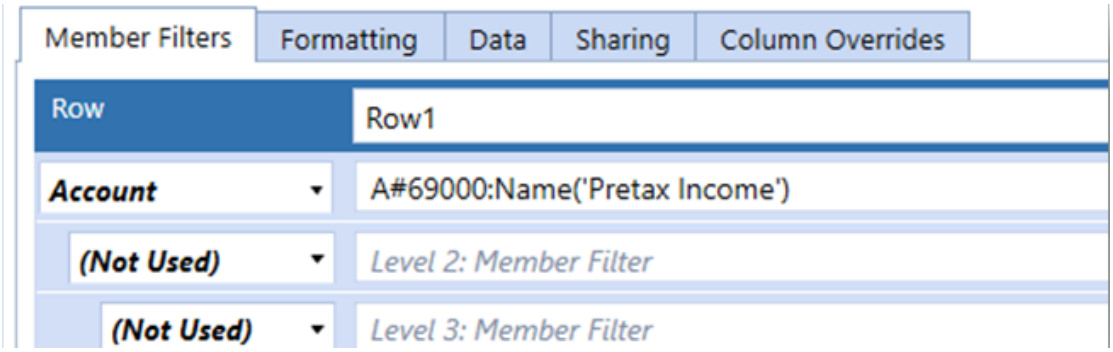




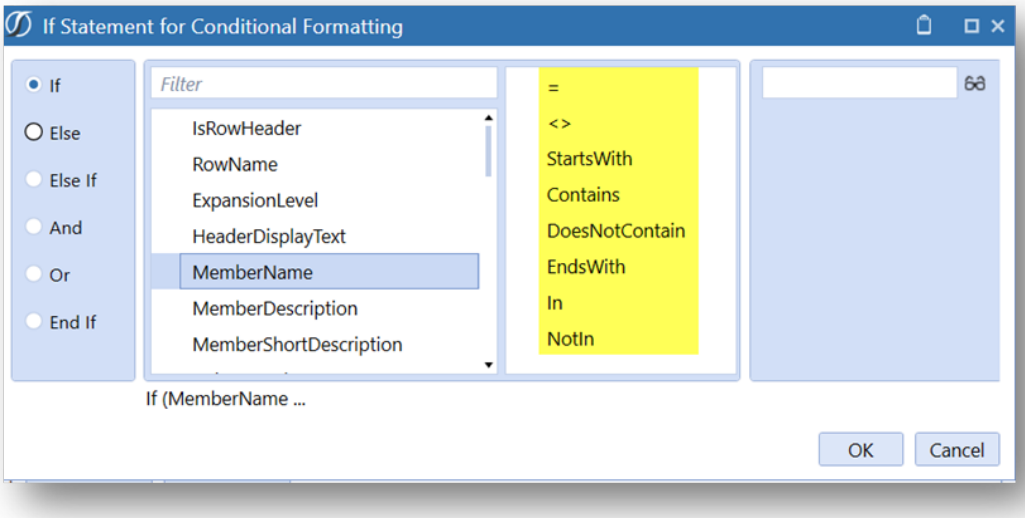
MemberName, MemberDescription or HeaderDisplayText

The ability to format based on the Name or Description is greatly enhanced by adhering to a standardized metadata naming convention. Summary level members having keywords such as “Total” or prefixes or suffixes such as “Tot” could be used in conditional formatting. Members could be formatted using dynamic criteria, such as StartsWith or EndsWith.

HeaderDisplayText differs from the MemberName and MemberDescription Property filter in that it references the custom Name parameter in a Member Filter.



Dynamic criteria can be applied to the Name and Description property filters to apply the required formatting.



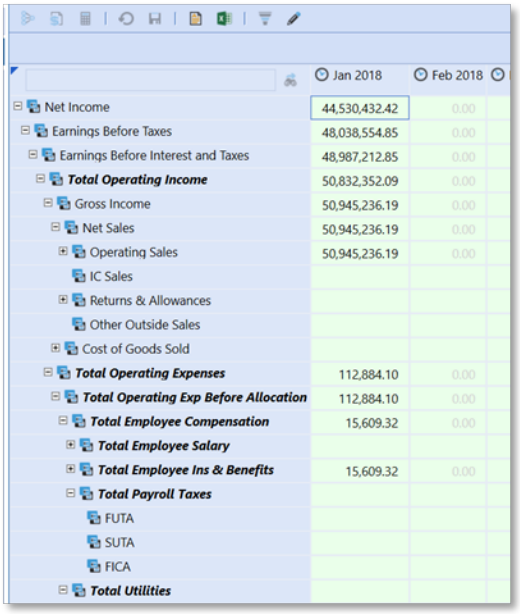
Apply conditional formatting to row header.

Header Format

Text

```
If (MemberDescription StartsWith 'Total') Then
   Bold = True, Italic = True
End If
```

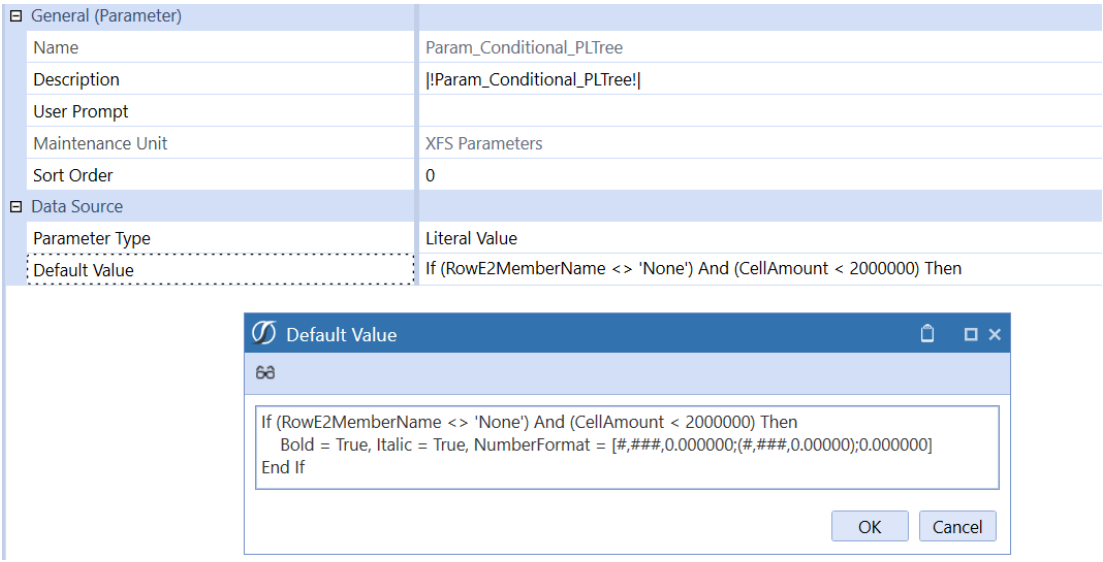
The result is a dynamically formatted report.

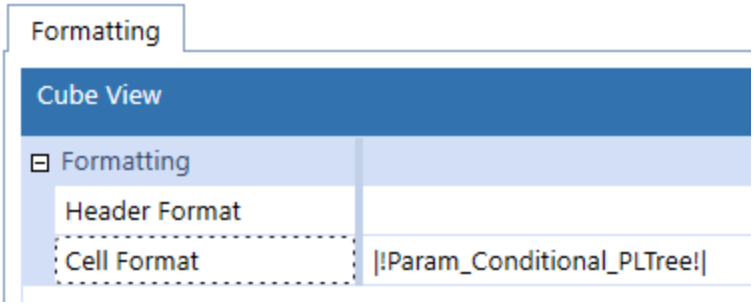


	Jan 2018	Feb 2018
Net Income	44,530,432.42	0.00
Earnings Before Taxes	48,038,554.85	0.00
Earnings Before Interest and Taxes	48,987,212.85	0.00
Total Operating Income	50,832,352.09	0.00
Gross Income	50,945,236.19	0.00
Net Sales	50,945,236.19	0.00
Operating Sales	50,945,236.19	0.00
IC Sales		
Returns & Allowances		
Other Outside Sales		
Cost of Goods Sold		
Total Operating Expenses	112,884.10	0.00
Total Operating Exp Before Allocation	112,884.10	0.00
Total Employee Compensation	15,609.32	0.00
Total Employee Salary		
Total Employee Ins & Benefits	15,609.32	0.00
Total Payroll Taxes		
FUTA		
SUTA		
FICA		
Total Utilities		

Parameter Formatting

Conditional formatting definitions can be applied to a cube view as a Literal parameter. To apply, select the format string from the cube view, press Ctrl-C, select the Default value property of a Literal Value-type parameter and press Ctrl-V. Once saved, this parameter can be referred to in a cube view format. As shown, the reference of |!Param_Conditional_PLTree!| in a cube view format would apply the associated format string.





General Settings

This section explains how to view, define, and apply standard report setting properties. These settings determine the information you see and the actions you can take in the cube view. To edit the following properties, on the **Cube Views** page, under **Cube View Groups**, select a cube view and then **General Settings** to expand the slider.

- **Sharing**: Share all or specific rows or columns from another cube view. Sharing can increase efficiency and consistency when building and managing cube views.
- **Common**: These options control general, shortcut, restrictions, suppression, and paging properties.
    - **Is Visible in Profiles**: When set to True, the Cube View is visible in the Cube View Profile. When false, the Cube View is not visible within the Cube View Profile. In these cases, the Cube View would not be visible in Workflow, Excel, Dashboards, or OnePlace.
    - **Page Caption**: Appears at the top of the Data Explorer grid when viewing the output of a cube view. If left blank, it will get its value from the cube view description. If the description is blank, it will display the cube view name.
    - **Is Shortcut**: Determines whether the cube view is a shortcut.
    - **Shortcut Cube View Name**: Enter the name of another cube view that will open when this cube view is being used as a shortcut.
    - **Is Dynamic**: Set this property to True to make the Cube View dynamic. When set to True, the developer can programmatically manipulate Cube View formatting and data at runtime through the Dynamic Cube View Service that can be added to the Workspace Assembly.
    - **Literal Parameter Values**: Enter a comma-separated list of name-value pairs for parameter values to be used when this cube view is used as a shortcut to open another cube view. Piped variable names cannot be used (for example, |!ParamView!|) in this setting.

      Example: Param1=Value1, Param2=[Value 2]
    - **Can Modify Data**: If set to True, the cube view can be modified in OnePlace. If set to False, the cube view is read only.

> **NOTE:** You can still make annotations to accounts in a cube view if this property is set to False.

- **Can Calculate**, **Can Translate**, and **Can Consolidate**: If set to True, you can right-click on a cell and calculate, translate, or consolidate data in OnePlace.

  > **NOTE:** These properties coincide with the Can Calculate from Grids Scenario security group. If set to True but the user is not in the group for that given scenario member, they cannot calculate from a cube view grid.

- **Can Modify Suppression**: Set to True to enable a row suppression icon in the data explorer grid toolbar and enable users to turn the suppression settings on and off when viewing a cube view.
- **Include Cell Attachment Status**: If set to True, a cell containing a data attachment will display a red tick mark. If set to False, there will be no indication of a data attachment in the cell. In large data sets, if set to True, it can affect performance.

  > **NOTE:** This setting only applies to Cube Views in Spreadsheet and the OneStream Excel Add-In. It does not currently apply to Quick Views.

- See Sparse Row Suppression and Cube View Paging.

- **Header Text**: The default is a cube view with standard row and column headers. All the header settings control the row and column headers presented when the cube view is run if the default is not used.
- **Header Size**: Column Header Heights align column headers with the bottom of a report. The default setting is -1 (which means they are auto-sized), so for bottom alignment to work in column headers, the height of the column header needs to be set. This is helpful for wrapped headers, which may cause the columns to expand.

Row Header Widths are viewed in the cube view. Excel Row Header Widths are viewed when exporting to or rendering a cube view in Excel. Report Header Widths are viewed when a report on-the-fly is generated from a cube view. These can be changed from the default value. The -1 default value means that these Row Header Widths are auto-sized. Otherwise, set a positive number of at least 100 to determine the number of pixels wide (96 pixels = 2.5 cm). These settings can change for a maximum of six nested rows. The same can be done when reporting to Excel by using the ExcelRowHeader1Width property. The default value for this is also -1.

> **NOTE:** If a cube view Row Header Width is not specified, text and font information from all the expanded headers is used to automatically determine the width. The automatic row width maximum is a half of a page. Any text longer than a half of a page is wrapped.

- **Header Overrides**: By default, the software determines which dimensions to show for row and column headers based on the member expansions specified in the Rows and Columns slider. These dimensions can also be selected manually.

  Report Column Index for Row Headers can be used for a report to display the row headers after the column instead of on the left side. The default value is -1, but setting this to a positive number changes the cube view column index. Specifying several columns greater than the existing number of columns reverts the row headers back to the left side of the report. Hidden columns are not included in the column count.

  > **NOTE:** If a cube view report exceeds the width of the page, the software automatically adds a page break on the appropriate column and repeats the row headers on the following page. If Report Column Index for Row Headers or Auto Fit to Page Width are enabled, the report does not repeat the row headers.

- **Report**: These properties apply advanced formatting to cube views.

- **Custom Report Task**: Custom report formatting can be applied to cube views using a cube view extender business rule or inline formula.
  - Select **No Task** if there are no cube view business rules or inline formulas running on this report.
  - Select **Execute Cube View Extender Business Rule** to call a cube view extender.
  - Select **Execute Cube View Extender Inline Formula** to include an inline formula.
- **Business Rule**: This property is only enabled when **Execute Cube View Extender Business Rule** is selected as the custom report task. Click the ellipsis and select the cube view extender business rule.
- **Formula**: This property is only enabled when **Execute Cube View Extender Inline Formula** is selected as the custom report task. Click the ellipsis and select the cube inline formula.

Additional options are available for report formatting to set paper size, page orientation, and margins, and to automatically fit the information to a specific page width or number of pages.

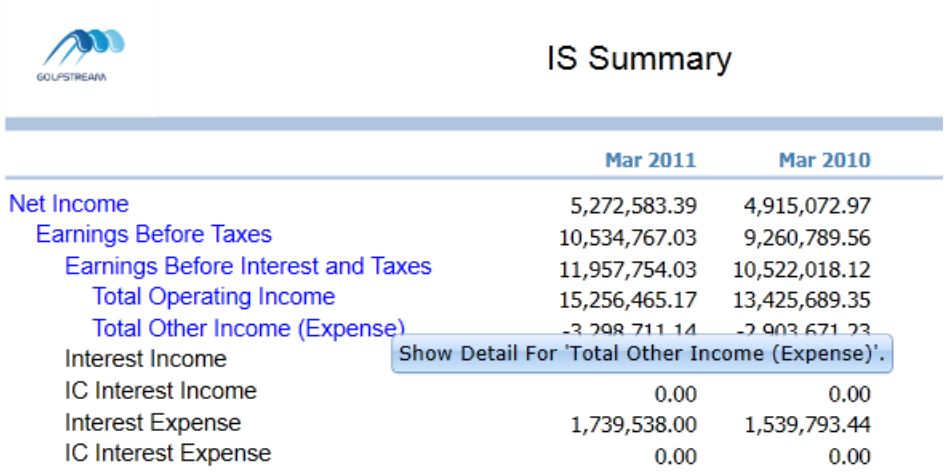
- **Excel**: Controls whether gridlines are displayed in the Excel file.
- **Navigation Links**: Determines the information you can view in a cube view in the data explorer grid.
  - **Linked Cube Views**: Enter a comma separated list of cube views that will be available to open when viewing a cube view in the data explorer grid. The cube views specified in this field apply to the entire cube view and are available when you right-click any cube view data cell. See Linked Cube Views.

- **Linked Dashboards**: Enter a comma separated list of dashboards that you can right-click to open when viewing a cube view in the data explorer grid.
- **Include Default NavLink Parameters**: Set to **True** in source cube views to use NavLink parameters - with a intuitive, standard syntax, as Bound parameters for any dimension. This simplifies cube view design and reduces error as you:
  - Do not have to define parameters for navigation, and can use a standard syntax to ensure parameter names match across cube views and other objects.
  - Can navigate cube views from any dimensional intersection to analyze data that is refined by passed point of view selections, in linked cube views. See Analyze Data in Linked Cube Views Wtih Passed Point of View Selections.
- **Bound Parameter Names for Navigation Links**: Lists all the dimensions that may be assigned to a row when the **Report Navigation Links** property is set to **True** and represents the member name being passed from this report to another using navigation links.
- **Bound Parameter Names**: For a linked cube view to open and display the correct member data for a selected cell, parameters with the member need to be specified using a bound parameter. This lists all the dimensions that may have a parameter included in a linked cube view.

Navigation Links

Navigation links let you set up one dashboard to launch another in order to drill into more detail or related detail on a certain row of data from a report, while other times it can create a chart of the reviewed data.

The following information provides a detailed example including the steps one must take to create Navigation Links. The implementation of this feature will differ, but for this example an Income Statement Summary Dashboard will launch another Dashboard with more detail:

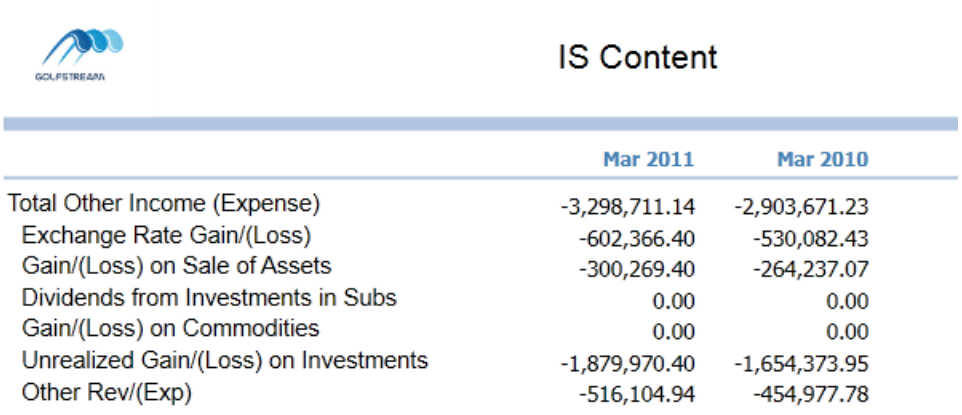


IS Summary

	Mar 2011	Mar 2010
Net Income	5,272,583.39	4,915,072.97
Earnings Before Taxes	10,534,767.03	9,260,789.56
Earnings Before Interest and Taxes	11,957,754.03	10,522,018.12
Total Operating Income	15,256,465.17	13,425,689.35
Total Other Income (Expense)	-3,298,711.14	-2,903,671.23
Interest Income		
IC Interest Income	0.00	0.00
Interest Expense	1,739,538.00	1,539,793.44
IC Interest Expense	0.00	0.00

Show Detail For 'Total Other Income (Expense)'.

Click **Total Other Income (Expense)** to display the following dashboard:

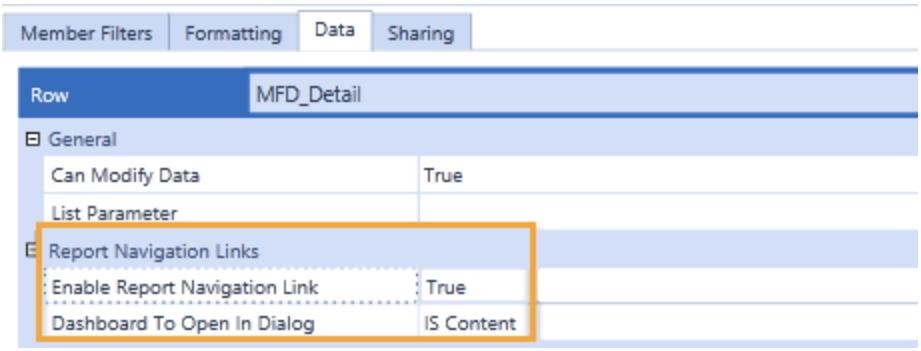


IS Content

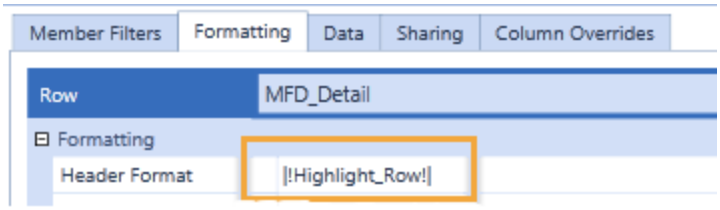
	Mar 2011	Mar 2010
Total Other Income (Expense)	-3,298,711.14	-2,903,671.23
Exchange Rate Gain/(Loss)	-602,366.40	-530,082.43
Gain/(Loss) on Sale of Assets	-300,269.40	-264,237.07
Dividends from Investments in Subs	0.00	0.00
Gain/(Loss) on Commodities	0.00	0.00
Unrealized Gain/(Loss) on Investments	-1,879,970.40	-1,654,373.95
Other Rev/(Exp)	-516,104.94	-454,977.78

NOTE: This only works in dashboard mode.

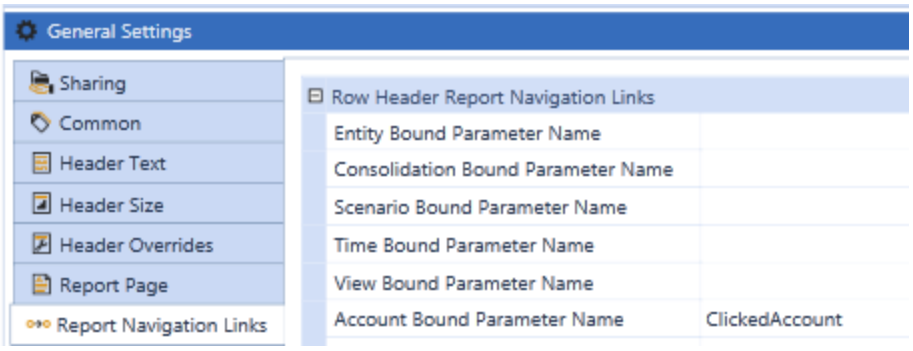
Below is a sample cube view called **IS Summary** which contains income statement summary accounts. First, select the row to be highlighted for navigation. Set Enable Report Navigation Link to True and type the name of the dashboard to open in the Dashboard to Open in Dialog field. The specified dashboard (not a cube view) opens when this highlighted row is selected from this cube view's related dashboard.



This next step is optional. Add a header format using a cube view style. This displays the row in a color different from the other rows to indicate the row is clickable. In this case, |!Highlight_Row!| is related to a parameter that was made in a sample application. When this cube view is run, the format of TextColor=Blue activates for this row.

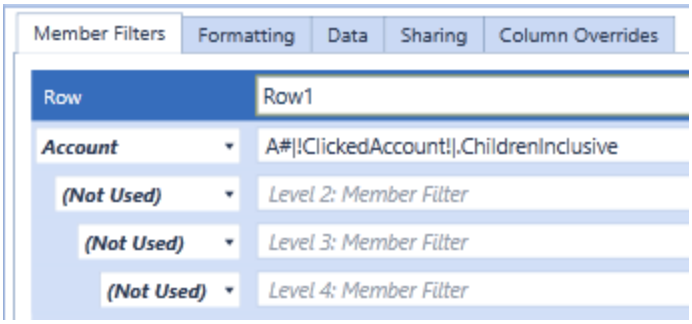


The cube view passes the clicked account to the other dashboard and there the account and its children can be viewed. Now, determine how the account is to be passed. Go to the General Settings slider and go to the Report Navigation Links section. A value of `ClickedAccount` was entered for the Account Bound Parameter Name which is going to be passed from one dashboard to another. More than one dimension can be passed if it is defined in rows, but in this case this does not need to happen.

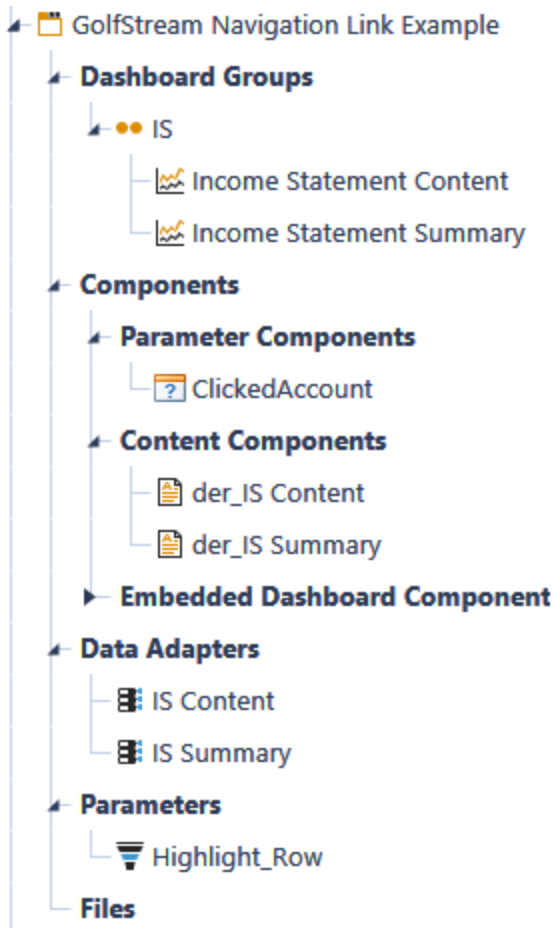


Copy this cube view to have the same columns and POV settings for the drilled cube view containing the contents. This one is called IS Content because it contains the content of the drilled data. Whatever it is called, be consistent and come to an agreement with the project team.

In the new cube view (called IS Content in this case), remove the rows that were there and add the rows that will be seen when this Dashboard appears. In this case, refer to the parameter for ClickedAccount and add the ChildrenInclusive extension.



Under dashboards, a maintenance unit was created and called GolfStream Navigation Link Example, which stores all the objects needed to get this to work properly.



Starting from the bottom of the example above is the parameter needed to highlight the row.

General (Parameter)	
Name	Highlight_Row
Description	
User Prompt	
Maintenance Unit	GolfStream Navigation Link Example
Sort Order	0
Data Source	
Parameter Type	Literal Value
Default Value	TextColor=Blue
Result Format String Type	Default
Result Custom Format String	

The data adapters are required to point to the cube views. First, set up one for IS Content. Set the Command Type to Cube View and set the cube view to IS Content.

Next, set up the data adapter for IS Summary. Set the Command Type to Cube View and set the cube view to IS Summary. Set the **Include Row Navigation Link** property to True to drill from Cube Views in a Dashboard.

Include Member Details	False
Include Row Navigation Link	True
Include HasData Status	True

Now, set up two content components for a dashboard. Namethese with the prefix of `der_` because they are data explorer reports and not charts or another form of component. Again, come up with a naming schema with the project team, so the result is organized. In this case, click to create a component and chose Data Explorer Report as the type. Do this for both IS Summary and IS Content and click in the dashboard toolbar to attach the appropriate data adapter to each.

Select Parameter Components and create a new component with a type of Supplied Parameter. In the Bound Parameter field, enter the name of the parameter being passed, which is ClickedAccount in this example. Do this for each parameter being passed from dashboard to dashboard.

Create a dashboard group (called IS in this example) and two dashboards. Create one for the launching dashboard (for example, IS Summary) and one for the launched dashboard (for example, IS Content). In this example, a layout of Uniform is being used, but the use of this feature may vary.

Under Dashboard Components, attach the data explorer report component. Do this for both IS Summary and IS Content. For just the IS Summary (the launched dashboard), also attach the parameter component being passed (ClickedAccount in this example). This allows that account to be passed from the initial dashboard to the other.

Linked Cube Views

Linked cube views provide the option to launch a cube view from another when viewing it in the Data Explorer grid, spreadsheet tool, or Excel Add-In. You can right-click a cube view's data cell and open a separate cube view to access more detail and visibility. This applies to any cube view, such as those for data entry forms. See:

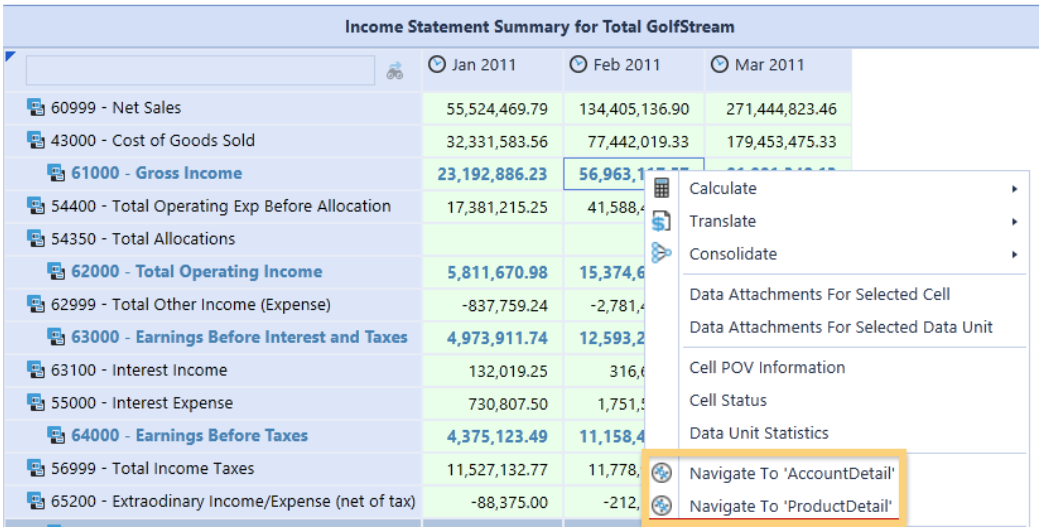
- Create a sample linked cube view
- Link a cube view to another entire cube view
- Link a cube view to specific rows or columns
- Nest linked cube views

> **TIP:** For information about evaluating source cube view data in a linked cube view using passed point of view selections, see Analyze data with passed point of view selections.

> **NOTE:** For Linked Cube Views, the Workspace Name will not be automatically prefixed. The search order of operations (requested workspace, current workspace, default workspace, and then shared workspaces) will be used to find the correct Cube View.

Create a Sample Linked Cube View

The following describes how to create a sample linked cube view. In this example, an Income Statement Summary cube view has an Account Detail cube view assigned to all data cells and a Sales by Product cube view linked to specific rows. You can also add a second level of linked cube views for an even more granular analysis.



Link a Cube View to an Entire Cube View

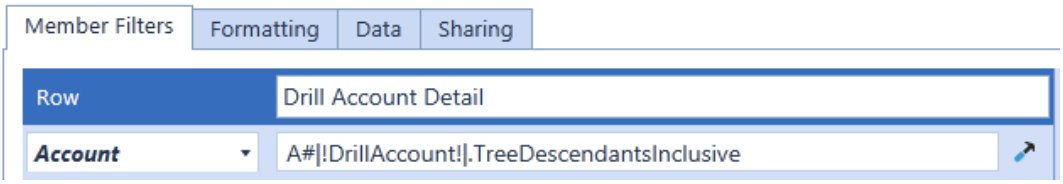
1. Select the cube view to be linked and apply to the entire cube view, specific columns, or specific rows. If the linked cube view applies to the entire cube view, you can right-click any data cell in the Data Explorer grid, spreadsheet tool, or Excel Add-In to access the cube view. If it only applies to columns or rows, the linked cube view is available when you right-click on a data cell in that row or column.

> **NOTE:** Column settings override cube view settings and row settings override column settings.

2. Once the base cube view is determined, in this example IncomeStatementSummary, create a cube view that will display more detailed data.

   The AccountDetail cube view will display Account details based on the selected data cell. The IncomeStatementSummary's data displays Parent accounts for a specific time, so the AccountDetail's data will display the selected Account's children for that same time period.

3. In the AccountDetail cube view, configure the columns and rows to query the correct data.

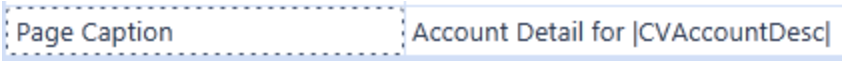


   |!DrillAccount!| is a Bound parameter and will display the correct account data based on the selected account in the IncomeStatementSummary cube view.

   !DrillTime!| is a Bound parameter and will display the correct Time data based on the selected time in the IncomeStatementSummary cube view

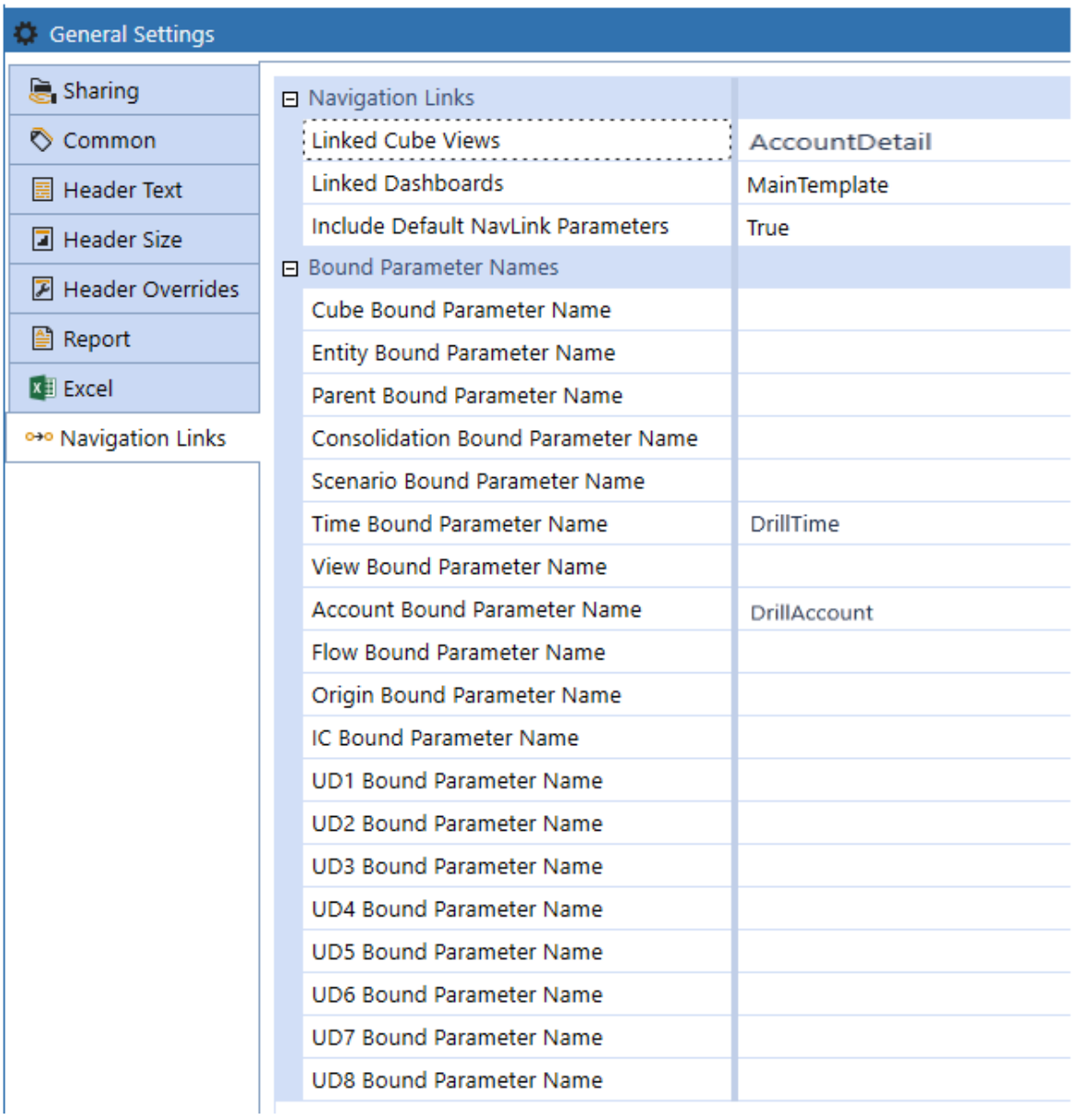
4. Ensure the AccountDetail's POV matches the IncomeStatementSummary's POV with the exception of Account and Time. Enter the |!DrillAccount!| and |!DrillTime!| parameters into these member fields to ensure that the AccountDetail cube view displays data based on the selected Account and Time in the IncomeStatementSummary.

5. Add any required formatting to the AccountDetail cube view. In this example, the following was added to Page Caption:

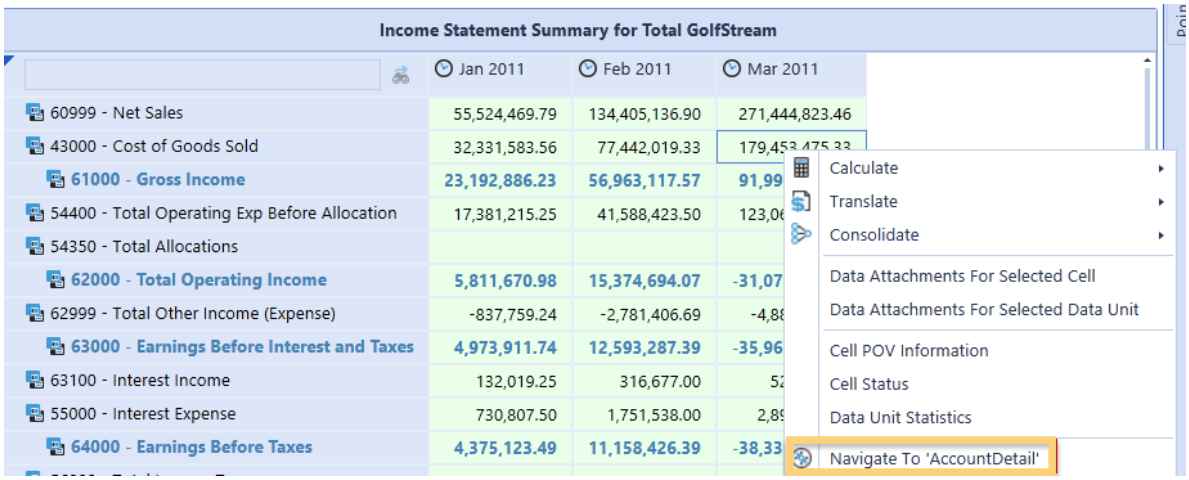


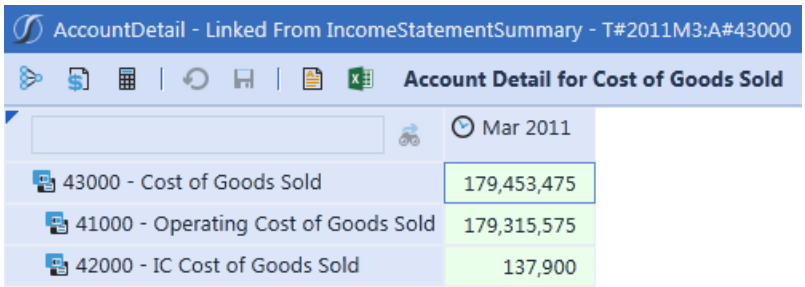
6. To assign the AccountDetail to the entire IncomeStatementSummary cube view, select IncomeStatementSummary and click **General** > **Navigation Links** . Enter the names of the linked cube view in **Linked Cube Views** or click the ellipsis to find the cube view using Object Lookup.

7. Enter the Bound parameters which will pass from one cube view to the next. In this example, a Bound parameter was entered for Account and Time so both members are based on the selected IncomeStatementSummary data cell.



Run IncomeStatementSummary, noting that you can right-click cells to access the AccountDetail cube view.



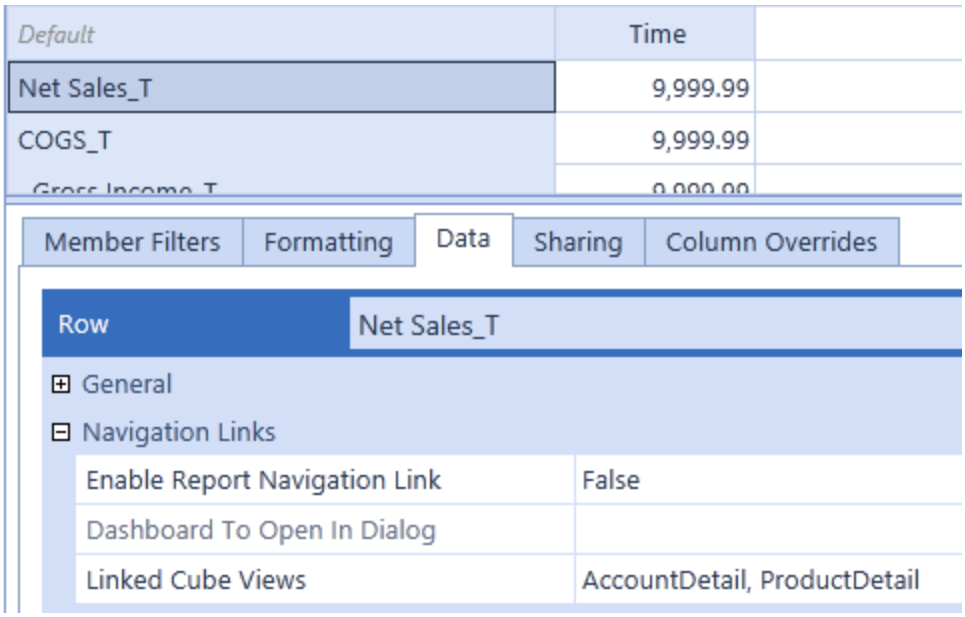


Link a Cube View to Specific Rows or Columns

This topic describes how to assign a linked cube view to specific rows. The ProductDetail cube view will display product details based on the IncomeStatmentSummary's selected data cell. The IncomeStatementSummary's data displays parent accounts for a specific time, so the ProductDetail's data displays the selected account's product sales.

In the ProductDetail cube, configure the rows/columns, POV, and formatting to ensure that the user sees the correct data. See the AccountDetail instructions above for details.

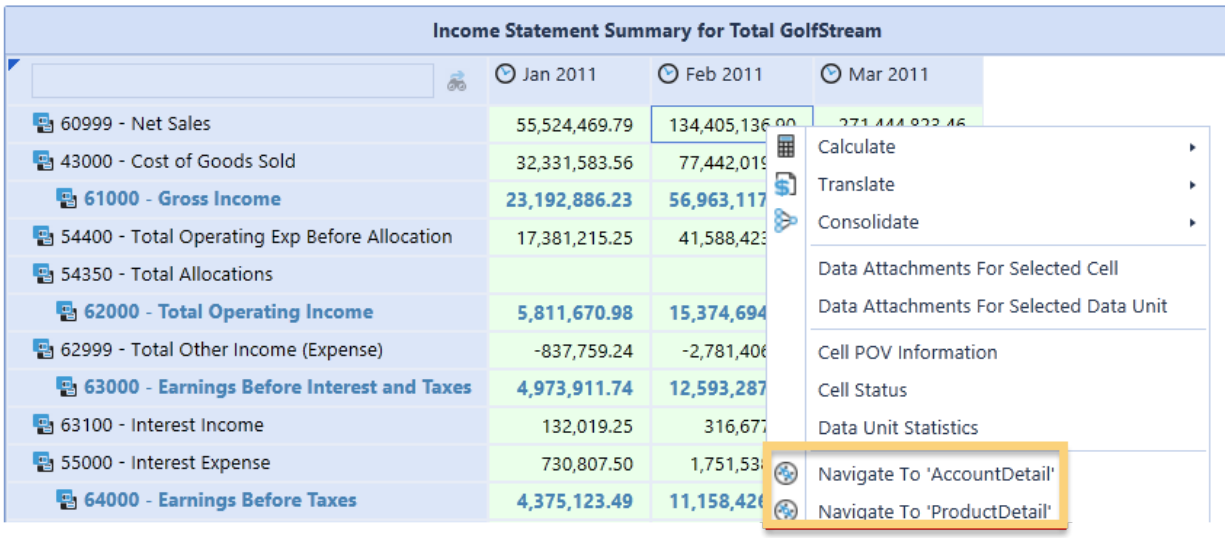
To assign the ProductDetail cube view to IncomeStatementSummary rows, select the IncomeStatementSummary cube view, then go to the Linked Cube Views property on the Data tab in the Rows/Columns. Select the row or column and enter the name of the cube view.



> **NOTE:** Due to overrides, assign the AccountDetail cube view to this property too. Previously, AccountDetail was assigned to the entire cube view. However, because a cube view is being assigned to a row, the cube view settings are overridden and only display the cube view specified for this property. For both cube views to be available on the row, specify them both.

If any Bound parameters need to be entered (in this example, the same Bound parameters are used for both cube views), enter them under Navigation Links in General Settings.

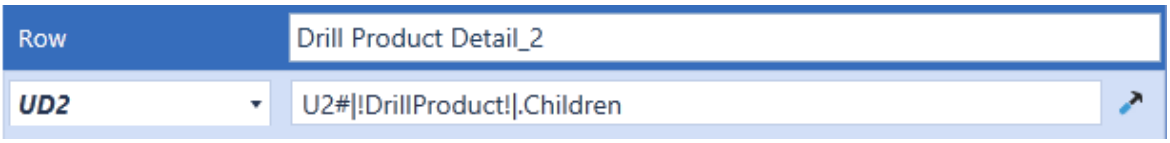
In the example above, the ProductDetail Cube View was assigned to the Net Sales row. When a user right-clicks on a Net Sales cell in the Data Explorer grid, Spreadsheet tool, or Excel Add-In, this cube view is available.



Nest Linked Cube Views

You can link a cube views to cube views that are already linked. In this example, the ProductDetail_2 cube view is linked to ProductDetail and displays specific product data tied to a selected ProductDetail cell.

1. In ProductDetail_2, configure the rows and columns as needed.



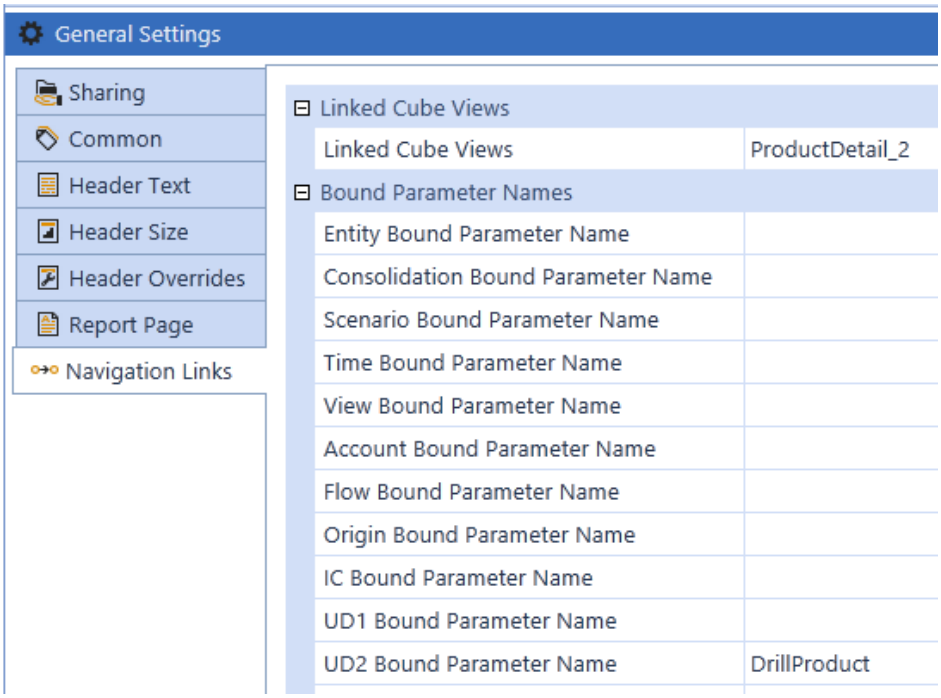
2. Enter a |!DrillProduct!| Bound parameter to display specific product details. For example, if you right-click a Clubs cell in the ProductDetail cube view and select the ProductDetail_2, sales details for Clubs displays.

3. Ensure a |!DrillTime!| parameter is also used.

4. Set the POV for ProductDetail_2 so it matches the ProductDetail POV, except for the User Defined 2 member. In this example, products are viewed using the UD2 member. Enter |!DrillProduct!| in User Defined 2.
5. Add any required formatting to the ProductDetail_2, such as page captioning:

Page Caption	Product Detail for |CVUD2|

6. Open ProductDetail and navigate to Linked Cube View under General Settings > Navigation Links.
7. Click the ellipsis and select the ProductDetail_2 cube view. This link the cube view to the entire ProductDetail cube view.
8. Enter the Bound parameter from ProductDetail_2.



9. Run ProductDetail from IncomeStatementSummary, right-click cells and select **Navigate To ProductDetail_2**.

Analyze Data in Linked Cube Views With Passed Point of View Selections

Use the Include Default NavLink Parameters property to enhance cube view design and navigation for optimal data analysis. Set the property to True to use the following standard syntax to more easily look up parameters and ensure parameter names match across linked cube views and other objects.

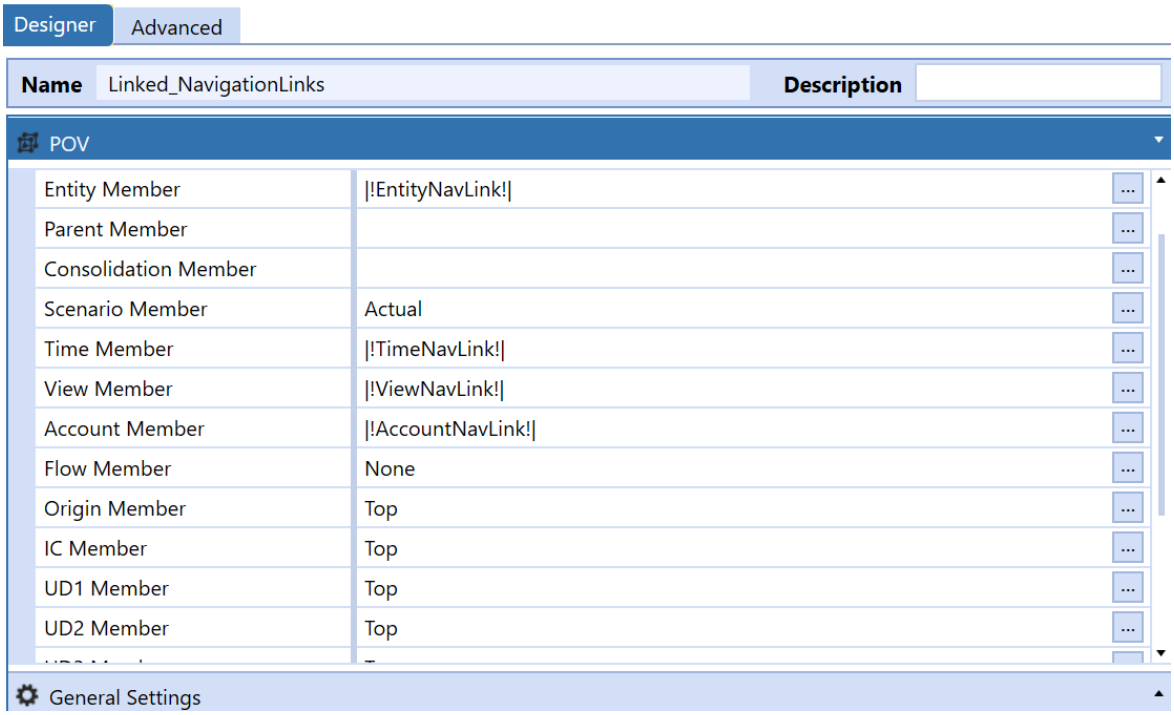
`|!dimensionNavLink!|`

Point of view selections you specify in a source cube view pass to the linked cube view, for a refined, focused analysis. For example, to pass the Time dimension from a source cube view to a linked cube view, specify `|!TimeNavLink!|` in the linked cube view's POV. The data in the cell point of view in the source cube view (2018Q1 as shown below) passes to the linked cube view's point of view at runtime.

To use passed point of view selections in a linked cube view:

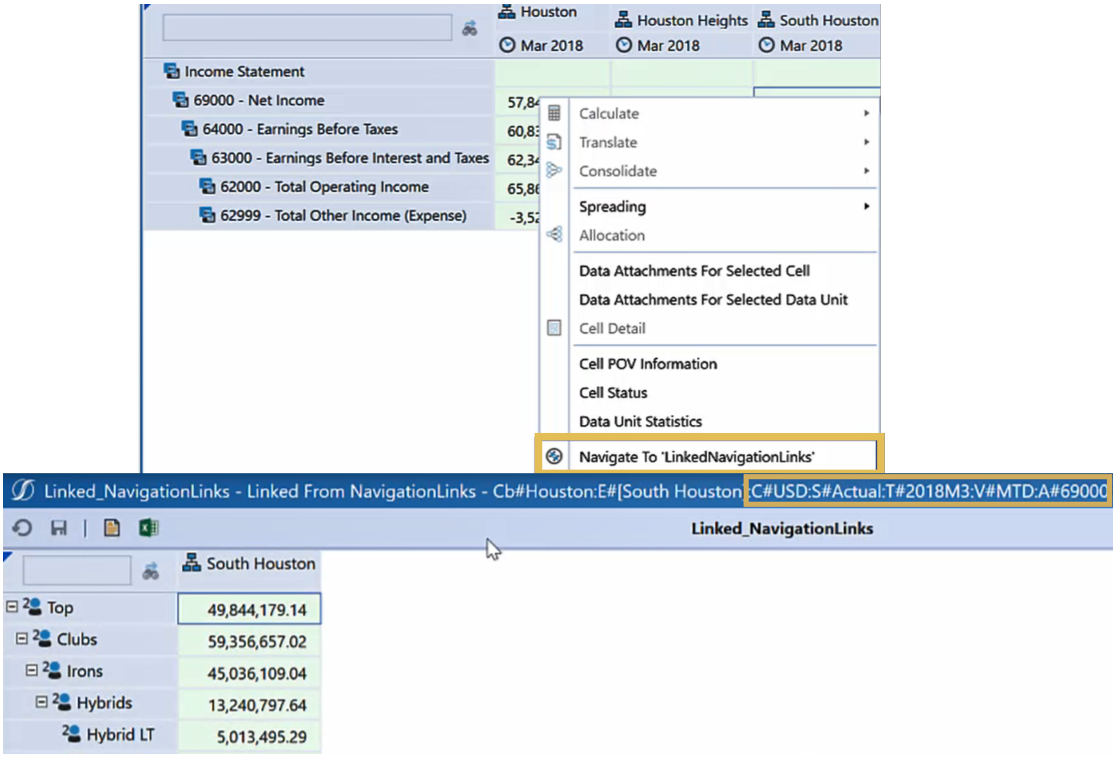
1. Click **Application** > **Cube Views**, and then the cube view containing the source data to access and analyze further in a linked cube view.
2. Click **General Settings** > **Navigation Links**.
3. Set **Include Default NavLink Parameters** to **True** and click **Save**.
4. Open the linked cube view in which to retrieve and evaluate the source data based on point of view selections. Set **Include Default NavLink Parameters** to **False**.

5. Click **POV** , specify |!dimensionNavLink!| for each dimension to analyze and then click **Save**.



6. Return to the source cube view and click **Open Data Explorer**.

7. Right-click cells to navigate to the linked view, retrieving data for the specified point of view parameters that are also identified in the header.



Format a Report Page

The following settings in the cube view address page formatting:

- Header (text, size, overrides)
- Report
- Headers and footers

Ensure the general settings are correct before formatting your page. See General Settings.

To edit the header and report settings, on the **Cube Views** page, under **Cube View Groups**, select a cube view and then **General Settings** to expand the slider.

Header Settings

Row and column headers can be adjusted to refine your cube view.

- **Header Text**: Decide if you want to see the name, description, or both for each of the dimension types. Also, set the Culture property of a cube view, which determines the language for the report. Ensure the Member Descriptions Culture properties are set up correctly before setting the Culture property.
- **Header Size**: Provide a height for columns and width for rows. The rows can be edited separately by Data Grid (Row Header), Excel (Excel Row Header), and PDF (Report Row Header). Each of these items defaults to -1, but you can change them by typing a pixel number.
- **Header Overrides**: Change which dimensions display in the rows and columns. The default for a cube view is to display what is available in the rows and columns. However, you can apply Header Overrides to change the order or add dimensions.

  The Report Column Index for Row Headers can also be set here. This changes the location of the row headers from their usual place (the left). Type the number of columns that should display before the row headers are shown. This only applies to the PDF version of the report.

Report Settings

These properties apply advanced formatting to cube views.

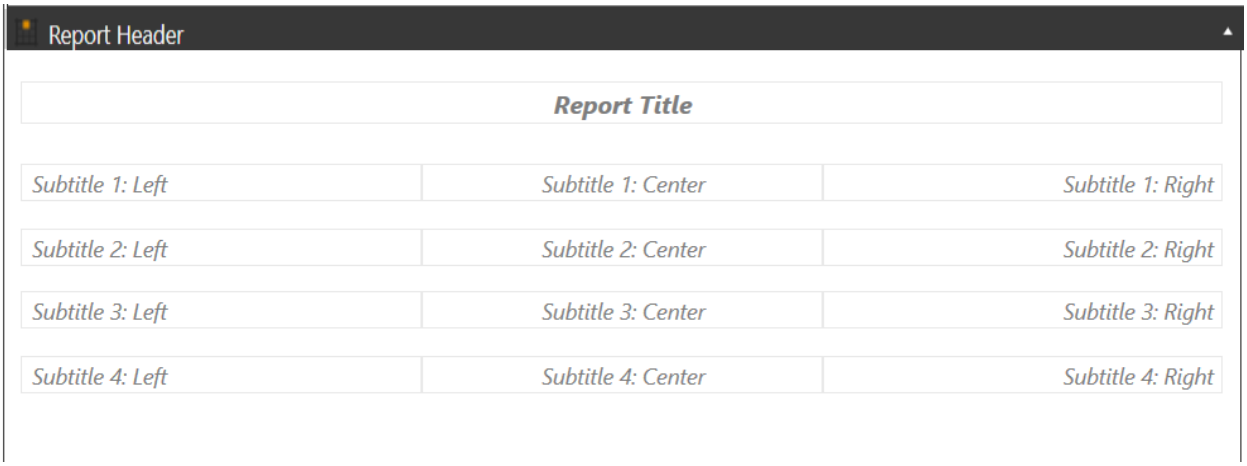
- **Custom Report Formatting**: Write or assign cube view extender rules to provide intricate cube view formatting. This only applies to the PDF version.

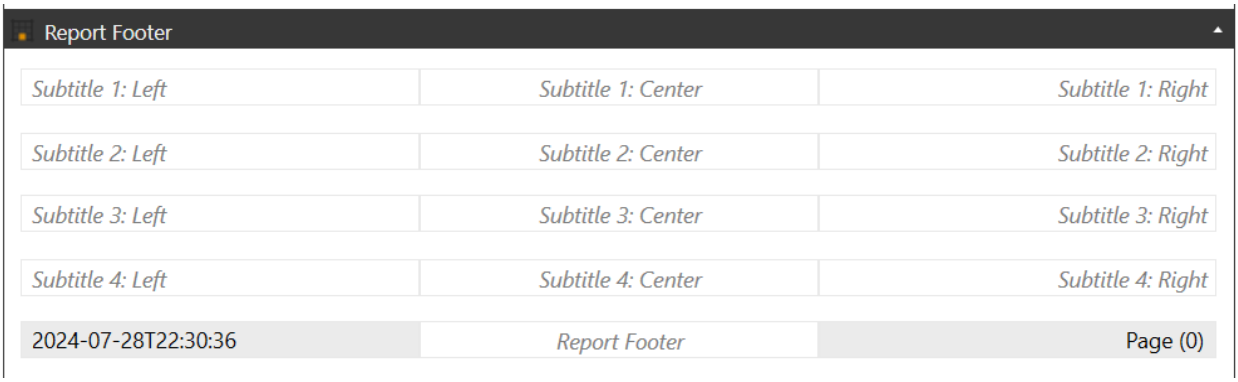
- **Paper**: Select the paper size.
- **Landscape**: Select the page orientation.
- **Margin**: Set margins on individual reports. Type a pixel number to override the default settings of the page in the general settings.
- **Auto Fit Settings**: Set **Auto Fit to Page Width** to **True** if the report should re-size to fit all columns on one page. Use the **Auto Fit Number of Pages Wide** property to type the number of pages to display the columns.
- **Scale Factor**: Sets and adjusts the scale of the report by this value and automatically centers report content horizontally. The default value is 1, and the maximum value is 2. If you enter a value greater than 2, it will default to a scale factor of 2.

> **NOTE:** Scale Factor is not compatible with Auto Fit to Page Width.

Header and Footer Settings

Report headers and footers control what displays at the top and bottom of the page when a cube view is run as a data explorer report. Standardize headers and footers across reports and make as dynamic as possible to reduce reporting maintenance. On the Designer tab of the Cube View, Report Header and Report Footer are separate sliders. Enter text in the fields or use substitution variables.





Use substitution variables to reduce the maintenance of headers and footers. Substitution variables are short scripts that use pipe characters to include a predefined substitution variable. They come with every installation of OneStream and cannot be edited, so you do not need to create or maintain them. Substitution variables can be used throughout the application. For cube views, you can use them for:

- Headers and footers
- Rows and columns
- Cube view page captions

To copy and paste a substitution variable into your application:

1. On the **Cube Views** page, click **Object Lookup** to open the dialog box.
2. Under **Object Type**, select **Substitution Variables** to see the list of options.
3. Select the name of the substitution variable.
4. Click **Copy to Clipboard**.
5. Paste the substitution variable where needed.

Calculations

When you create a calculation in a cube view, the data is generated on-the-fly and not stored in the database. When you run the cube view, the calculation returns a value.

You can use calculations in a cube view to calculate:

- Variances
- Sums
- Differences
- Ratios

Calculations are useful for variance reporting, What If scenarios in cube views, and KPI calculations.

Types of Cube View Calculations

There are three ways to create calculations in cube views. Each option uses the GetDataCell function, which retrieves specific cells and performs math or business rule operations.

The most common ways to create a calculation in a cube view use:

- GetDataCell Expressions
- Column and Row Expressions
- Dynamic Member Formulas and Cube Views

Choose the option that has the least maintenance and highest usability.

> **NOTE:** You can also create more complex calculations with business rules.

GetDataCell Expressions

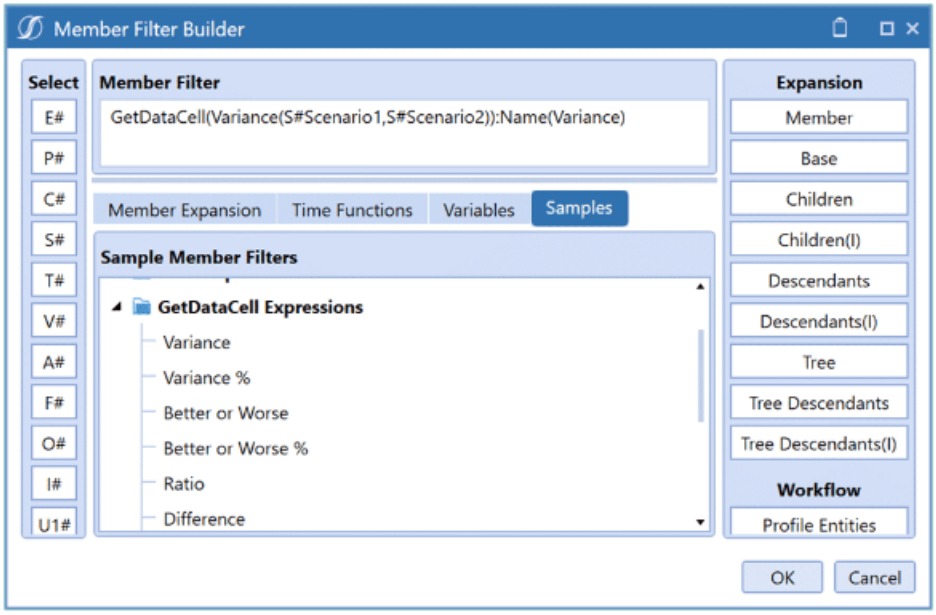
Write GetDataCell expressions in the cube view row or column. It will calculate at runtime. The expressions require a reference to specific dimension members to return a data point. Use GetDataCell expressions if you are running calculations on members that are not currently in the cube view or if the expression is easy to create and maintain. GetDataCell expressions are simpler versions of dynamic calculations.

> **NOTE:** Use one of the other calculation types if you are querying multiple member combinations.

Samples Tab

In the Member Filter Builder, use the Samples tab to create simple GetDataCell expressions in your cube view.

In Member Filter Builder, click **Samples**.



There is a list of preexisting GetDataCell expressions you can use. For detailed information, see the following table.

GetDataCell Expressions

Function Name	Description	Example
Variance	Calculates the difference as a ratio between two scenarios using the Variance function.	GetDataCell(Variance(S#Scenario1, S#Scenario2)):Name(Variance)

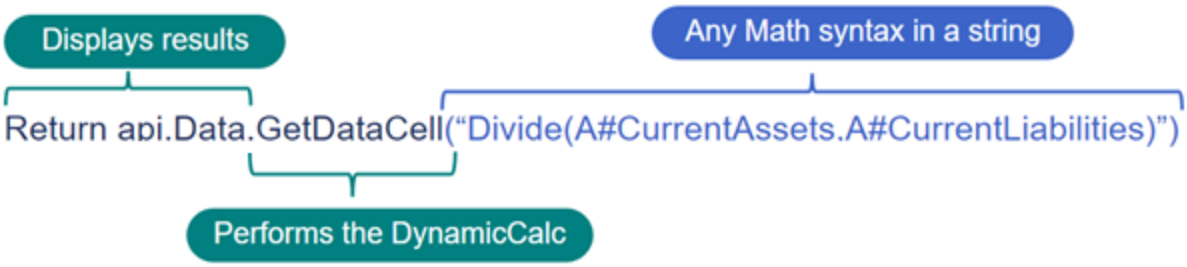
Function Name	Description	Example
Variance%	Calculates the difference as a percentage between two scenarios using the VariancePercent function.	GetDataCell(VariancePercent (S#Scenario1, S#Scenario2)):Name(Var%)
Better or Worse	Calculates the difference between two scenarios while considering account types using the BWDiff function.	GetDataCell(BWDiff(S#Scenario1, S#Scenario2)):Name(BW Diff)
Better or Worse%	Calculates the difference as a percentage between two scenarios while considering account types using the BWPercent function.	GetDataCell(BWPercent(S#Scenario1, S#Scenario2)):Name(BW%)

Function Name	Description	Example
Ratio	Calculates the ratio between two scenarios using the Divide function.	GetDataCell(Divide(S#Scenario1, S#Scenario2)):Name(Ratio)
Difference	Calculates the difference between scenarios using the Subtraction function.	GetDataCell(S#Scenario1-S#Scenario2): Name(Difference)
Sum	Calculates the sum of two scenarios using the Addition function.	GetDataCell(S#Scenario1+ S#Scenario2: Name(Total)
Custom Function (condensed syntax)	Calls a Finance business rule using the BR# function.	GetDataCell(BR#[MyBusinessRuleName, MyFunctionName]): Name(Custom Function)
Custom Function	Calls a Finance business rule using the BR# function.	GetDataCell(BR# [BRName=MyBusinessRuleName, FunctionName=MyFunctionName]):Name (Custom Function)

Function Name	Description	Example
Custom Function with Parameters	Calls a Finance business rule using the BR# function.	GetDataCell(BR#[BRName=MyBusinessRuleName, FunctionName=MyFunctionName, Name1=Value1, AnotherName=[Another Value]]):Name(Custom Function)

Components of GetDataCell

A GetDataCell expression is the performance section of a dynamic calculation. It tells the dynamic calculation what it will be doing. The following example shows a common GetDataCell expression. GetDataCell expressions are simpler versions of dynamic calculations.



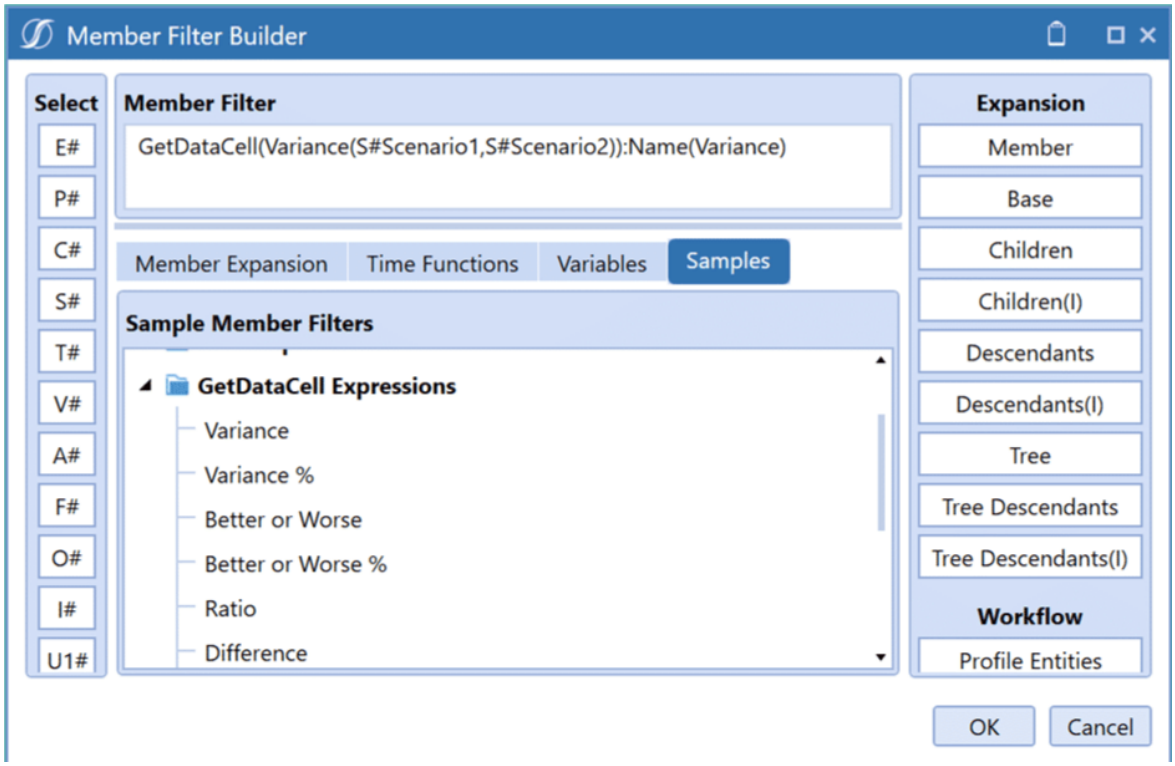
Return api.Data.GetDataCell("Divide(A#CurrentAssets,A#CurrentLiabilities)")

- **Return api.Data**: Tells OneStream which results to display.

> **NOTE:** This is not required for GetDataCell expressions, but it is required for dynamic calculations.

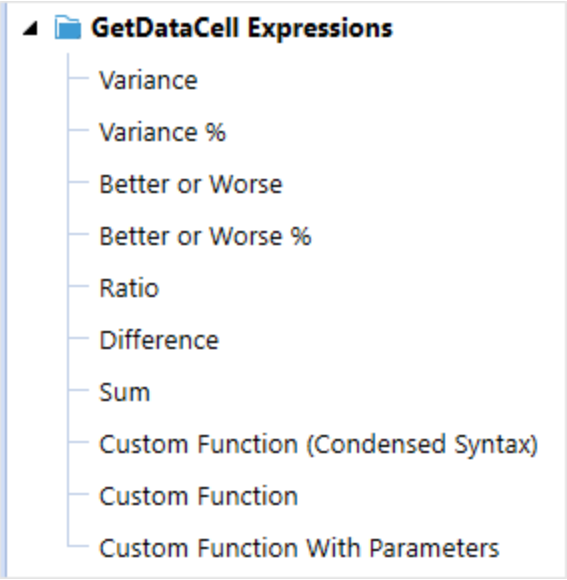
- **GetDataCell**: Performs the calculation.
- **("Divide(A#CurrentAssets,A#CurrentLiabilities)**: Math syntax in a string format to calculate.

You can use the Samples tab to create simple GetDataCell expressions in your cube view.



GetDataCell Types

There are several GetDataCell expressions. You can find a complete list under **OnePlace** > **Dashboards** > **Application Reports** > **Application Reports Dashboards** > **Report Groups** > **Application Analysis**.



- **Variance**: Calculate the difference as a ratio between two scenarios using the [Variance] function in a GetDataCell expression.

  For example: GetDataCell(Variance(S#Scenario1,S#Scenario2)):Name(Variance)

- **Variance %**: Calculate the difference as a percentage between two scenarios using the [VariancePercent] function in a GetDataCell expression.

  For example: GetDataCell(VariancePercent(S#Scenario1,S#Scenario2)):Name(Var %)

- **Better or Worse**: Calculate the difference between two scenarios while considering account types using the [BWDiff] function in a GetDataCell expression.

  For example: GetDataCell(BWDiff(S#Scenario1,S#Scenario2)):Name(BW Diff)

- **Better or Worse %**: Calculate the difference as a percentage between two scenarios while considering account types using the [BWPercent] function in a GetDataCell expression.

  For example: GetDataCell(BWPercent(S#Scenario1,S#Scenario2)):Name(BW %)

- **Ratio**: Calculate the ratio between two scenarios using the [Divide] function in a GetDataCell expression.

  For example: GetDataCell(Divide(S#Scenario1,S#Scenario2)):Name(Ratio)

- **Difference**: Calculate the difference between two scenarios using the [Subtraction] function in a GetDataCell expression.

  The example that appears when you double click this expression in Member Filter Builder: GetDataCell(S#Scenario1-S#Scenario2):Name(Difference)

- **Sum**: Calculate the sum of two scenarios using the [Addition] function in a GetDataCell expression.

  For example: GetDataCell(S#Scenario1+S#Scenario2):Name(Total)

- **Custom Function (Condensed Syntax)**: Call a Finance business rule using the [BR#] function in a GetDataCell expression.

  For example: GetDataCell(BR#[MyBusinessRuleName, MyFunctionName]):Name(Custom Function)

- **Custom Function**: Call a Finance business rule using the [BR#] function in a GetDataCell expression. For example:

  GetDataCell(BR#[BRName=MyBusinessRuleName,FunctionName= MyFunctionName]):Name(Custom Function)

- **Custom Function with Parameters**: Call a Finance business rule using the [BR#] function in a GetDataCell expression. For example:

  GetDataCell(BR#[BRName=MyBusinessRuleName, FunctionName=MyFunctionName, Name1=Value1, AnotherName=[Another Value]]):Name(Custom Function)

Create a GetDataCell Calculation

GetDataCell expressions run a calculation directly in a cube view by referencing specific dimension members.

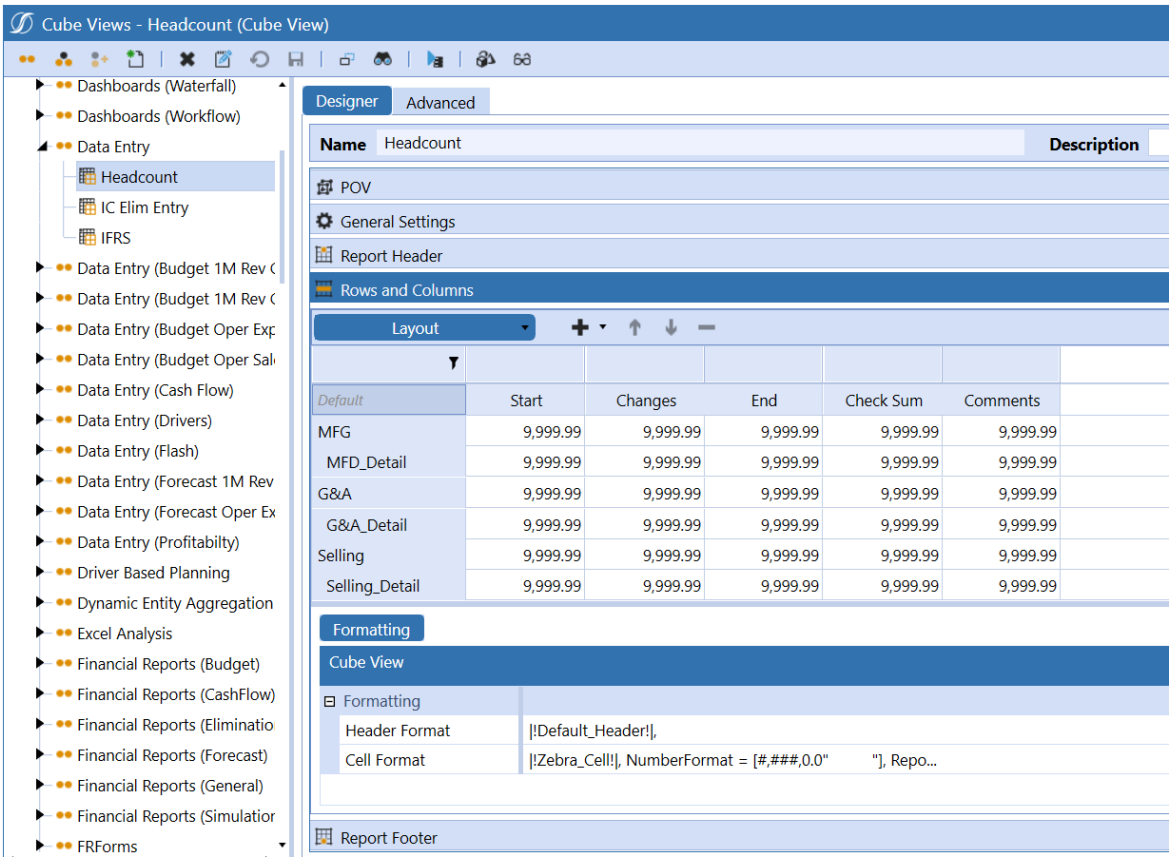
In the Member Filter Builder, use the Samples tab to create simple GetDataCell expressions.

Some common errors when developing GetDataCell expressions are:

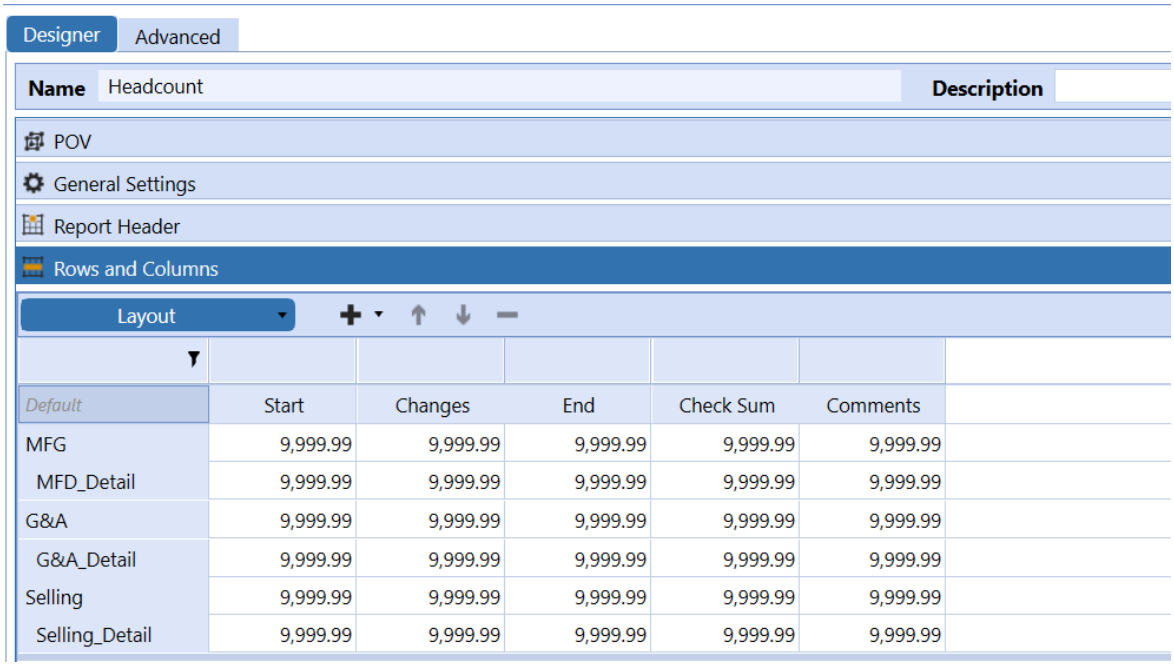
- Not having the correct dimension listed. Make sure to use the Member Filter Builder if you are not sure what to use.
- Not having the correct number of parentheses. Make sure to use the correct number of opening and closing parentheses.

To add a row expression to a column in a cube view:

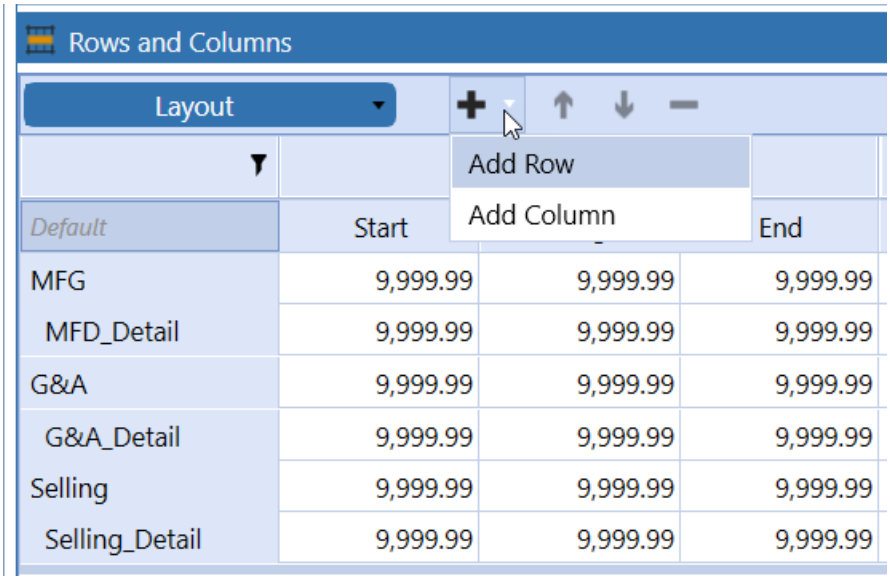
1. In OneStream, go to **Application** > **Presentation** > **Cube Views**.
2. Select a cube view group and then a cube view.



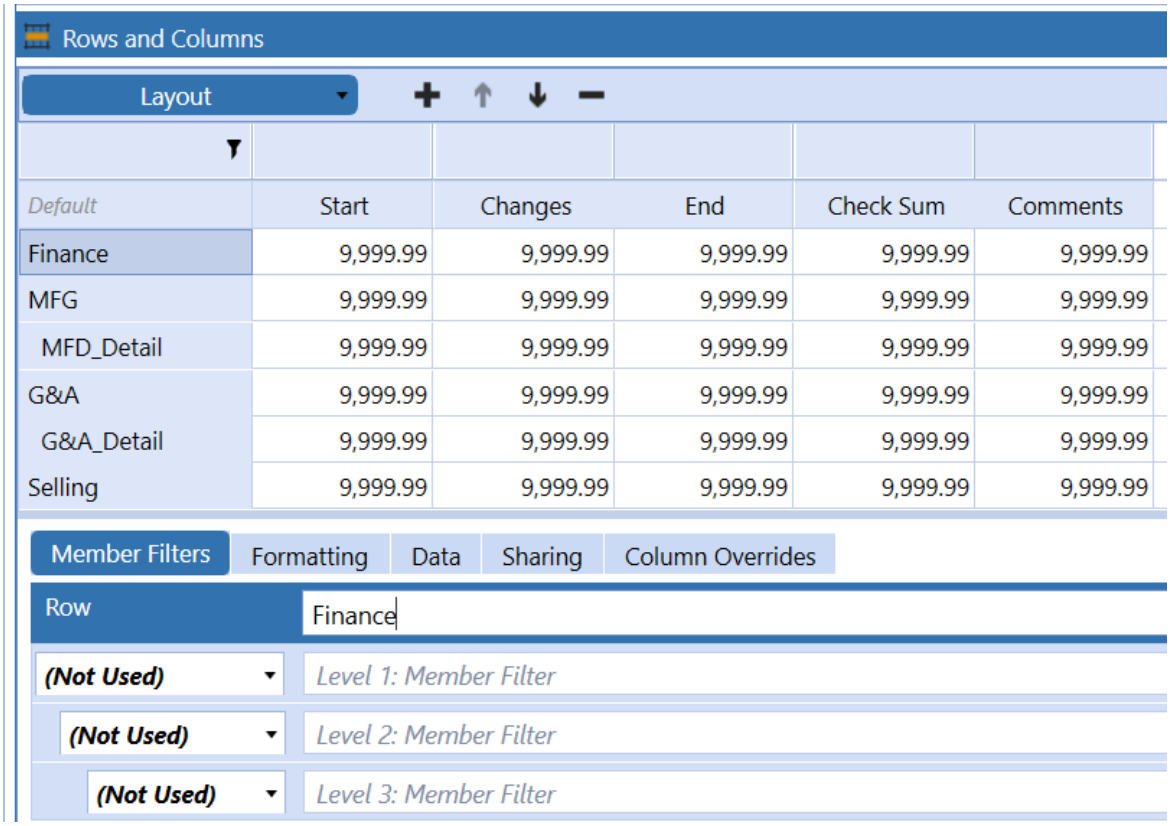
3. Expand the Rows and Columns slider.



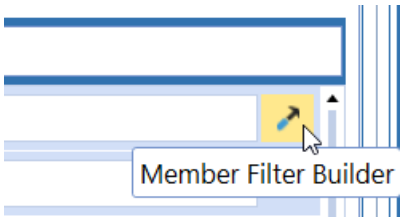
4. To add a row to insert the CVC/CVR calculation, in Rows and Columns, click the plus sign.



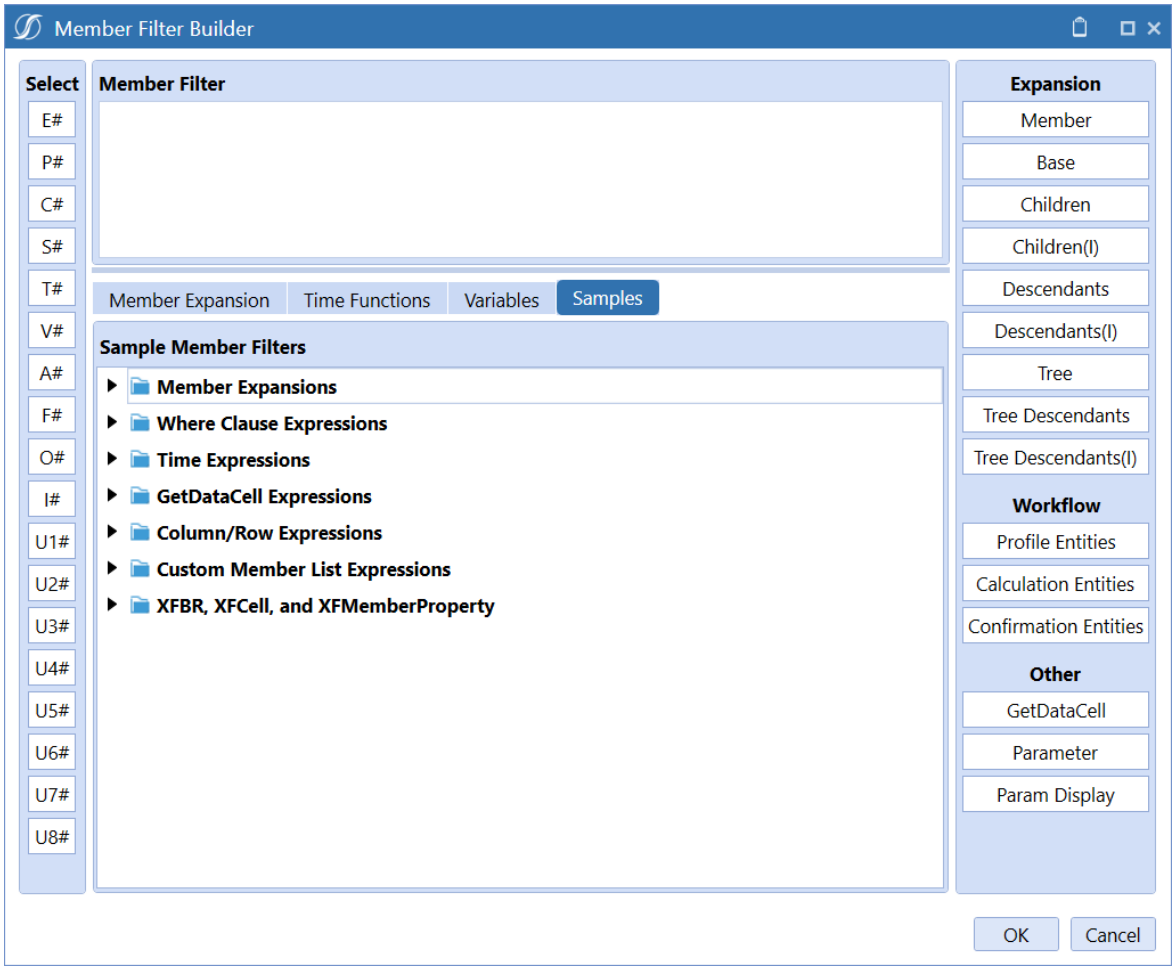
5. In Member Filters, rename the row.



6. Click **Member Filter Builder** for the row to insert a calculation.



7. Delete any existing content in the Member Filter field.
8. Click **Samples**.



9. Expand **Column/Row Expressions**.
10. Click **Sum of Rows (CVR)**. The GetDataCell syntax is generated and displays in Member Filter.
11. Update the row names to add the rows together. Rename Row1 to Cash and Row2 to AR.
12. Replace the name at the end of the expression to reflect the row name. The expression is complete.
13. Click **OK** and then click **Save**.
14. Click **Open Data Explorer** to run the cube view and verify the result.

Troubleshoot GetDataCell Errors

These are some common errors when using GetDataCell expressions:

- Incorrect dimension: Make sure you have the correct dimension listed. If you are unsure, use the Member Filter Builder.
- Incorrect use of parentheses: Verify that you have the correct number of parentheses and that they are positioned correctly to avoid a script error.

Column and Row Expressions

Cube views accept calculated columns and rows using the GetDataCell function and references to cube view column and row names. This is known as Column/Row math.

Write column and row expressions in the cube view row or column itself. They calculate at runtime.

These expressions require a reference to the specific cube view column or row name to return a data point.

Use column and row expressions if you need to perform math across columns and rows. A common use case is with variance columns. Column and row expressions are useful if the query is more complex or if the expression is easy to understand and maintain.

> **NOTE:** Use one of the other expression types if you are using multiple member expansions, especially when performing math on rows.

Create Cube View Column and Row Calculations

Cube views accept calculated columns or rows using the GetDataCell() function and references to cube view column or row names. This is also known as cube view column/row math.

These are based on the names given to a cube view column, for example "Col1" or "TimePeriods" or cube view row, for example “Row2” or “Accounts” by the creator of the cube view.

> **NOTE:** Using the column or row names may be more useful than using member filters.

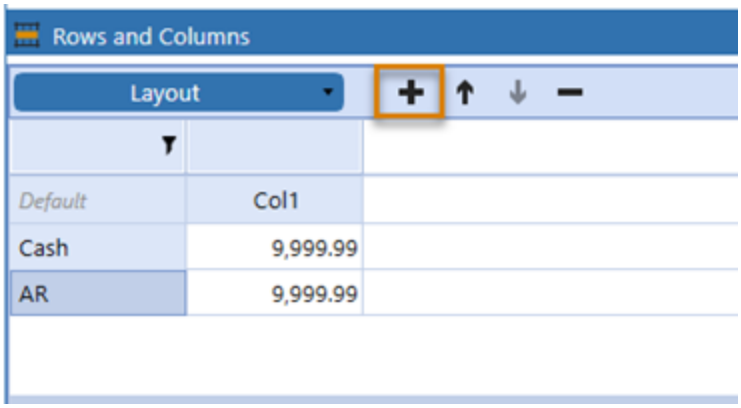
- Cube view columns: Uses the GetDataCell function to perform math on columns.
- Cube view rows: Uses the GetDataCell function to perform math on rows.

There are several variations on this method, depending on whether the expression refers to columns, rows, or a combination.

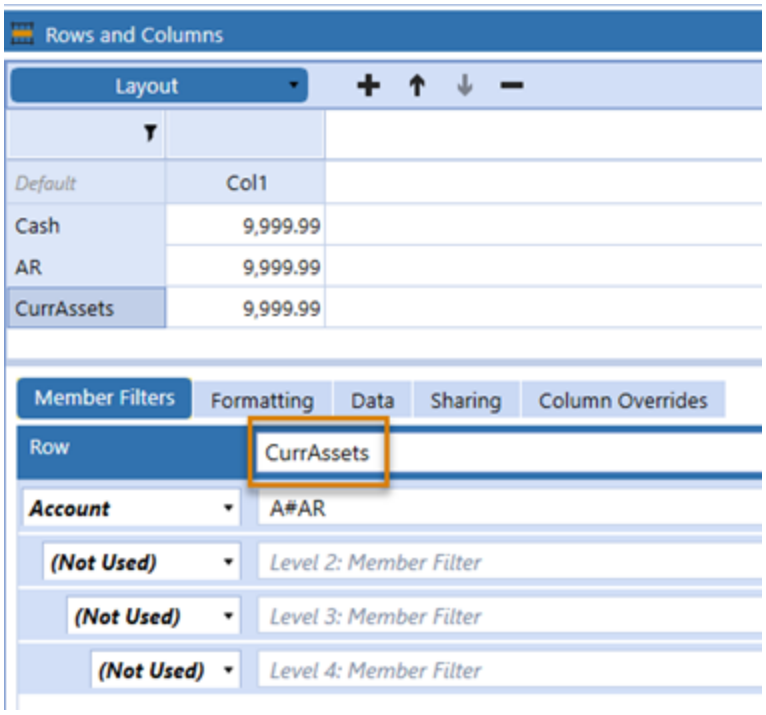
When naming a new calculation, do not include spaces or special characters that could make the column/row calculations read incorrectly.

Create a CVC/CVR Calculation

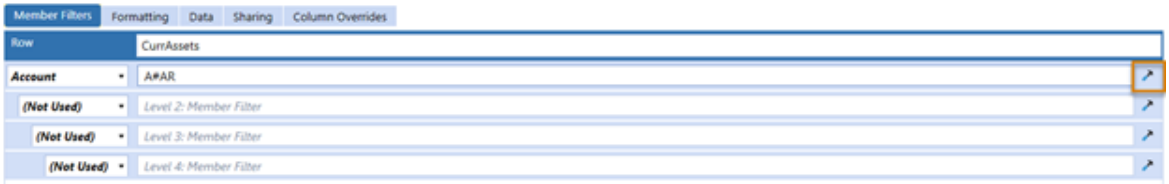
1. Add a row or column to insert the calculation.



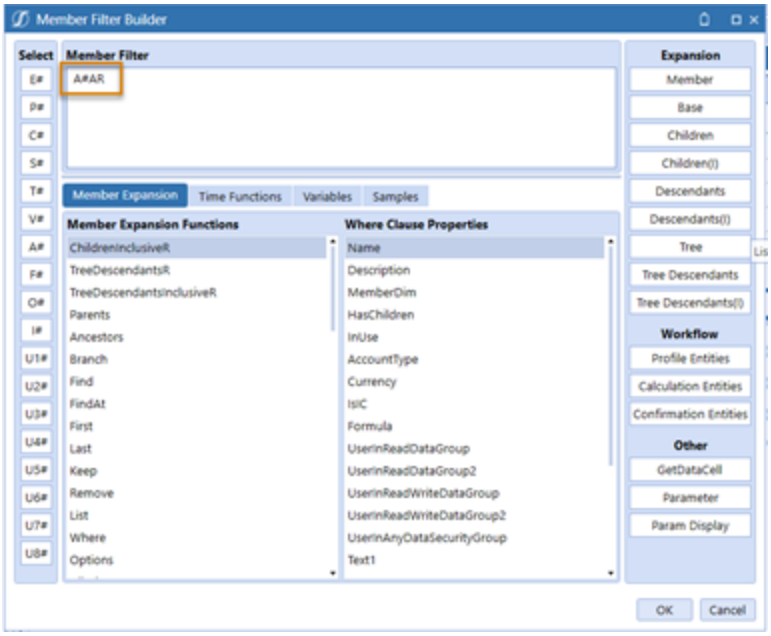
2. Rename the row relevant to your calculation. In this case, the is CurrAssets.



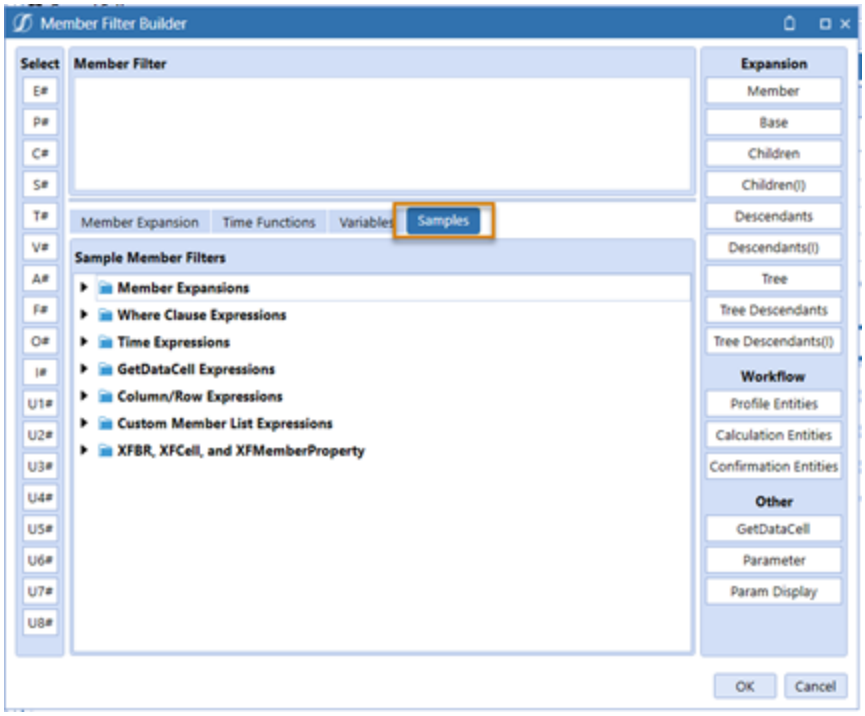
3. Open the Member Filter Builder for the row or column where you want to insert a CVC/CVR calculation.



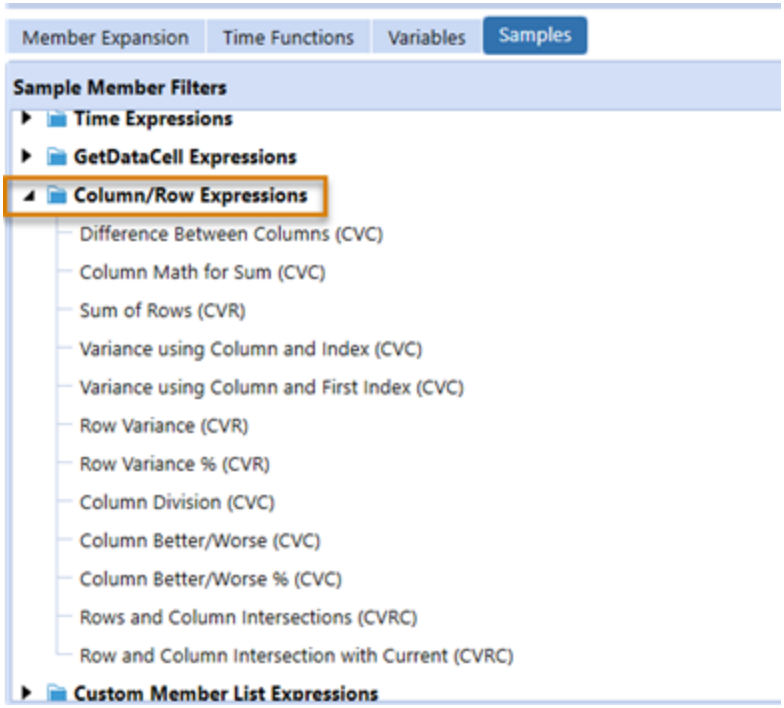
4. Delete what is in the Member Filter pane.



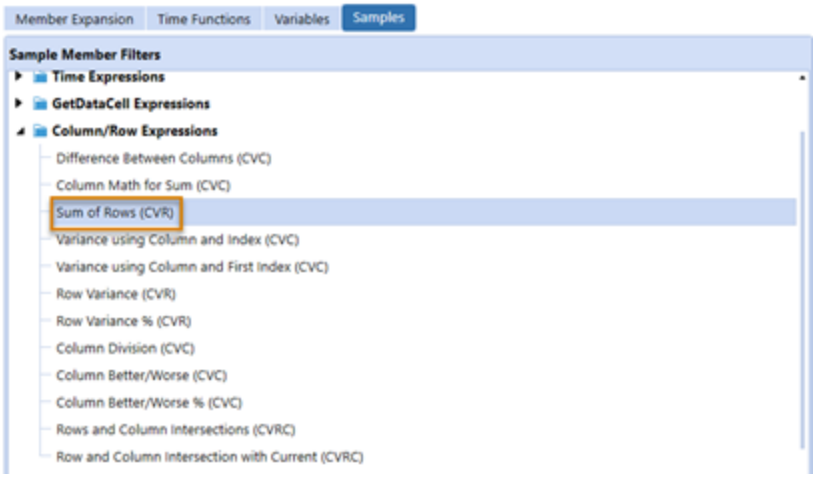
5. Navigate to the Samples tab.



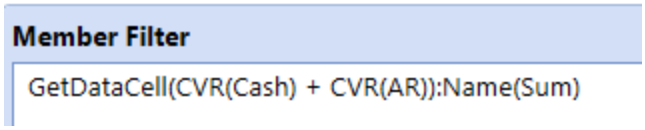
6. Expand Column/Row Expressions.



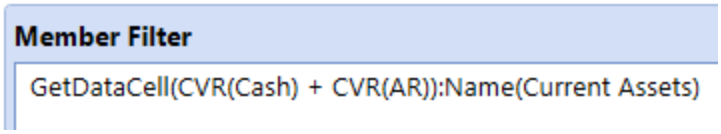
7. Double-click Sum of Rows (CVR). This will populate the GetDataCell expression that you need.



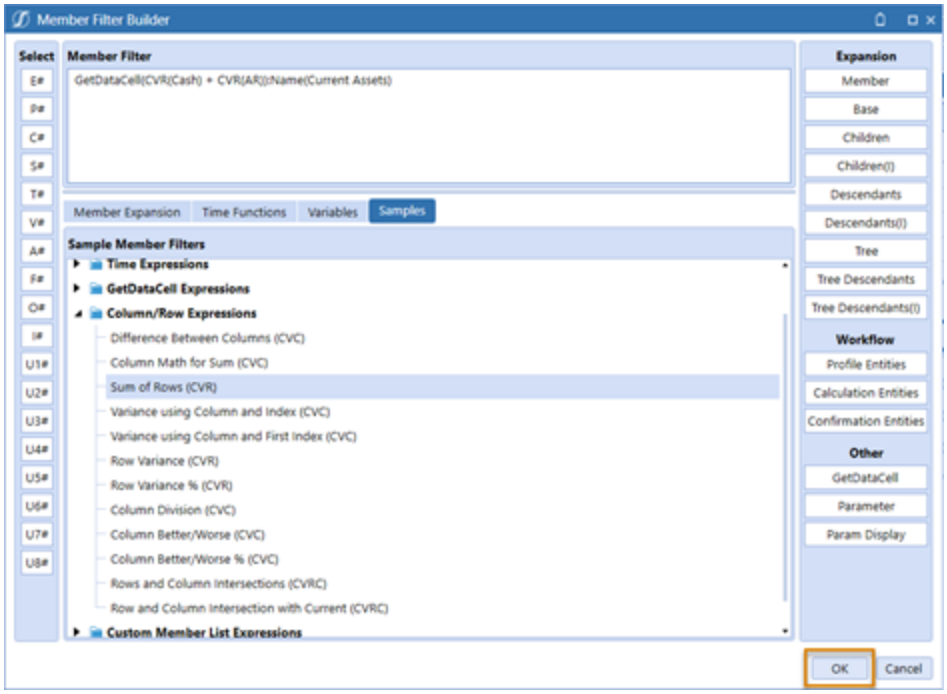
8. Update the row names to add the rows together. Rename Row1 to Cash and Row2 to AR because Cash and AR are row header names used in the cube view.



9. Replace the name at the end of the expression to display Current Assets.



10. Click **OK**.



11. Click **Save**.

12. Run the cube view.

CVC: Cube View Column Calculations

The example below shows a Column Math example of the difference between columns (CVC).

The syntax is GetDataCell(CVC(SomeColumnName) - CVC(SomeOtherColumnName)):Name (Header Name)

The example of simple member math includes the :Name() function typically applied to a column in a cube view:

```
GetDataCell(CVC(Col1) - CVC(Col2)):Name(Variance)


GetDataCell(CVC(Col1) + 1):Name(Column Plus One)

GetDataCell(CVC(Col1) * (-1)):Name(Column with Sign Flipped)
```

CVR: Cube View Row Calculations

The example below shows a Row Math example of Sum of Rows (CVR).

The syntax is similar, but instead of CVC, a calculated row uses CVR for retrieving the value of a row in a formula.

An example of the syntax: GetDataCell(CVR(SomeRowName) + CVR (SomeOtherRowName)):Name(HeaderName)

> **NOTE:** If a column name is numeric (for example, 500), single quotes are required when specifying the row name. Square brackets are allowed, but not required. For example: GetDataCell(CVR('123') - CVR(['4,567'])):Name(Difference)

When you need a column and row index: Col1 has a member filter of S#Actual, S#Budget, meaning it will return two columns. In this case, a variance between Actual and Budget scenarios can be shown like this: GetDataCell(CVC(Col1, 1) - CVC(Col1, 2)):Name(Variance).

Alternatives: GetDataCell(CVC(Col1, First) - CVC(Col1, 2)):Name(Variance) GetDataCell(CVC (Col1) - CVC(Col1, 2)):Name(Variance)

Column Math Example with Division and Other Advanced Functions

Member Expansions may require some more intricate formulas. This may require using indexes or a CVRC expression. Use functions like Divide to avoid divide by zero situations.

GetDataCell(Variance(CVR(Col1,2), CVR(Col1,1)):Name(Variance)GetDataCell (VariancePercent(CVR(Col1,2), CVR(Col1,1)):Name(Variance %) GetDataCell(Divide(CVC (Col3), CVC(Col2))):Name(Ratio) GetDataCell(BWDiff(CVC(Col1), CVC(Col2))):Name (BetterWorse Difference)GetDataCell(BWPercent(CVC(Col1), CVC(Col2))):Name(BetterWorse %)

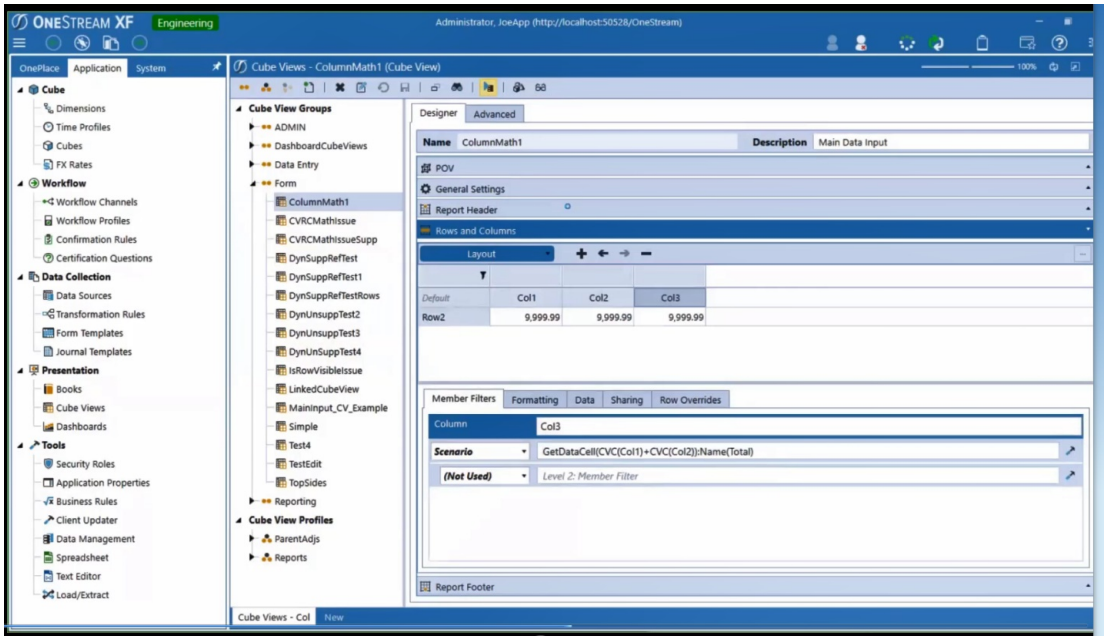
Create Advanced Column Row Expressions

Use an index or a CVRC expression to create more intricate formulas. The Member Filter Builder has complex options in the Samples tab.

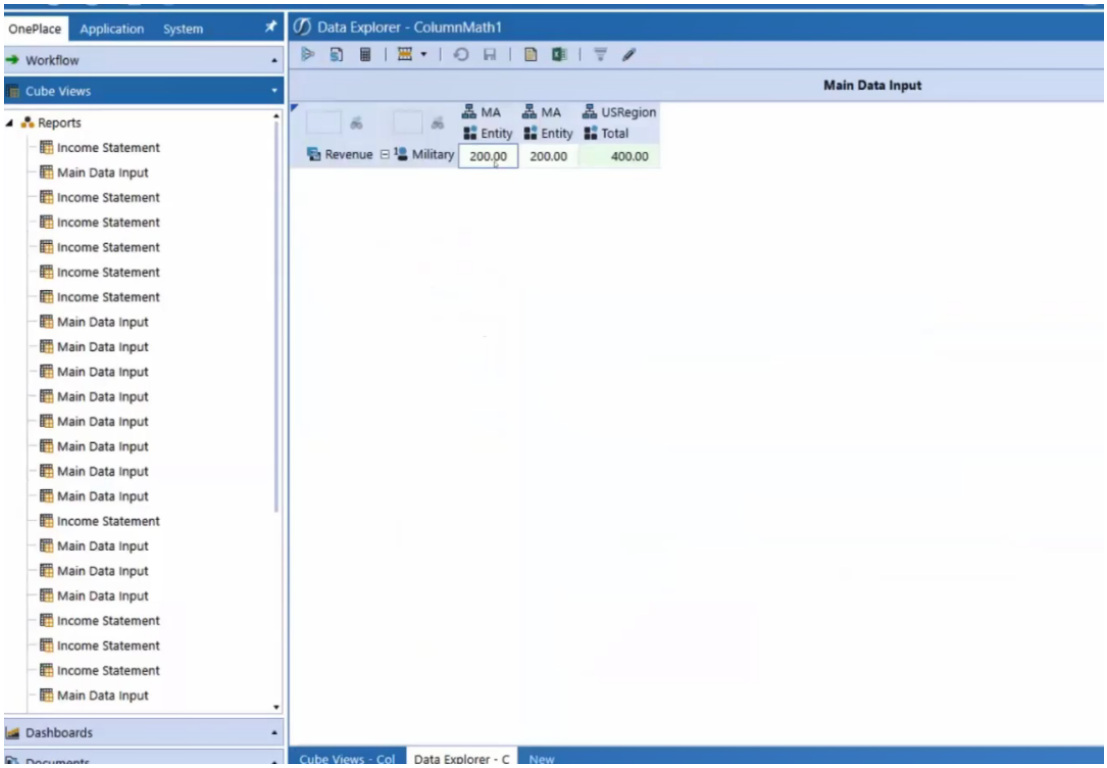
Reference Hidden Columns with Cube View Column Math

Use the **CVMathOnly** option to perform math on columns when the columns are not visible.

1. From the Application tab, under Presentation, click **Cube Views**.
2. Expand Cube View Groups, expand a specific cube view group and select a cube view.
3. Click **Rows and Columns**. In the example below there are three columns, where Column 3 has the GetDataCell() function applied for the total of Column 1 and Column 2.



4. Click **Open Data Explorer** to generate the Cube View and see the data. In the example below you see that it is adding the value 200.00 from Column 1 to the value 200.00 from Column 2 to generate the total of 400.00 in Column 3.



5. Click **Edit**.
6. Select **Column 1** and at the bottom of the window click the Formatting tab.
7. In Header Format, click the ellipsis. The Header Format window opens.
8. Click **Format**.
9. In the General section, find isColumnVisible and select **CVMathOnly**.

Dynamic Member Formulas and Cube Views

Dynamic calculations enhance the consolidation performance because the amount is calculated when requested for display and is not written to the database.

Use dynamic calculations in cube views and business rules.

- There is no performance impact on consolidation times.
- You can make dynamic calculations drillable so users can see the source for the calculated result when drilling down.
- Use these calculations for ratios or variances.

In the dimension library, write dynamic calculations on specific members. They will calculate at runtime.

Dynamic calculations have a syntax similar to GetDataCell expressions because they also require reference to specific dimension members to return a data point.

> **NOTE:** A difference between dynamic calculations and GetDataCell expressions is that dynamic calculations require proper business rule logic.

Use dynamic calculations if you need a repository of commonly used calculations which allows for consistency and reduces maintenance.

Create common calculations in the UD8 dimension. It allows you to have the most flexibility with your rule syntax.

Use another calculation type if the calculation is only used on a specific cube view because you may not want to create a member in the dimension library. Also, consider if you will grant a report builder access to the dimension library to view, create, and edit these calculations.

Create Dynamic Calculations

Dynamic calculations enhance the consolidation performance because the amount is calculated when requested for display and is not written to the database.

Account, Flow, and all UDs can contain dynamically calculated members.

- There is no performance impact on consolidation times.
- You can make dynamic calculations drillable so users can see the formulas running.
- These calculations can be used for ratios or variances.

Two properties must be set to make the calculation dynamic: Account Type and Formula Type. Ensure that the Formula Type is set to DynamicCalc. If using on an Account dimension type, both the Account type and Formula Type need to be set to DynamicCalc.

Account Type	DynamicCalc
Formula Type	DynamicCalc

Here are some common examples of how dynamic calculations are used:

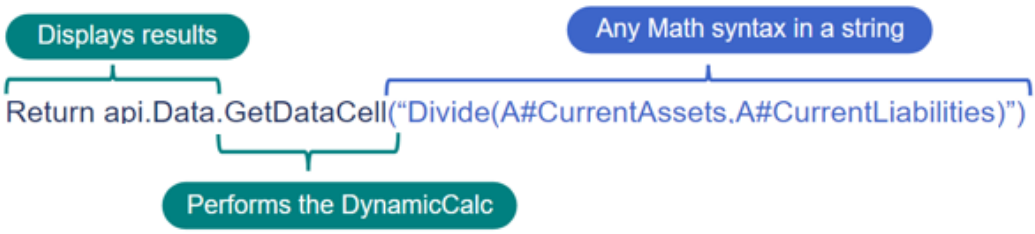
- Api.Data.Calculate("Target = A + B")
- Api.data.calculate("A#InsuranceExp = A#Payroll * A#InsuranceCostPcent")
- Api.Data.GetDataCell("A + B")
- Return Api.data.GetDataCell("A#GrossSales – A#ReturnsAndAllowances")

Calculated Members and Example

In the Dimension Library, there are likely a few members that will be calculated using a member formula. Dynamically calculated members are members that use the GetDataCell function to return values based on math when they are queried.

These members require that a Formula Type of DynamicCalc be defined before writing a member formula.

> **NOTE:** If this property is not defined, the member will return an error and the rule will not compile.



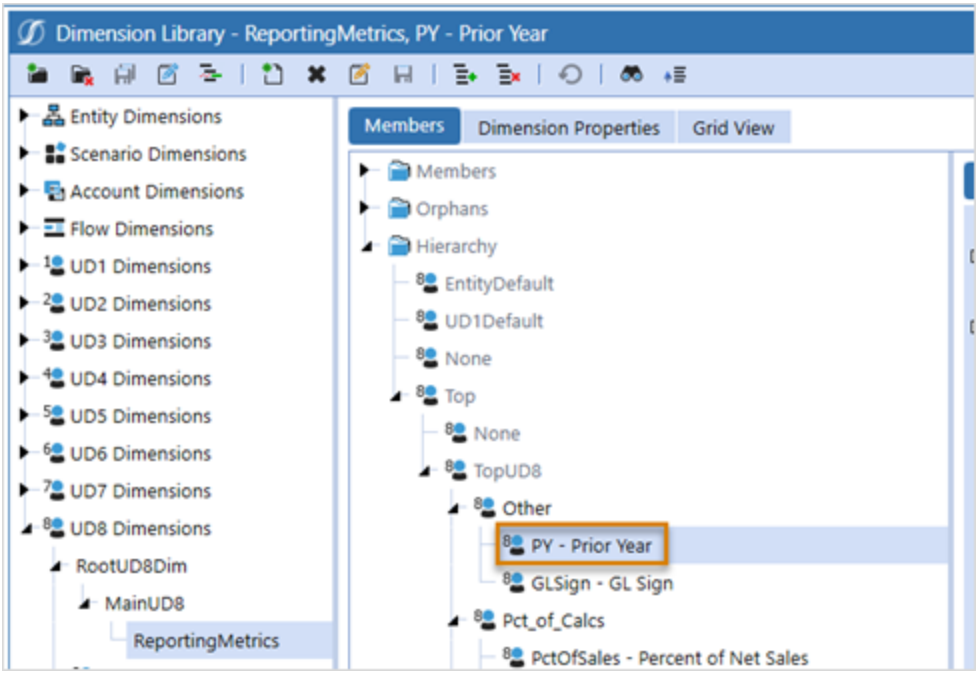
- **Display Results**: This section tells the system how to display the results of the function.
- **Performs the DynamicCalc**: This section tells the system to perform the DynamicCalc using the GetDataCell function.
- **Any Math syntax in a string**: This tells the system what math you want it to perform.

Define a Dynamic Calculation

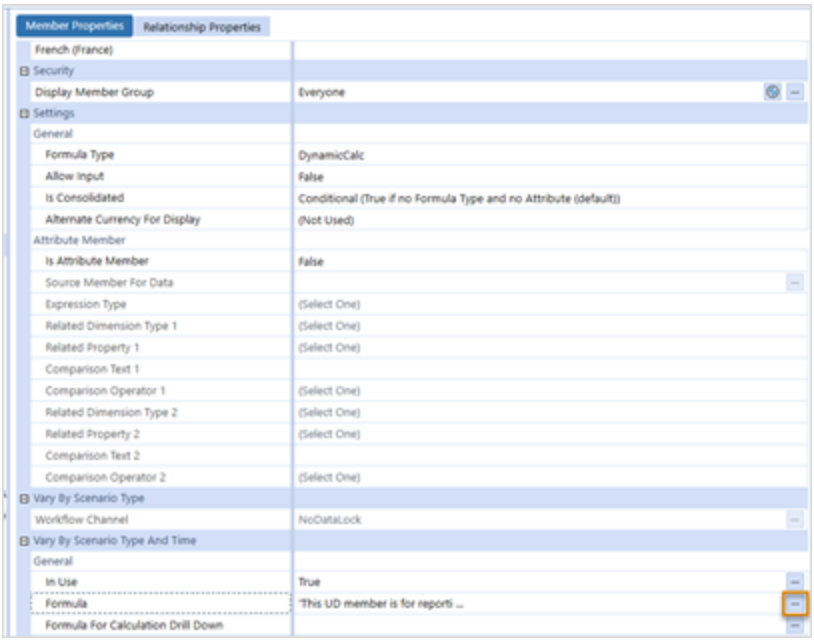
Before you add a dynamic calculation to a cube view, you must define it or write the simple dynamic calculation.

In this example, we are going to navigate to a UD8 member that you want to update to define the simple dynamic calculation.

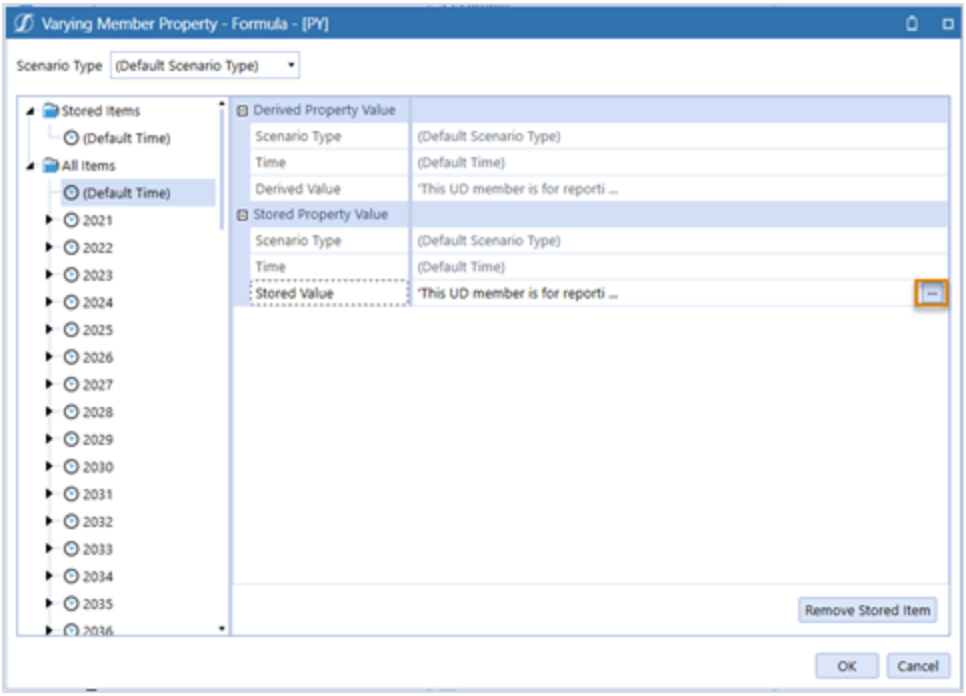
1. Open the UD8 member.



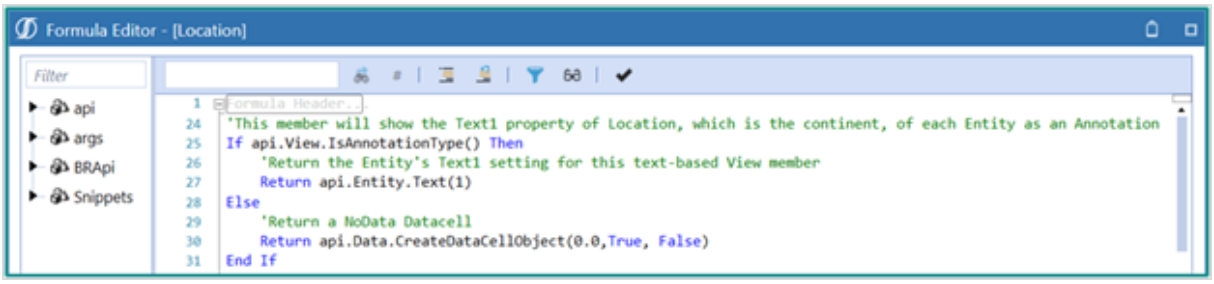
2. Open the formula editor for the member and ensure that the formula type is set to DynamicCalc.



3. Open the stored value.



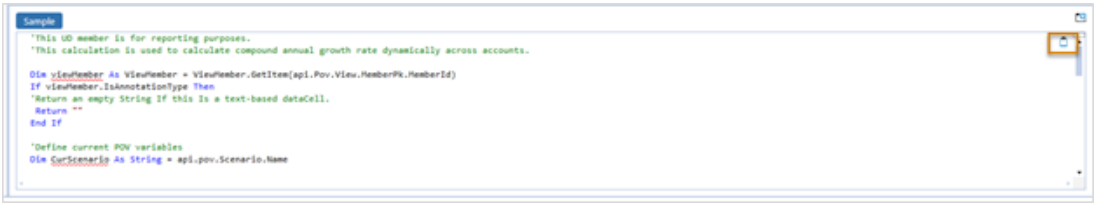
Now you can write a dynamic calculation. In this example, the formula will display an entity's Text1 value if an Annotation type View member is referenced.



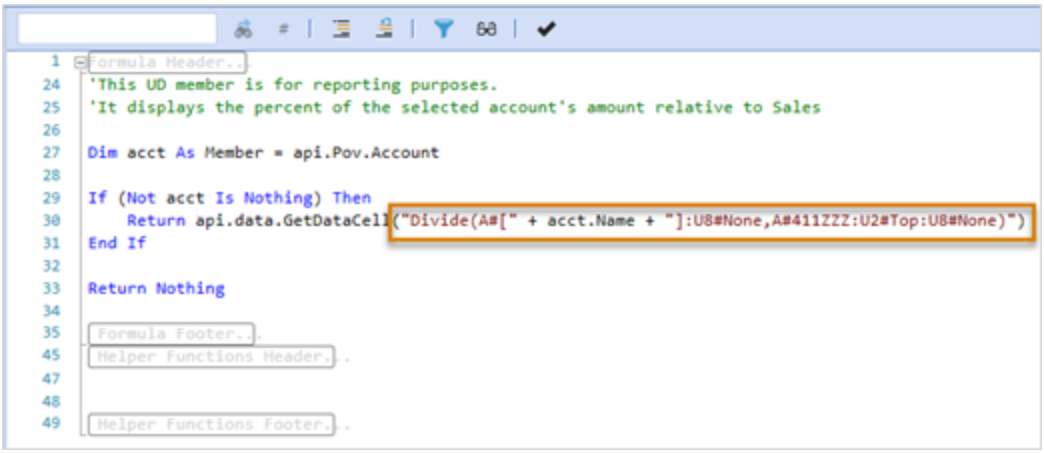
Write a Dynamic Calculation

You can write a simple dynamic calculation before adding it to a cube view.

1. Open the formula editor for the member that you want to update and navigate to **Snippets** > **Presentation Helpers** > **Reporting**.
2. Select the dynamic calculation you want to use.
3. Copy and paste the syntax.



4. Edit the dimensions of the pasted syntax to make sure they are correct.

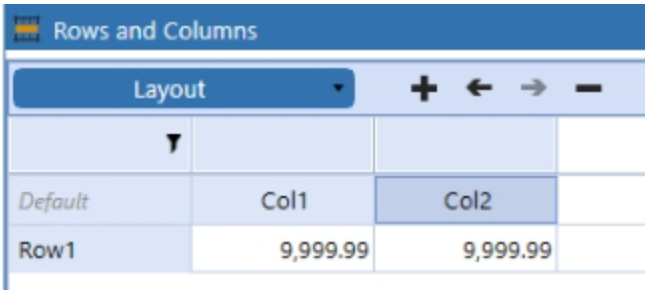


5. Double-check your syntax.
6. Click **OK**.
7. Close the Formula Member Property.
8. Save the member.

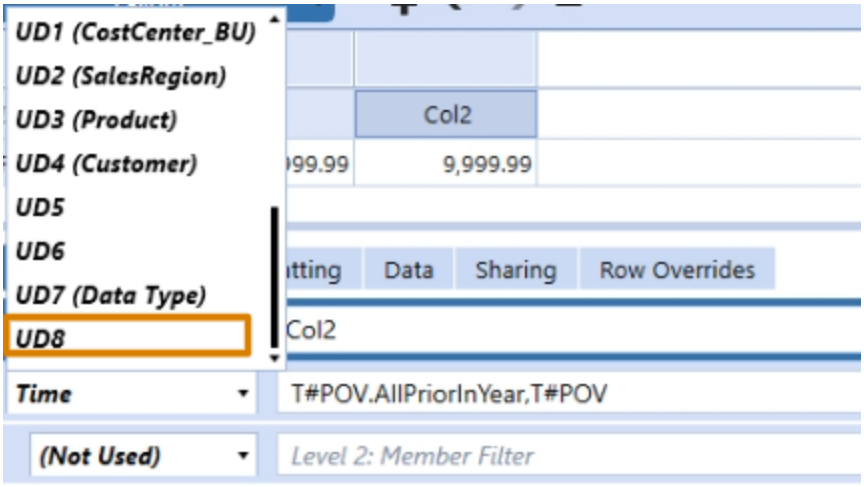
Add a Dynamic Calculation to a Cube View

After you define and write a dynamic calculation, you can add the calculation to a cube view.

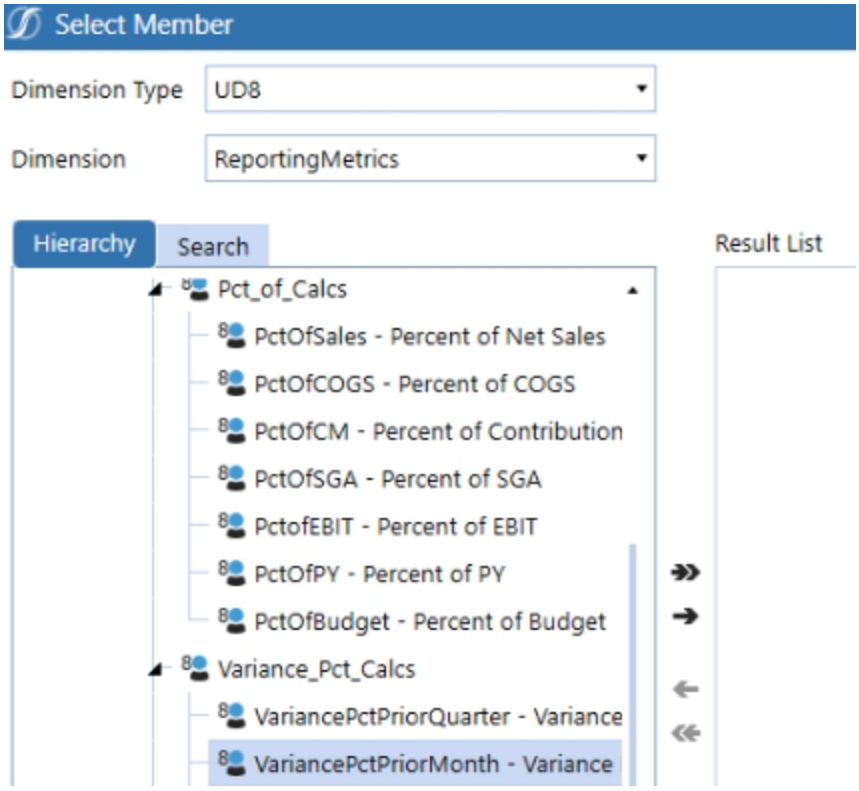
1. In OneStream, go to **Application** > **Presentation** > **Cube Views**.
2. Select a cube view group and then a cube view.
3. Create a new column. You will add a dynamic calculation to the second column. All the selections and properties of Col1 are carried over to the new Col2. Our reporting dynamic calculations are located in UD8. Therefore, the dimension will need to change.



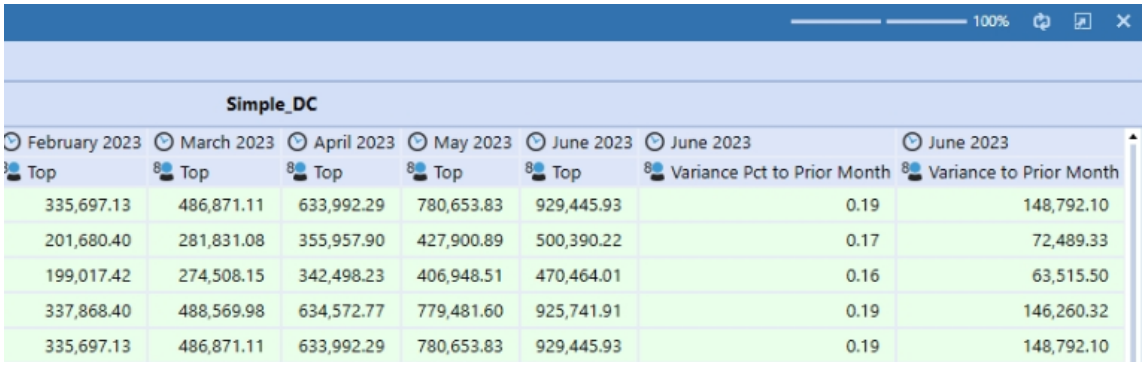
4. Open the column dimension drop-down and select UD8.



5. Click the Member Filter Builder.
6. Delete the current filter.
7. Select the U8# dimension token.
8. Select VariancePctPriorMonth and move the selection over.



9. Select the VariancePriorMonth member and move the selection over.
10. Click **OK**.
11. Click **OK** to confirm the Member Filter selection.
12. Save your changes.
13. Run the cube view as a Data Explorer Grid. The cube view renders and displays the dynamic calculations in the two far-right columns.



Simple_DC

February 2023 Top	March 2023 Top	April 2023 Top	May 2023 Top	June 2023 Top	June 2023 Variance Pct to Prior Month	June 2023 Variance to Prior Month
335,697.13	486,871.11	633,992.29	780,653.83	929,445.93	0.19	148,792.10
201,680.40	281,831.08	355,957.90	427,900.89	500,390.22	0.17	72,489.33
199,017.42	274,508.15	342,498.23	406,948.51	470,464.01	0.16	63,515.50
337,868.40	488,569.98	634,572.77	779,481.60	925,741.91	0.19	146,260.32
335,697.13	486,871.11	633,992.29	780,653.83	929,445.93	0.19	148,792.10

Override Complex Calculations

When you apply dynamic calculations to both rows and columns the row formula takes precedence, whether it is a dynamic calculation member or cube view math.

Rows and columns contain several override properties by row:

- When you use the row override in the column it will use the column formulas for the specified rows.
- When you use the column override in the rows it will use the formulas for the specified columns.

Apply Overrides

1. In the cube view, select a column.
2. Select the Row Overrides tab to apply row overrides in the column.
3. Define the row range. Enter a row name or apply a range of row names to override the results for the specific rows in the column of the cube view.
4. Enter the member filter to override the value for the row range. This defines what dimension member data displays or specifies complex calculations for the row range.
5. Set the format for the row range, which includes selecting fonts, text colors, cell background

color, grid lines, cell borders, and so on.

6. Select a value from the list parameter drop down list.

Row/Column Overrides

You can have up to four overrides for your column and four overrides for your row. These can be used to change the display for a given column or row. In this example, two columns for the Flash Scenario (Col1) vs. Budget (Col2) are needed. If Row1 is being used by the Net Sales Account (e.g. A#60999), and it needs to be compared to the Flash Net Sales which is a different account (e.g. A#FlashNetSales), then you would use a column override.

Row/Column Range
Type a column name to display a different value in the Cube View. You need to use the name of your rows or columns. The row range isn't looking for a numerical order, it's looking for the actual name of the rows and columns in your Cube View. In this example, Col1 would be typed to show a different value for Flash Scenario Column.

> **TIP:** Your Row Range can be a single row or column. To apply to multiple rows or columns, use a comma-separated list or use a hyphen to define a range.

Member Filter
Type the Member Filter to override the value for the columns or rows. In this example, type A#FlashNetSales to override the results from A#60999.

The override results will display the value from A#FlashNetSales in Col1 for the Flash Scenario.

Cell Format
See Cube View Formatting.

List Parameter
See List Parameter under Data earlier in this section.

Dashboards

Dashboards are a powerful way for users to view formatted cube views, pixel perfect financial reports and expressive charts. Dashboards are contained in a dashboard group and comprised of components, data adapters and potentially parameters. All of the objects are managed and shared across Dashboard Maintenance Units. Each is a building block on the other. Dashboard Maintenance Units are stored, created, and maintained in Workspaces. Workspaces vary by dashboard project needs and applications.

Workspaces store Dashboard Maintenance Units and facilitate community development by providing an isolated environment for developers to segregate and organize Dashboard objects. Workspaces are the foundation for dashboards and will continue to evolve into the framework that encapsulates the artifacts needed to develop business solutions.

The main use of a Dashboard Maintenance Unit is to enable the sharing of key dashboard artifacts like parameters, data adapters and components across multiple dashboard groups. These objects do not have security access settings, so they assume the settings of the maintenance unit. Once a maintenance unit is created, dashboard groups, components, data adapters, parameters, and files can be created within the unit.

Dashboard groups are created to organize dashboards and work as placeholder where the dashboards reside. The dashboard groups are then available to assign to dashboard profiles for viewing throughout the workflow process.

Dashboards are composed of components which are broken into parameter components, content components, and embedded components. Components can have one to many data adapters and can be used across multiple dashboards and dashboard groups in the same Dashboard Maintenance Unit.

When creating a component, give it a name and description. Feed parameters and other Substitution Variables into the description, so they display in the resulting component.

Account: |!TrendAccountParam!| Region: |!!Region Parameter!!| Year: |WFYear| Will appear as:Account: Net Income Region: Asia Pacific Year: 2012

Data adapters are the minimum building block for components. These specify where the data is coming from for the dashboard component. A list can be made of each data adapter which names the resulting table created when the dashboard runs.

Parameters can be used to filter data in the resulting dashboard. They are not required, but extremely useful. See Presenting Data With Books, Cube Views and Dashboards.

The File section allows an administrator to create company specific dashboards by uploading documents and images.

> **NOTE:** Optimal display size is 1920 x 1080 resolution. For the Windows desktop application, we recommend setting the scale to 100% and using the zoom functionality to zoom in. This is especially helpful in dashboards such as People Planning and Reporting Compliance.

The process works as follows:

Dashboards

Dashboards are contained in Dashboard Groups made up of Components, Data Adapters, potentially Parameters, and Files. All these objects are managed and shared across Maintenance Units which are then assigned to Dashboard Profiles.

Dashboard Toolbar

Create Group

Use this to create Dashboard Groups which is where all Dashboards are organized.

Create Profile

Use this to create a Dashboard Profile which is where all Dashboard Groups are organized.

Manage Profile Members

Use this to assign Dashboard Groups to Dashboard Profiles.

Create Workspace

Use this to create a workspace. Workspaces store maintenance units and facilitate community development by providing an isolated environment for developers to segregate and organize dashboard objects. See Workspaces

Create Maintenance Unit

All Dashboards and their corresponding Components, Data Adapters, Parameters, and Files are kept in a Maintenance Unit.

Create Dashboard

Use this to create a Dashboard.

> **NOTE:** A Maintenance Unit and a Dashboard Group need to be created first for this icon to be available.

Create Cube View

Use this to create a cube view.

Create Dashboard Component

Use this to create a Component to assign to a Dashboard.

Create Data Adapter

Use this to create a Data Adapter to assign to a Component.

Create Parameter

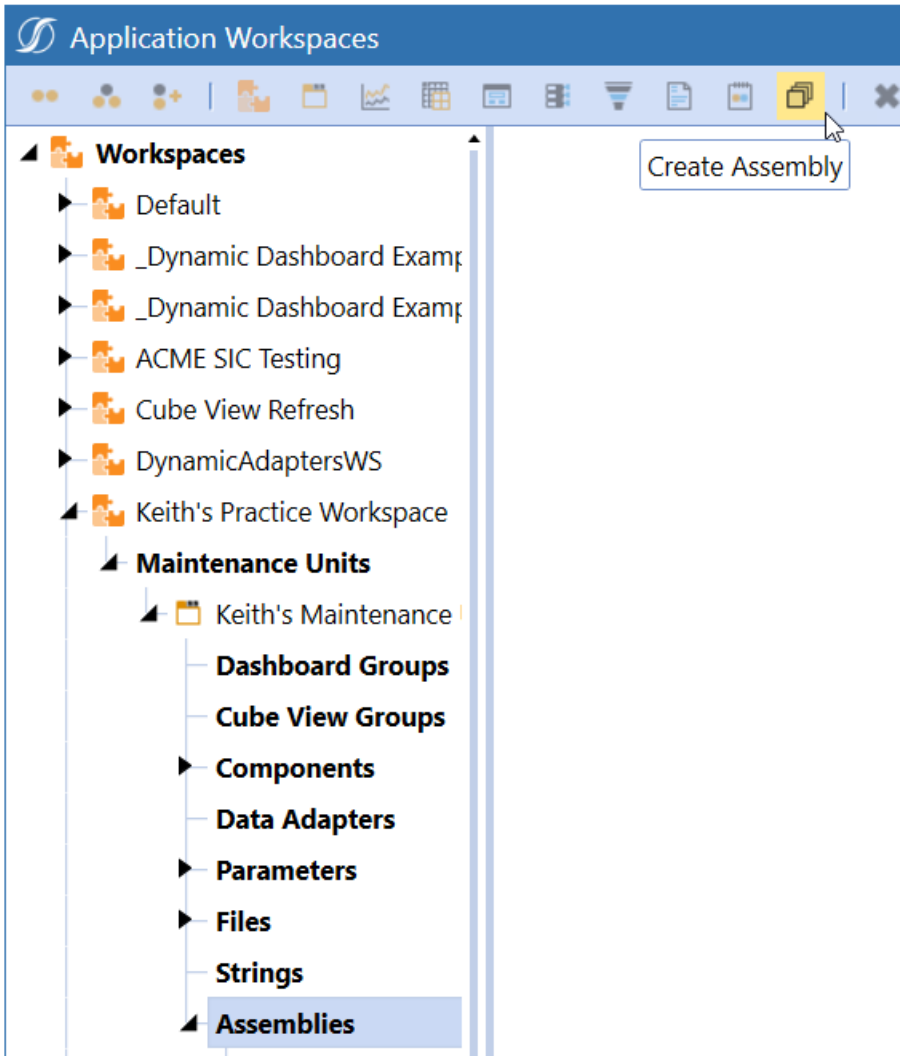
Use this to create a Parameter to use in a Dashboard or cube view.

Create File

Use this to store files to be used in a Dashboard.

Create String

Used with XFString functionality and displayed as a caption in a cube view.



Create Assembly

Use this to create an assembly.

Delete Selected Item

This allows the deletion of a single child item (e.g., Data Adapter or Component) within a Maintenance Unit, or the deletion of an entire Maintenance Unit including all child items.

Rename Select Item

Use this to rename the selected item.

Cancel All Changes Since Last Save

Use this to cancel any changes made to items since the last save.

Save

Use this to save all changes.

Copy Selected Dashboards, Components, Adapters, Parameters, Files, and Strings

Use this to create copy of the selected Dashboards, Components, Adapters, Parameters, Files, and Strings within or across Maintenance Units.

Multiple items can be selected all at once using Shift+click or Ctrl+click. Items can be copied and pasted multiple times and each time the item is pasted a suffix of _Copy will be added. For every additional instance of the Dashboard item that is pasted a suffix of _Copy (instance#) will be added.

> **NOTE:** Users can Copy using right-click on the Dashboard Item or toolbar menu Copy button.

Paste Selected Dashboards, Components, Adapters, Parameters, Files, and Strings

Use this to paste copies of selected Dashboards, Components, Adapters, Parameters, Files, and Strings within or across Maintenance Units.

Items can be pasted multiple times and each time the item is pasted a suffix of _Copy will be added. For every additional instance of the Dashboard item that is pasted a suffix of _Copy (instance#) will be added.

> **NOTE:** Users can Paste using right-click on the Dashboard Item or toolbar menu Paste button.

Search

Use this to open the Search Workspace Object dialog box and search for an object.

Add Dashboard Component

Use this to add a new dashboard component.

Move Up/Down

Use these to move components up or down in a list.

Remove Selected Dashboard Component

Use this to remove a dashboard component.

Open Data Explorer

Use this to open the Data Explorer dialog box

View Dashboard

Use this to run a Dashboard from the Application Tab.

View Dashboard in Design Mode

Select a Dashboard and click the black arrow in order to Set Selected Dashboard as Default. Click the icon to launch the default Dashboard in design mode. The Dashboard must be set as the default prior to viewing it in design mode. Select Clear Default Dashboard in order to remove the current saved Dashboard from the default state. See Dashboard Design Mode in Presenting Data With Extensible Documents for more details on this feature.

Show Objects that Reference the Selected Item

Use this to see other areas where the selected item is being used. For example, you can see where a parameter is being used in an application and therefore know the impact when you want to make a change.

Object Lookup

See Object Lookup.

Navigate to Security

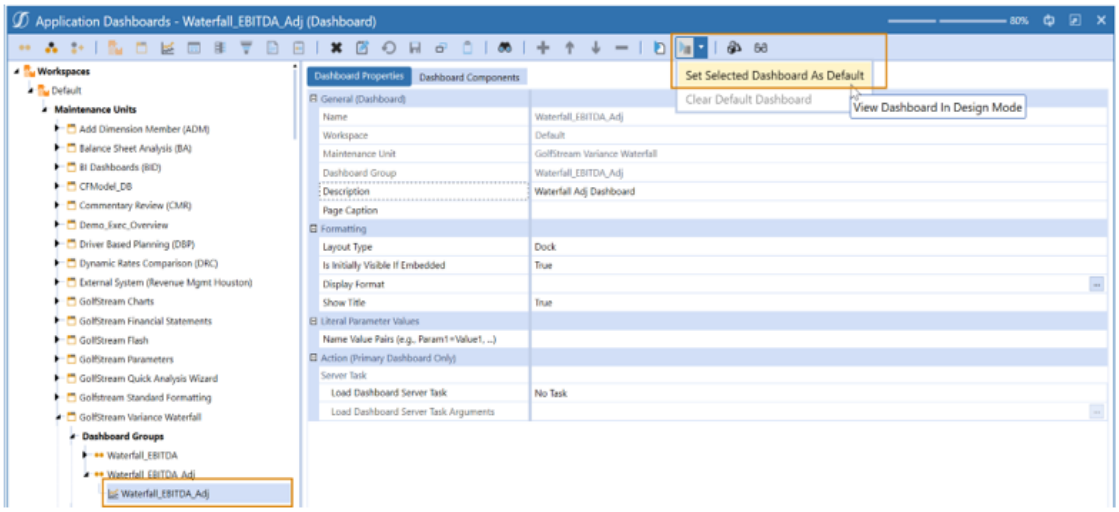
This icon appears in the Maintenance Group, Dashboard Group, and Profile Security properties and when clicked it navigates to the Security screen. This is an easy way to make changes to Security Users or Groups before assigning them to specific Maintenance Units, Groups, and Profiles.

Dashboard Design Mode

Dashboard design mode makes it easy to design and edit dashboards because it pinpoints exactly where the desired change needs to be made. Dashboard in design mode displays a dashboard draft and all the dashboard maintenance unit items associated with it. You can then select different sections of the maintenance unit hierarchy which highlight that portion of the completed dashboard, or select a portion of the dashboard and be directed to that exact item in the maintenance unit.

To run a dashboard in design mode:

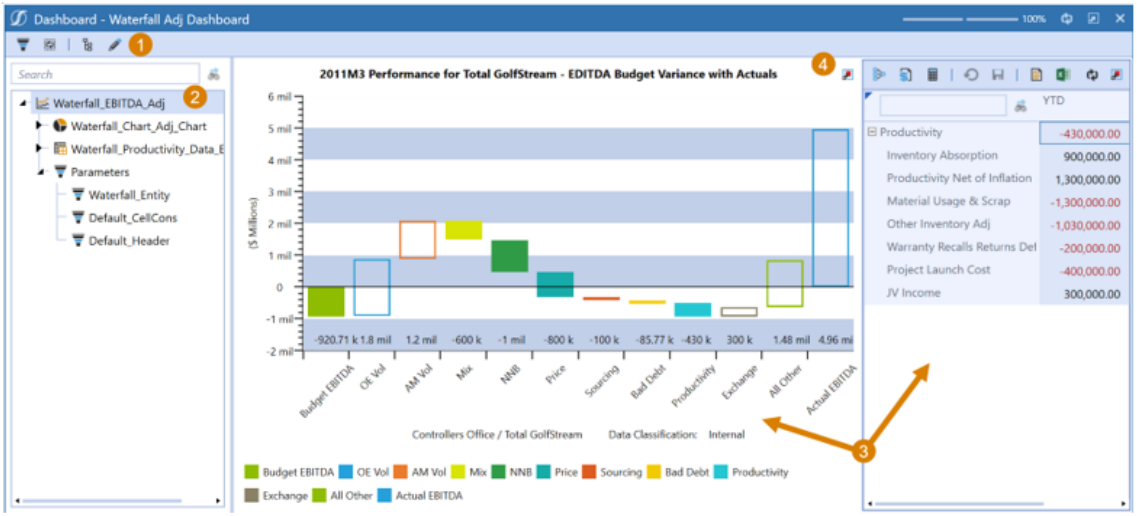
1. Select the dashboard to which changes need to be made and set it as the default dashboard.



2. Click to run the dashboard in design mode.

Design Mode

The following describes the design mode page elements.



> **TIP:** Highlight a specific dashboard item (such as a data adapter or component) and select this icon to go directly to that item in the dashboard maintenance screen.

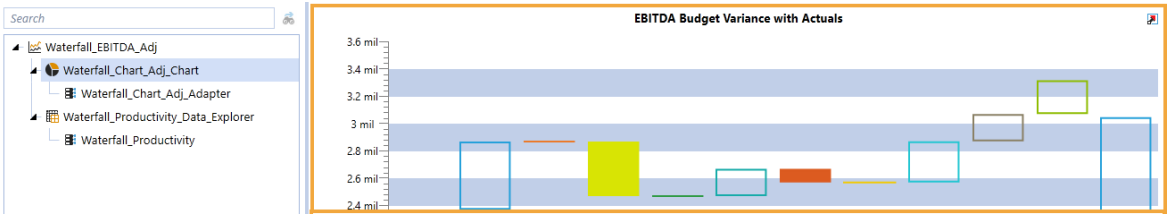
1. **Toolbar**:

   Click this to hide or display the Dashboard Maintenance Unit items.

   Click this to go back to maintenance mode.

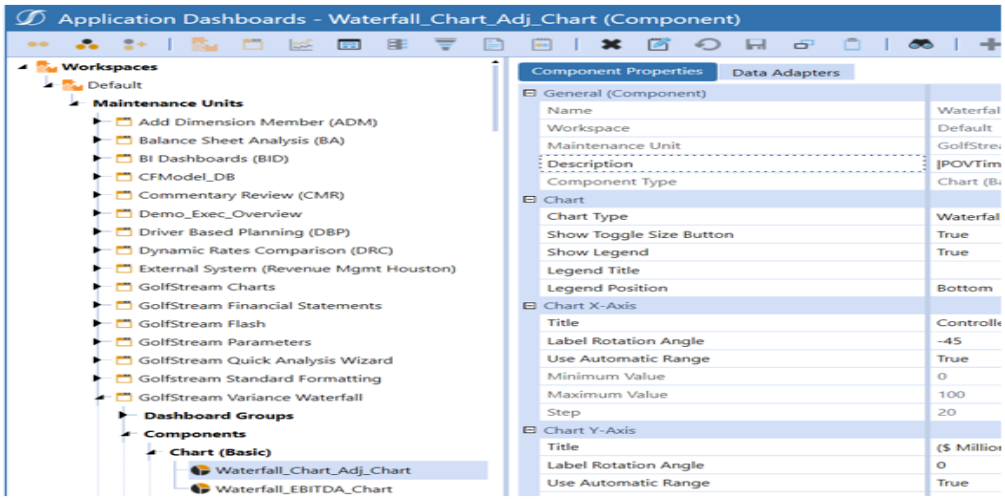
   > **TIP:** Highlight a specific dashboard item (such as a data adapter or component) and select this icon to go directly to that item in the dashboard maintenance screen.

2. **Dashboard Maintenance Unit items**: This provides the entire dashboard maintenance hierarchy for the default dashboard. For more complex dashboards, use the search to find the desired item. This dashboard example contains one chart component, one cube view component and the custom substitution variables used (parameters and their variables). Hover the mouse over each item to display a tool tip that identifies the value currently being used. Each component has its own data adapter.
3. **Dashboard Draft**: This is the default Dashboard.
4. **Find Tree Item** When a user clicks this icon, the maintenance items involved in this part of the Dashboard are highlighted in the tree. For example, to change the dashboad title above, click to see what part of the maintenance hierarchy controls the chart

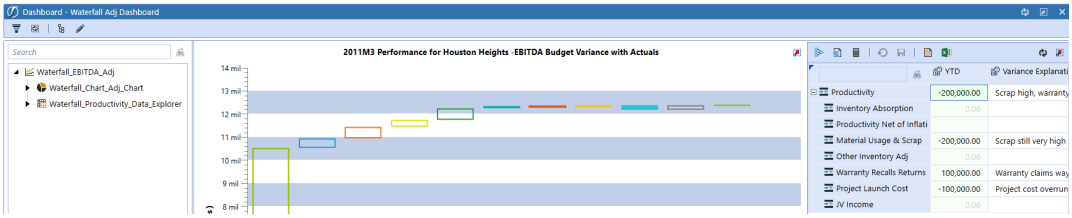


This indicates exactly where these changes can be made.

Click to return to edit mode and it navigates directly to the chart component selected in design mode



Make the necessary changes and preview it once more in design mode to see the outcome.

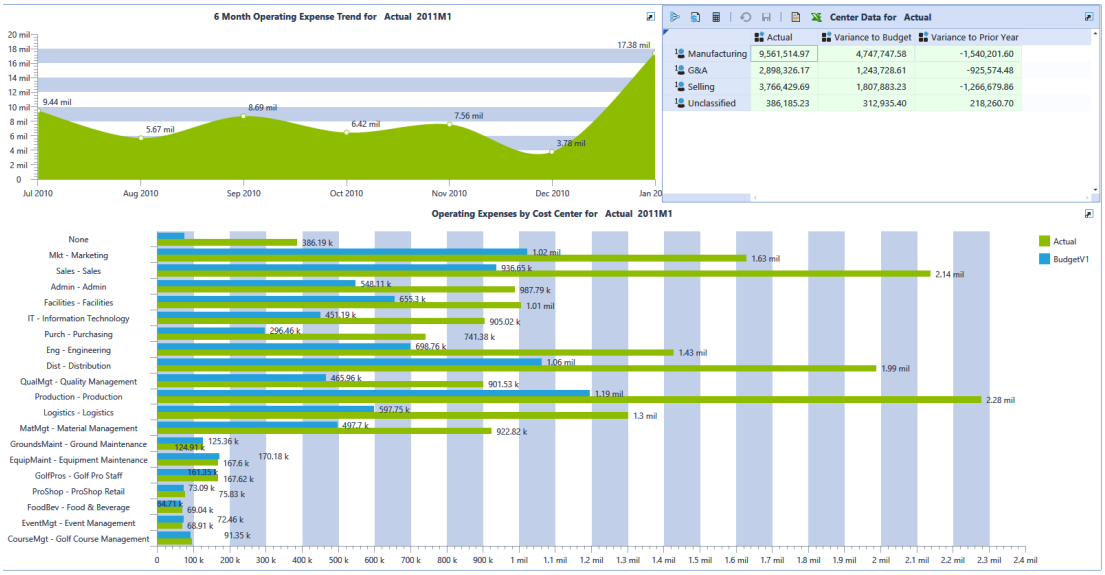


> **NOTE:** Authoring dashboards which are designed to contain more than six levels of nested dashboards is a rare use case and not recommended. When viewing in the Silverlight web browser, results may be inconsistent. If there is a desire to nest dashboards to this level or beyond, it is recommended to use the OneStream Windows application interface.

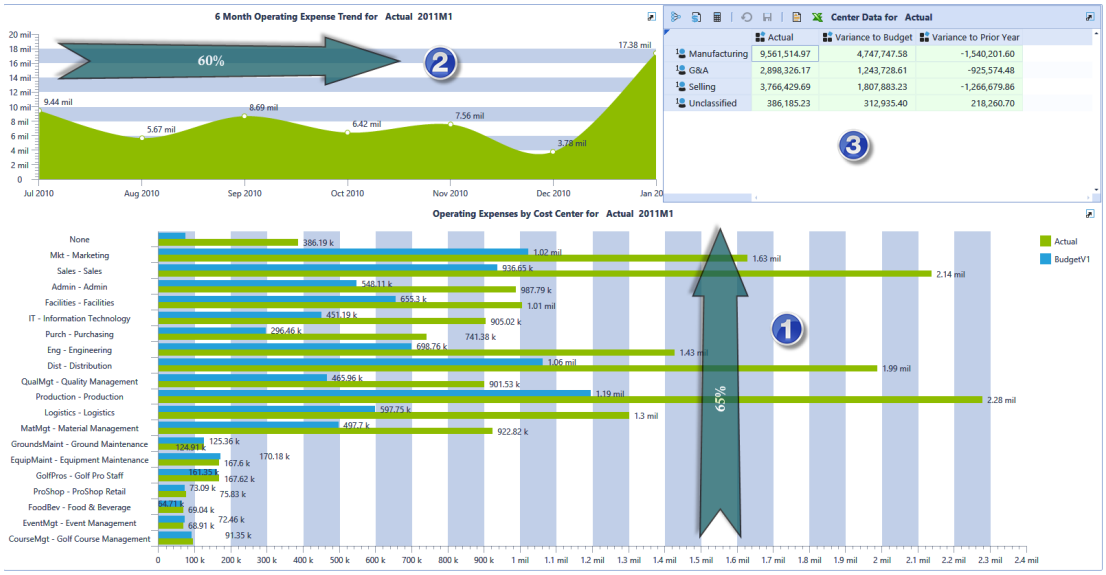
Preview Dashboards

Click to preview a completed Dashboard. Once in preview mode, Parameters may need to be entered, and then click Ok to get the Dashboard to run. Click to go back to Parameters and run the Dashboard again under a different input value.

Click to toggle back to design mode.



The example above shows a Dashboard with two Chart Components and one Data Explorer Component.



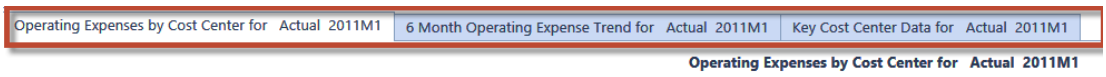
In this example above, there are three Components listed in the Dashboard Components tab in the order in which they appear on the screen.

Component 1 has the Dock Position set to Bottom and Height to 65%.

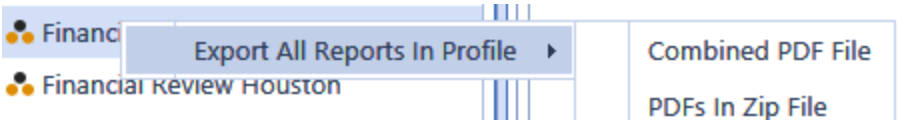
Component 2 has the Dock Position set to Left and Width to 60%.

Component 3 has the Dock Position set to any position such as Right and all other setting were left blank. The Dashboard will automatically dock this last Component to fill the rest of the available space.

Set layout type to Tabs in order to view each Component in its own tab.



By right clicking on a Profile under OnePlace Dashboards on the left in the Navigation Pane, a user can export all the Dashboards in the Profile out to a combined PDF file. Right click and choose Export All Reports in this Profile… and then choose Combined PDF File or PDFs in Zip File. This is a quick and easy way to produce Report books. The only exception is that currently graphs and charts do not export to PDF in these mass forms.



Dashboard Maintenance Unit Properties

Is Mobile

If set to True, the Dashboards within this Maintenance Unit can be used with the OneStream Mobile Solution. If set to False, the Maintenance Unit may only be used within the OneStream application. This property cannot be changed once the Maintenance Unit is saved.

> **NOTE:** Dashboards intended for OneStream Mobile have Dashboard Component limitations. These limitations are noted under Dashboard Components. Any Dashboard within a Mobile Dashboard Maintenance Unit can be launched from the Application Dashboard screen but cannot be assigned to Workflows or used in OnePlace.

Dashboard Group Properties

General (Dashboard Group Unit)

Name

The name of the Dashboard group.

Description

A short description of how the Dashboard group is used, or what it contains.

Security

Access Group
Members of this group can access the dashboards in the Dashboards group.

> **NOTE:** Click and begin typing the name of the Security group in the empty field. As the first few letters are typed, groups are filtered making it easier to find and select the desired group. Select a group, press CTRL and then double-click.

Dashboard Profile Properties

Once a Dashboard group is completed, it is organized into a dashboard profile. Profiles are then assigned to various functions. Click to create a new profile. Click to assign a group to a profile, allowing users to select the groups to be in a profile.

Dashboard Visibility controls what dashboards are viewed throughout the application. This is the key property for all dashboard profiles. The options are as follows:

Never
This is used for dashboards that are expired, or no longer being used.

Always
This allows the dashboard to be available in OnePlace and Workflow.

OnePlace
This allows the dashboard to be viewed in the Dashboard section under the OnePlace tab.

Workflow
This allows the dashboard to be attached to a workflow profile. This can be completed in Workflow Profiles under the Integration and Data Quality Settings.

Dashboard Properties

General

Name
The name of the dashboard.

Description
A quick description of the dashboard.

Page Caption
This is used to give a name to the dashboard when it is displayed. The name will be displayed at the top left of the dashboard. If the Page Caption is left blank, then it will default to the name given to the dashboard.

Dashboard Group
The Dashboard group to which it belongs.

Behavior

Dashboard Type
This indicates how each dashboard within a group is used. It helps reduce the number of groups needed and simplifies dashboard maintenance.

- **(Use Default)**: Default is **Top Level**.
- **Top Level**: This dashboard type indicates which dashboards in a group should be visible if assigned to OnePlace or a Workflow.
- **Top Level Without Parameter Prompts**: This dashboard type has the same behavior as Top Level, but if the dashboard has runtime parameters, it will have a slight performance improvement. It does not prompt for parameters.

- **Embedded**: This dashboard type indicates which dashboards in a group should not be visible in OnePlace because they are dependent on other dashboard objects.
- **Embedded Dynamic**: This dashboard type enables you to dynamically modify dashboard properties, component properties, parameters, adapters, and cube view properties. See Dynamic Dashboards.
- **Embedded Dynamic Repeater**: This dashboard type enables you to dynamically modify dashboard properties, component properties, parameters, adapters, and cube view properties. This dashboard type can be used to add multiple instances of the same component without having to re-create each component individually. See Embedded Dynamic Repeater Dashboard.
- **Embedded Top Level**: This dashboard type acts as a new top level with its own parameter resolution and rendering stages. As a result, it allows for the Load Dashboard Server Task to be run even when embedded.

  > **NOTE:** Currently, Embedded Top Level dashboards should not be used with Smart Links.

- **Embedded Top Level Without Parameter Prompts**: This dashboard type acts as a new top level with its own parameter resolution and rendering stages. As a result, it allows for the Load Dashboard Server Task to be run even when embedded. It does not prompt for parameters, resulting in a slight performance improvement.

  > **NOTE:** Currently, Embedded Top Level Without Parameter Prompts dashboards should not be used with Smart Links.

- **Custom Control**: This dashboard type has the same behavior as Embedded, but each instance maintains its own set of input parameters, variables, and Event Listeners (actions).

Custom Control

A new dashboard type with required input parameters, which is a comma separated list of parameter names that must be provided by the containing dashboard when processing a custom control dashboard. For example, Param 1, Param 2, Param 3. A parameter component inside the custom control forwards events to the parent dashboard.

Formatting

Layout Type

Canvas

A component docks to the top, bottom, right, or left of the dashboard. The component only shows in the small space defined by the dock position and does not display fully on a screen. Slide bars are used to view the components.

Dock

A component docks to the top, bottom, right, or left of the dashboard. Use Dock Position to determine how much of the screen will be used for this layout.

Grid

This allows the ability to make a dashboard in a grid format by using rows and columns to add Components, Lines, or Moveable Splitters.

Horizontal Stack Panel

Each Component will display horizontally. Horizontal Stack Panel dashboards are generally used as part of other dashboards.

Tabs

Each component will display in its own tab.

Uniform

Each Component is docked, but all are equal in size.

Vertical Stack Panel
Each component will display vertically. Vertical Stack Panel dashboards are generally used as part of other dashboards.

Wrap
See Canvas

Is initially Visible if Embedded
When set to True, the selected dashboard will be hidden if it is retrieved from another dashboard. The hidden dashboard can be set to visible from a Parameter component by listing its name in the dashboard to Show property under User Interface Action and selecting Refresh for the dashboard containing the embedded dashboard.

Display Format
This allows unique display formatting per dashboard for background color, dialog width and height, and border formatting. See "Display Format Settings".

Literal Parameter Values

Name Value Pairs (e.g., Param1=Value1,…)
Enter a comma-separated list of name-value pairs for this dashboard's static parameter values. Piped variable names cannot be used in this setting.

Action (Primary Dashboard Only)

Load Dashboard Server Task
Dashboard Extender Rules can run upon launching and refreshing dashboards. This only applies to the main dashboard, not embedded dashboards.

No Task
A Dashboard Extender business rule is not being used on this dashboard.

Execute Dashboard Extender Business Rule (Once)
This runs the assigned Business Rule once upon refresh or initial launch

Execute Dashboard Extender Business Rule (All Actions)
This does an iterative action and runs the business rule as many times as needed.

Load Dashboard Server Task Arguments
Enter the arguments required by the server task. When executing a Dashboard Extender business rule, the arguments include {BusinessRuleName}{FunctionName}{Optional Named-Value Pairs} each enclosed in curly brackets. Click the ellipsis and select the desired argument format and then enter each object.

```
Example: {DashboardLoadBRName}{TestFunction}{Param1=[Value1], Param2=[Value2]}
```

Grid Layout Type

This only becomes available if Grid is chosen as the Layout Type.

Number of Rows
The number of rows the dashboard will use for the grid layout.

Number of Columns
The number of columns the dashboard will use for the grid layout.

Row/Column Type

Component
Use the Component option if a Content or Parameter component is needed for the row or column of the grid layout for the dashboard.

Line
Use the Line option if a line is needed for the row or column on the dashboard.

Moveable Splitter
Use the Moveable Splitter option if one is needed for the row or column on the dashboard. This allows the ability to move the grid up and down, or left and right.

Row/Column Height
Utilize Row/Column Height to customize the sizing of the rows and columns heights and widths to be displayed on the dashboard. Auto expands the size based on the contents of the row. You can use an * to create a fractional value of the remaining space. For example, if there are three rows, then each row will take a third of the space.

Display Format Settings

General

Background Color
This is used to display the background color of one dashboard.

InnerBackgroundColor
This is used to define the background color of a dashboard containing embedded dashboards.

Dialog

DialogDisplayStyle
Controls how a dashboard is sized when viewed as a dialog box. If the specified value exceeds the current screen, the value will be changed to the maximum display setting for height and width to fit to screen.

Specify **Maximize** or **Normal**. The default is **Normal**. **Maximize** opens using the Dialog Maximize mode to full screen. **Normal** opens to the defined Dialog Height and Width settings.

DialogWidth
Customize the width of the dialog for the dashboard. The dialog width supports up to 10,000. If set to -1, the size will set at run time to the maximum width for the active screen.

DialogHeight

Customize the height of the dialog for the dashboard. The dialog height supports up to 10,000. If set to -1, the size will set at run time to the maximum height for the active screen.

EnableSystemClose

This is used to control the display and use of the Close button of the dialog. Setting this to True removes the Close button.

Styles

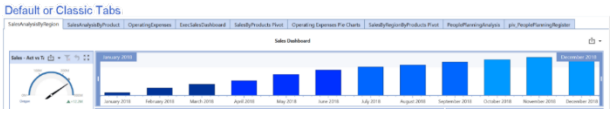
TabControlStyle

Choose from Classic, No Border, or Rounded Corners to define how tabs appear in the application. The tab control functionality remains the same. TabControlStyle is only available in the Windows version.

Dashboard Tabs

There are three different tab styles that can be selected for dashboards:

1. Use Default or Classic – The original style for dashboard tabs.



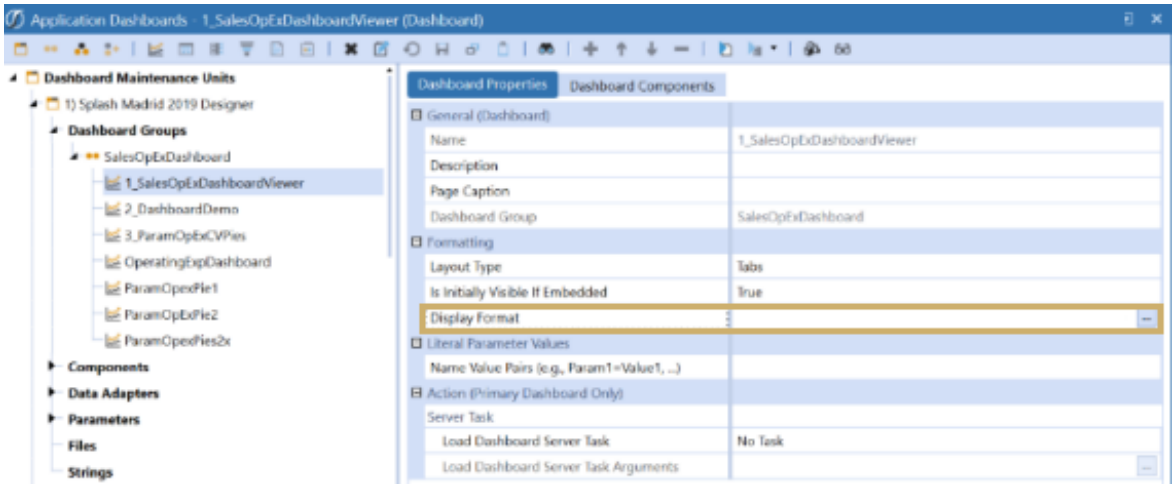
2. No Border – Rounded corners and no borders.

3. Rounded Corners – Rounded on the edges with borders.

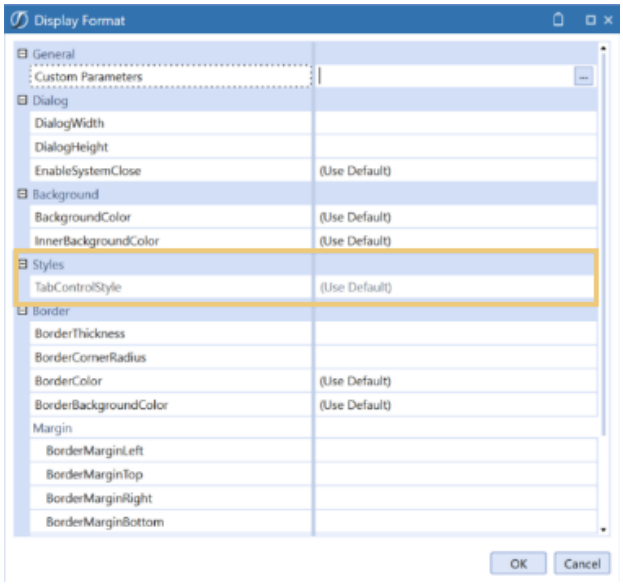
These tabs are only available for use with the dashboards and do not apply to the rest of the product.

Select a Tab

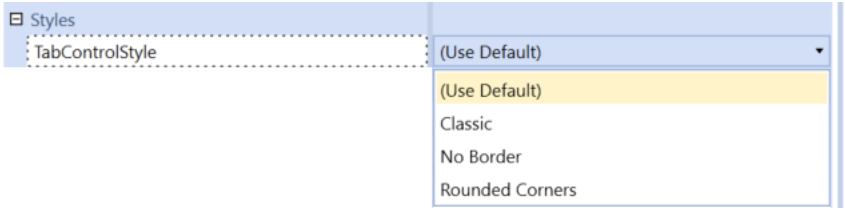
1. Go to **Application > Dashboards** and select a dashboard.
2. Click to open the dashboard and in **Dashboard Properties > Formatting** make sure the **Layout Type** is **Tabs**.
3. Go to Display Format and click the ellipsis to edit.



4. Click **Styles > TabControlStyle** and click in the field to get a menu option. If the Layout Type is not set for Tabs, the field is unavailable.

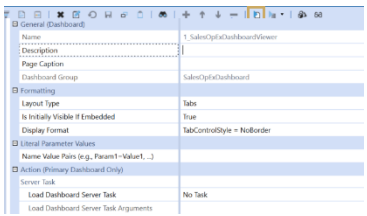


5. Choose the type of tab for the dashboard.

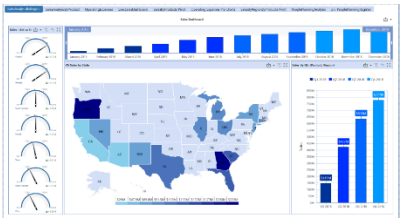


6. Click **OK** and then **Save**.

8. Click **View Dashboard**.



9. The tabs have no borders.



Border

BorderThickness

This is used to customize the border thickness for the dashboard.

BorderCornerRadius

This is used to customize the border corner shape for the dashboard. If left blank, the border will be square.

BorderColor

This is used to customize the outer border coloring for the dashboard.

BorderBackgroundColor

This is used to specify the border background color of a dashboard containing embedded dashboards.

Margin

BorderMarginLeft/Top/Right/Bottom

Used to customize the margin of the border on the dashboard.

Padding

BorderPaddingLeft/Top/Right/Bottom

Used to customize the padding or spacing of the border on the dashboard.

Dashboard Components

General

Name

The name of the dashboard component.

Description

A free form field to enter a description of the dashboard component.

Components

Add Dashboard Component

Use this to add and assign a dashboard component.

Move Up

Use this to move a dashboard component up in the Components list.

Move Down

Use this to move a dashboard component down in the Components list.

Remove Selected Dashboard Component

Use this to remove a selected dashboard component from a Components list.

> **NOTE:** To select and add multiple components, press CTRL and then click to select. Once all items are selected, click OK.

Position on Dashboard

Dock Position

Left/Top/Right/Bottom
Allows the ability to dock a Component in a specific position on the dashboard.

Left
This sets the amount of dashboard space to allocate to the left dock position.

Top
This sets the amount of dashboard space to allocate to the top dock position.

Width
Enter a percentage to set the width of the dashboard.

Height
Enter a percentage to set the height of the dashboard.

Component Display Format Properties

The following properties are available under the Display Format field for Parameter and Content components. Not all properties in this section are available for all components. For information about parameters, see Filter Runtime Data and Reduce Maintenance with Parameters.

General

Custom Parameters
Click the ellipsis in order to select and assign a custom parameter such as a cube view style to the cube view header.

IsVisible

This determines if the component is visible on the dashboard. Settings are True, False, or Use Default.

ShowSearch

This adjusts the behavior of the ComboBox component. When True, the ComboBox retains typed text and auto complete is enabled. When False, typed text is not retained and auto complete is disabled.

HorizontalAlignment

This determines how the component should be aligned horizontally on the area allocated in the dashboard.

Left
The component is aligned on the left of the allocated area.

Center
The component is aligned in the center of the allocated area.

Right
The component is aligned on the right of the allocated area.

Stretch
The component is stretched to fill the allocated area.

Use Default
Use the default setting for horizontal alignment.

VerticalAlignment
This determines how the component should be aligned vertically on the area allocated in the dashboard.

Top

The component is aligned on the top of the allocated area.

Center

The component is aligned in the center of the allocated area.

Bottom

The component is aligned on the bottom of the allocated area.

Stretch

The component is stretched to fill the allocated area.

Use Default

Use the default setting for vertical alignment.

Width

The width of the control.

Height

The height of the control.

MarginLeft, Top, Right, and Bottom

The spacing used on the sides of the control.

BorderThickness

The thickness of the border used when displaying the control.

FontFamily

The font type used on the component (e.g., Ariel, Times Roman, etc.).

FontSize

The font size used on the control text.

Bold

If set to True, the font will be bold.

Italic
If set to True, the font will be italicized.

Color
The color of the text displayed in the control.

TextColor
Choose a color from the drop-down options for the control text.

BackgroundColor
Choose a color from the drop-down options for the text background.

BorderColor
Choose color from the drop-down options for the border of the control.

HoverColor
Choose color from the drop-down options to change the hover color of a button dashboard component.

Image

HorizontalContentAlignment
This determines how the image's content should be aligned horizontally on the area allocated in the dashboard.

VerticalContentAlignment
This determines how the Image's content should be aligned vertically on the area allocated in the dashboard.

ImageStretch
This setting allows the image to be re-sized when being displayed.

None
This does not re-size the image.

Fill
The image stretched to fill the control.

Uniform
This preserves the aspect ratio of the image.

UniformtoFill
This scales the source image to fit within the bounds of the image object and keeps the source image centered within the image object.

ImageCropLeft, Top, Width, Height
Use these settings to crop and position the image in order to align the desired content appropriately.

Label

LabelPosition
The location where the control label is placed.

None
This does not display a label for the control.

Left
This displays the label to the left of the control.

Top
This displays the label above the control.

Right
This displays the label to the right of the control.

Bottom
This displays the label below the control.

Use Default
Use the default label display location

LabelFontFamily
The font type to use on the label (e.g., Ariel, Times Roman, etc.).

LabelFontSize
The font size to use on the label.

LabelBold
If set to True, the font is bold.

LabelItalic
If set to True, the font is italicized.

Colors

LabelZeroOffsetForFormatting
This holds a number with the default value of 0.0. When a label is being used to display a number, and the number is greater than the number in this field, it will be displayed using the color associated with LabelTextColor. Otherwise, the number will be displayed using the color associated with LabelNegativeTextColor.

LabelTextColor
Choose a color from the drop-down options for the label text.

LabelNegativeTextColor
Choose a color from the drop-down options for the label's negative text.

Parameter Components

Parameter components are objects such as a button or list box that, when selected, could drive additional actions, such as opening a different dashboard or running a Custom Calculate.

Custom calculate executed from a Dashboard Parameter component allow the passing of a complete set of name value pairs that align to the dimension Point of View (i.e. Entity=MyEntity,Time=2017M11,Consolidation=Local, etc) and the custom calculation processor will use those variables to initialize the Finance Engine API POV Variables (i.e. api.POV.Entity.Name will equal MyEntity). All API.POV objects (Cube, Entity, Parent, Consolidation, Scenario, Time, View, Account, Flow, Origin, IC, UD1, UD2, UD3, UD4, UD5, UD6, UD7, UD8) are initialized and available to this function. This makes it possible to share custom calculation rules with standard consolidation/calculation rules because both rule sets have a fully initialized set of API.POV variables. For consolidation business rules, this POV is related to the data unit. When using a Parameter component with a cube view on the same dashboard, API.POV is related to the 18-dimension POV that is available from each cell rendered. See "Custom Calculate" under Finance Function Types for more information.

Button

This displays a button on a dashboard. Buttons can be used to initiate a variety of functions including business rules, dashboards, file uploads and so on. This component can be used with dashboards.

Formatting

Text:
The text to be displayed with the button. This is an optional field. If left blank, the Text field will default to display the button name. To clear the default Name from displaying, enter |SPACE|.

Tool Tip
The text to be displayed when hovering over the button.

Display Format
The formatting assigned to the control. See Component Display Format Properties.

Image File Source Type
The Image (if any) displayed on the button.

URL

The image displayed on the button is based on a URL.

Workspace File

The image displayed on the button is in a file stored in the Workspace Maintenance Unit File section.

Application Database File

The image displayed on the button is in a file in the Application Database Share.

System Database File

The image displayed on the button is in a file in the System Database Share.

File Share File

The image displayed on the button is in the File Share.

URL or Full Name File

The URL, file path or file name for the displayed image.

URL Syntax

Access a built-in toolbar image, or use an image from any web server

Built-in OneStream Image Example

/OneStream.ClientImages;component/Misc/ToolbarImage.png.

Web Server Example

http://www.onestreamsoftware.com/img/onestream-logo.png

Database File System Example

Enter the path and file name Documents/Public/TestImage.png.

Workspace File Syntax

Enter the name of the file resource TestImage.png.

The following Excel properties are used to specify an Excel file from either an Extensible Document or Report Book in order to display it as an image or button.

Excel Page Number
Enter the page number of the Excel document to display.

Excel Sheet
Enter the name of the Excel worksheet to display.

Excel Named Range
Enter the named range from Excel to display. The named range must be in the specified Excel sheet in the Excel Sheet property, or the Excel Sheet property must be blank in order to use a named range. See Image under Component Display Format Properties to position and align the Excel image as needed.

Button

Button Type
Specify the functionality to be assigned for the selected button.

Standard

Displays a generic button that can have custom functionality assigned such as display a dashboard, execute a business rule, and so on.

File Explorer

This sets the button to display the File Explorer dialog.

File Upload

This sets the button to display the File Upload dialog.

Select Member

This sets the button to display the Select Member dialog.

Dimension Type

Specify the Dimension Type. This is required.

Use All Dimensions for Dimension Type

When **True**, it overrides the Cube Restriction and displays all dimension types. When **False**, the dimensions are limited to defined dimension type.

> **NOTE:** If no dimension type is defined, the selections are reverted to the default dimension type.

Dimension

Specify a specific dimension. This is optional.

> **Example:** HoustonSalesRegionalBudget

Cube

Specify a cube. This is optional.

Member Filter

Specify a Member Filter. This is optional.

> **Example:** U3#Root.TreeDescendantInclusive

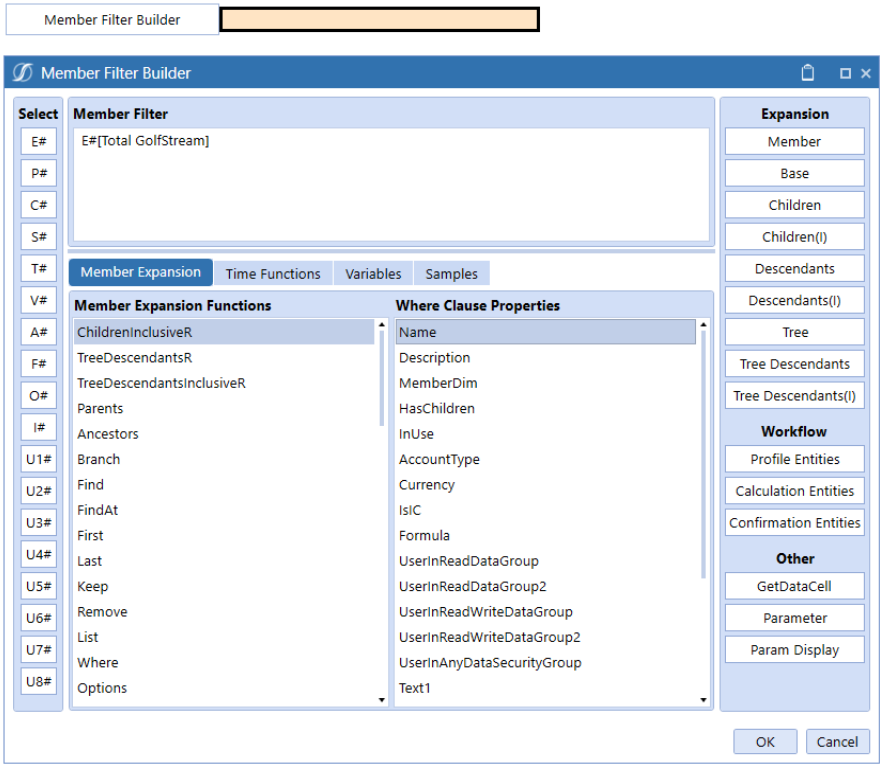
Member Filter Builder

This sets the button to display the Member Filter Builder dialogue.

Default Member Filter

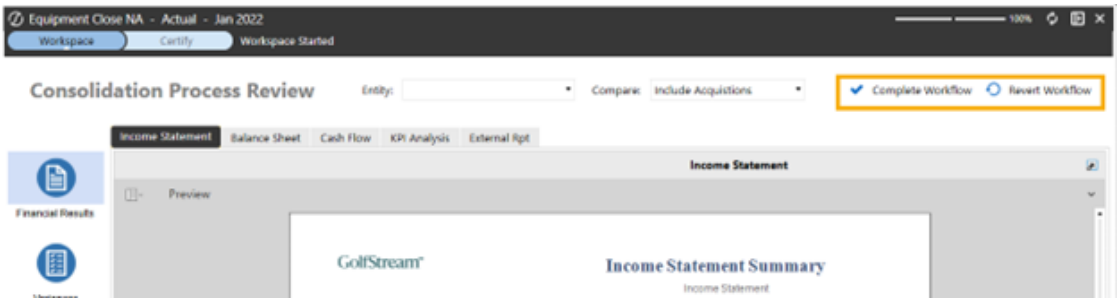
Specify default member filter to display upon opening. This is optional.

> **Example:** E#[Total Golfstream]



Workflow

This sets the button for Complete and Revert Workflow, replacing the need for a custom business rule.



Workflow Button Action Type

Select **Complete Workflow** to enable the Complete action button or **Revert Workflow** to enable the Revert action button.

Workflow Button Image Type

Select one of the standard button images, **Use Button Image** to use a custom image, or **Not Used** to not use a button image.

Action

Bound Parameter

This is the name of the parameter the component will represent. Piped variable names are not available in this setting. The component will use the parameter to get its list of options. The selection will specify a value for the parameter when it is referenced elsewhere via piped variable naming. Click and begin typing the name of the parameter in the blank field. As the first few letters are typed, the names are filtered making it easier to find and select the one desired. If the name is unknown, expand a Workspace Maintenance Unit and scroll to identify a parameter. Then press CTRL and double-click.

Parameter Value for Button Click (Button Component Only)

The value or string to pass into the Bound parameter.

Apply Selected Value to Current Dashboard

If set to True, the Bound Parameter on the current dashboard will use the current updated value or use the value prior to update. If set to False, the current dashboard will not be updated.

Save Action

Selection Changed Save Action

What occurs when the button is pressed.

No Action

When the Parameter components are changed, changes will not be saved.

Save Data for Components
This will save the data modified by the Dashboard components specified in the Selection Changed Save Arguments property.

Prompt to Save Data for Components
This will ask the user if the modified data should be saved or not for a defined list of Dashboard components.

Save Data for all Components
This will save all data modified by each of the Dashboard components assigned to the dashboard. These components do not have to be specified in the Selection Changed Save Arguments property.

Prompt to Save Data for All Components
This will ask the user if the modified data on any included dashboard component should be saved or not.

Save Data for Components and Save Files
This will save the components modified by the Dashboard components specified in the Selection Changed Save Arguments property and their files.

Prompt to Save Data for Components and Save Files
This will ask the user if the modified data should be saved or not for the defined list of Dashboard components and their files.

Save Data for All Components and Save Files
This will save all components modified by the Dashboard components included in the Dashboard and their files.

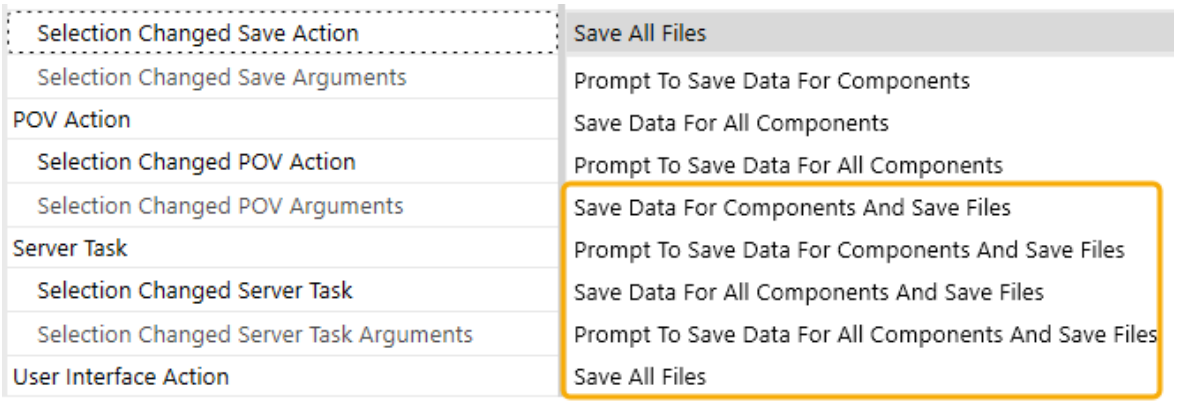
Prompt to Save Data for All Components and Save Files
This will ask the user of the modified data should be saved or not for all dashboard components and their files.

Save All Files

This will save all modified file data within the dashboard and exclude saving any modified component data.

> **NOTE:** The Selection Changed Save Argument field is only active when you choose a Selection Changed Save Action that requires you to specify components.

Save Files Properties



You can use these Save Files properties to save edits made in Text Editor and Spreadsheet Component files through a dashboard button. For example, the user has a dashboard containing a spreadsheet and button component, and the developer sets the button component's Selection Changed Save Action property to one of the Save Files options. When the user edits the file data in the spreadsheet component's file and then clicks the dashboard button, all file changes are saved. The dashboard component changes will also be saved. Component changes are only not saved if the Save All Files property is selected, which only saves modified file data. Users can utilize these Save Files properties through button components instead of having to navigate to **File** > **Save** in the component's toolbar to save file changes.

Considerations

- Since Text Editor components cannot have nested components, the Save Data For Components And Save files, Save Data For All Components And Save Files, and Save All Files options will only save the Text Editor component's file.
- When working in a spreadsheet component embedded in a dashboard, the Save Files properties enable users to save nested components while also saving the spreadsheet file. When the dashboard button is set to Save All files, only spreadsheet file modifications are saved.

USE CASE

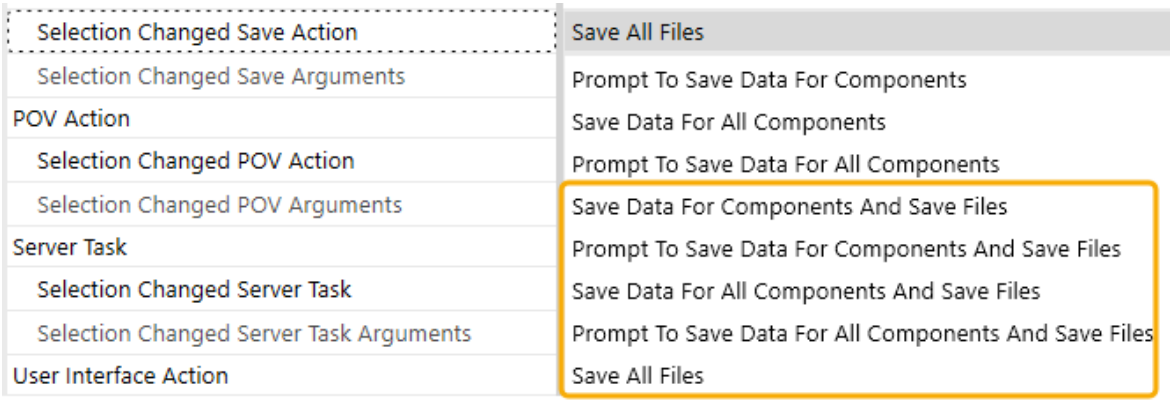
Add a Save Files Action to the Narrative Reporting Dashboard Button

Role	Benefits
Developer: sets dashboard button properties User: utilizes the dashboard button	The developer can streamline the save process for users in Narrative Reporting by reducing the possibility of users checking in files without saving.

A user is in the Narrative Reporting solution which utilizes a Dashboard Text Editor component that displays the content of a specific file, and a developer wants to give users the ability to save their file edits using the Check In button to prevent them from checking in their file without saving. The developer also wants to do this so that users will not have to navigate to the Text Editor's **File** > **Save** toolbar button before clicking **Check In**.

1. Navigate to the correct Maintenance Unit that contains the Narrative Reporting solution's Check In button component.
2. Go to **Component Properties** > **Action** > **Save Action** > **Selected Changed Save Action**.

3. Select one of the Save Files save actions from the **Selection Changed Save Action** drop-down menu. If you select Save Data For Components And Save Files or Prompt To Save Data For Components And Save Files, you need to specify the Text Editor Component's name in the **Selection Changed Save Arguments** field.



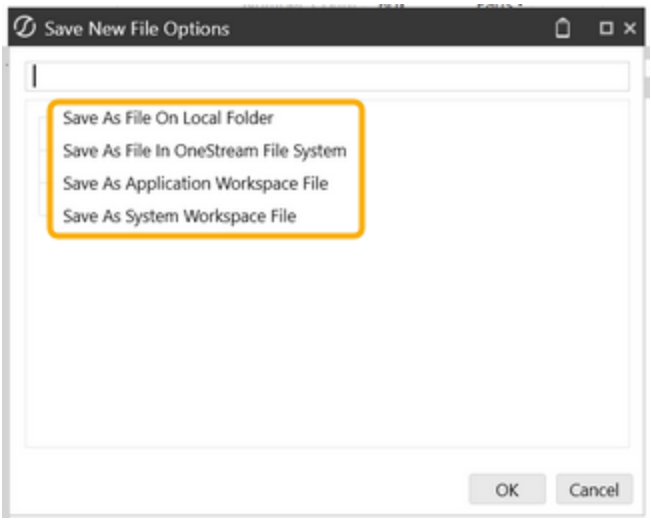
4. Click the **Save** button.

With one of these options selected, when the user clicks the **Check In** button, it will perform the same save action as the Text Editor's **File** > **Save** toolbar button in the ribbon and successfully save any file modifications.

Save As

If the Text Editor or Spreadsheet components are not linked to or displaying an existing file, clicking a button component with the Save File options opens the Save New File Options window. This enables users to save a new file in different locations and performs the same function as the components' **File** > **Save As** toolbar button action.

> **NOTE:** If you have already used the button component to Save As and created a new file, when you select the same button component again, the updates made in the dashboard component will save to that initially created file. It will not prompt the Save New File Options window to create a second new file until the user refreshes or reloads the dashboard.



Local Folder

Select location on local computer/network to save a file to.

> **NOTE:** The Save As File on Local Folder option is not available in the Modern Browser Experience.

OneStream File System

Select location within OneStream File Explorer to save a file to.

Application Workspace File

Select Application Workspace location to save a file to.

System Dashboard File

Select System Workspace location to save a file to.

POV Action

Selection Changed POV Action

No Action When Parameter components are changed, no change will be made to the user's POV.

Change POV

This setting will change the user's POV to a specified POV when Parameter components are changed.

Change Workflow

This changes the user's Workflow to a specified Workflow when Parameter components are changed.

Change POV and Workflow

This changes the Workflow and POV to a specified value when Parameter components are changed.

Selection Changed POV Arguments

Enter the arguments for the selected POV action.

Change POV Example

Cube=Corporate, Entity=|!MyEntityParam!|, Scenario=Actual

Change Workflow Example

Enter a value for WFProfile, WFScenario, and/or WFTime.

WFProfile=|!MyWFProfileParam!|, WFScenario=Actual, WFTime=2012M1

Change POV and Workflow

See above for examples

> **NOTE:** The Selection Changed POV Action must be set to something other than No Action for this change to occur.

Server Task

Selection Changed Server Task

No Task
When Parameter components are changed, no server task is executed.

Execute Dashboard Extender Business Rule (server type)
Executes the assigned Business Rule when Parameter components are changed. For a long-running business rule, it is optionally able to run on a specific server type, such as a Consolidation server type. Here are the options:

- Execute Dashboard Extender Business Rule (General Server)
- Execute Dashboard Extender Business Rule (Stage Server)
- Execute Dashboard Extender Business Rule (Consolidation Server)
- Execute Dashboard Extender Business Rule (Data Management Server)

Execute Finance Custom Calculate Business Rule
Executes a single year custom calculation from a Finance business rule when Parameter components are changed. The calculation is performed on the cube view assigned to the dashboard Data Adapter and calculates time based on the cube view Time member set in the Point of View. This is considered a partial calculation and does not store the calculated data or run the calculation during a consolidation. Running a custom calculation from a dashboard will impact calculation status for the affected data unit even if the data does not change. A calculated dimension member can call this business rule to do a full stored calculation.

See About the Financial Model for an example of this calculation logic.

Execute Data Management Sequence
Executes the assigned Data Management Sequence when Parameter components are changed.

Calculate
Executes a Calculation when Parameter components are changed.

Force Calculate

Executes a Force Calculate when Parameter components are changed.

Calculate with Logging

Executes a Calculate with Logging when Parameter components are changed.

Force Calculate with Logging

Executes a Force Calculate with Logging when Parameter components are changed.

Translate

Executes Translate when Parameter components are changed.

Force Translate

Executes Force Translate when Parameter components are changed.

Translate with Logging

Executes Translate with Logging when Parameter components are changed.

Force Translate with Logging

Executes Force Translate with Logging when Parameter components are changed.

Consolidate

Executes Consolidate when Parameter components are changed.

Force Consolidate

Executes Force Consolidate when Parameter components are changed.

Consolidate with Logging

Executes Consolidate with Logging when Parameter components are changed.

Force Consolidate with Logging

Executes Force Consolidate with Logging when Parameter components are changed.

> **TIP:** For more details on these calculation types, see Data Management in Application Tools.

Selection Changed Server Task Arguments

Enter the Parameter components to which the Save action applies. If Save Data for all Components or Prompt to Save Data for All Components is selected, this can be left blank.

Dashboard Extender Business Rule Example (each enclosed with curly braces)

{Business Rule Name}{Function Name}{Optional Name-Value Pairs}

Finance Custom Calculate Business Rule Example (each enclosed with curly braces)

{Business Rule Name}{Function Name}{Optional Name-Value Pairs}

> **NOTE:** Name Value Pairs can have settings for Cube, Parent and any Dimension Type. It can also have a setting for any Time Member Filter including CurrentPeriod, AllPriorInYearInclusive or AllInYear. It can have any other name-value pair for use by the Business Rule since they are passed into the rule using the CustomCalculateArgs object.

Data Management Sequence Example (each enclosed with curly braces)

{Sequence Name}{Optional Name-Value Pairs}

Calculation, Translation or Consolidation Example (each enclosed with curly braces)

{Data Unit Member Script}

{E#|!EntityParam!|:C#Local:S#Actual:T#|!TimeParam!|}

User Interface Action

Selection Changed User Interface Action

No Action

When Parameter components are changed, no user interface action is executed.

Redraw

This repaints the screen without making a connection to the database acquiring updated data.

Refresh

This refreshes the dashboard specified in the Dashboards to Redraw property which makes a connection back to the database acquiring all new data.

Close Dialog

This closes the dialog upon performing a specified action in a dashboard.

Close Dialog As OK

This closes the dialog after the user clicks OK.

Close Dialog As Cancel

This closes the dialog after the user clicks Cancel.

> **NOTE:** The refresh of the parent dashboard is based on performing a specified action setup in a dashboard. For example, if a dashboard is set to "Open Dialog With No Buttons and Refresh," the "Close Dialog As OK" will refresh the parent dashboard and run the UI action.

Close All Dialogs

This closes all open dialogs upon performing a specified action in a dashboard.

Open Dialog

Opens a Dashboard in a modal window.

Open Dialog With No Buttons

Performs an Open Dialog action and does not display the Close button.

Open Dialog With No Buttons and Redraw

Performs an Open Dialog, allows the user to apply changes, and then Redraw without displaying the close button.

Open Dialog With No Buttons and Refresh

Performs both an Open Dialog & Refresh action without displaying the close button.

Open Dialog and Redraw
Performs both an Open Dialog & Redraw action.

Open Dialog and Refresh
Performs both an Open Dialog & Refresh action.

Open Dialog, Apply Changes, and Redraw
Performs an Open Dialog, allows the user to apply changes, and then Redraw.

Open Dialog, Apply Changes, and Refresh
Performs an Open Dialog, allows the user to apply changes, and then Refresh.

Open Dialog, Apply Changes, and Redraw if OK
Performs an Open Dialog, allows the user to apply changes, and then Redraws once the user selects OK.

Open Dialog, Apply Changes, and Refresh if OK
Performs an Open Dialog, allows the user to apply changes, and then Refreshes once the user selects OK.

Dashboards to Redraw
Comma separated list of embedded dashboards to redraw when the user changes the selection in this Parameter Component.

Dashboards to Show
Comma separated list of embedded dashboards to make visible when the user changes the selection in this Parameter component. This is typically used when multiple Parameter components are needed to collect settings to display an embedded content dashboard. After the last selection has been made, that Parameter Component will show the previously hidden embedded content dashboard.

Dashboard to Hide
Comma separated list of embedded Dashboards to hide when parameter components are changed.

Dashboards to Open in Dialog

Single Dashboard to open in a modal dialog when the selected button is clicked.

Dialog Initial Parameter Values

Enter a comma separated list of name-value pairs to specify initial parameter values for the opened dialog. For example: DlgParam1=Value1, DlgParam2=[Value 2].

Dialog Input Parameter Map

Enter a comma separated list of name-value pairs to map parameter names used by this component's dashboard to parameter names used by the opened dialog. For example, DlgParamName1 = This ParamName1, DlgParamName2=ThisParamName2. When the dialog is opened, the names of the parameters adjust to accommodate the dialog.

Dialog Output Parameter Map

Enter a comma separated list of name-value pairs to map parameter names modified by the closed dialog to parameter names used by this component's dashboard. For example, ThisParamName1=DlgParamName1, ThisParamName2=DlgParamName2. When the dialog is closed, the names of the parameters adjust to accommodate this component's dashboard.

Navigation Action

Selection Changed Navigation Action.

No Action

No navigation action is performed when Parameter components are changed.

Open File

When Parameter components are changed by a user, a new file will open. This is specified in the Selection Changed Navigation Arguments field.

Open Page

When Parameter components are changed by a user, a new page will open. This is specified in the Selection Changed Navigation Arguments field.

Open Website

When Parameter components are changed by user, a website will open in a new browser. This is specified in the Selection Changed Navigation Arguments.

Selection Changed Navigation Arguments

Enter the arguments for the selected navigation.

Open Page Example

Specify the page name followed by a colon and the page type's arguments separated by two ampersands

XFPage=Dashboard:NameofDashboard or XFPage=Workflow:NameofWorkflow

Open Page in New Tab Example

An option in the Open Page Navigation type allows dashboards launched from within a dashboard to launch into a new tab. Valid values are True and False.

XFPage=Dashboard:MyDashboardName, OpenInNewXFPage=True

Open File or Website Example

Specify the file source type (e.g., Dashboard, Application, System, FileShare or URL) and the file name. Also use PinNavName=True/False or PinPOVName=True/False to open or close the Navigation or POV panes.

FileSourceType=FileShare, FullFileName=Document/Public/NameofFile, OpeninXFPageIfPossible=False

> **NOTE:** Click ... and begin typing the argument string in the blank field. As the first few letters are typed, the names are filtered making it easier to find and select the one desired. If the name is unknown, scroll through the list to select the correct one. Once the argument string is selected, click CTRL and Double Click. This will enter the correct string into the appropriate field.

Chart (Advanced)

See Advanced Chart Examples in Presenting Data With Extensible Documents for some examples on the Diagram Types and associated properties.

Action

See Button.

Diagram Type

Polar2D

This is used to display data as a circular graph and displays values on the basis of angles.

Radar2D

This is used to display data as a circular graph and has multiple axes along which data can be plotted.

Simple2D

This is commonly used to compare percentage values of different point arguments in the same series.

XY2D

This is commonly used to show filled areas on a diagram. Values can be used separately or aggregated.

> **NOTE:** The properties below do not apply to all of the Diagram Types.

Show Toggle Size Button

This controls whether the toggle icon displays on the Dashboard at runtime.

Swap Axes

If set to True, this exchanges the X-axis with the Y-axis.

Domain Color

This is the chart's background color.

Show Point Labels

If set to True, this displays data values above each data series on the chart.

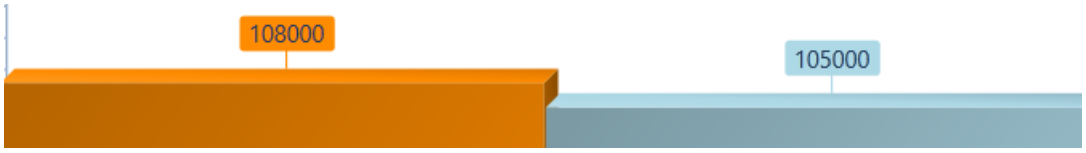
Point Label Text Format

Within curly braces enter a Value Identifier and optionally a colon followed by a Format String. The Value Identifier can be S for series name, V for value, or A for argument (argument is typically used for the X-axis text format). Format Strings specify how to display the series name, value, or argument.

For example, to display a value to two decimal places enter {V:0.00} where 0.00 is the Format String. Another common example is {V:0,,M} which will display the value in the millions or {V:0,T} which will display the value in thousands.

In order to apply a standard number format to a value enter {V:NumberFormat}.

For example, {V} displays a value as follows:



Applying a number format after the colon displays the value in a desired format.
{V:#,###,0}

Any text written within the curly braces after the Format String will be displayed as text. Additionally, any text written before or after the curly braces will also be displayed as text. Refer to MSDN's topic on Format Strings for additional information.

Enable Animations

This controls how the chart renders at runtime. Set this to True to include chart animations upon launching the dashboard.

Show Border

Set this to True in order to display a border around the chart.

Use Clockwise Rotation

Set this to True to display the legend in clockwise order.

Show Legend

Set this to True in order to create and display a customized legend for the Dashboard chart.

Title

Enter the legend title.

Vertical/Horizontal Position

This controls where on the screen the legend will display.

Orientation

This determines whether the legend displays vertically or horizontally.

Show Check Boxes

If set to True, this creates a check box next to each legend item. Users can then uncheck/check specific data points in order to hide/display them while viewing the dashboard chart.

Show Border

Set this to True in order to display a border around the legend.

Crosshair Enabled

A crosshair is similar to a point label, but only displays when the user hovers over the data series. Set this to True, in order to enable the crosshair function.

Show Crosshair Lines

Identifies where the axis hits per data point.

Show Crosshair Labels

This displays the crosshair data in a pop-up box.

Crosshair Label Mode

Show Common for all Series

This displays the crosshair data for the entire chart no matter where the user is hovering.

Show for Each Series

This displays crosshair data for each series simultaneously.

Show for Nearest Series

This only displays crosshair data when the user is hovering on a specific series.

Crosshair Label Text Format

See Point Label Text Format.

Show X-Axis/Show Y-Axis

Set this to True in order to display the X and Y-axis. To disable either axis, set the appropriate property to False.

Title

Enter a title for each axis.

Argument Type (X-Axis Only)

This specifies the type of information feeding into the chart's arguments.

DateTime

Use this if dates are included in the arguments. This causes the chart's x-axis to be continuous meaning the x-axis ticks are independent of the point's argument.

Double

This causes the chart's x-axis to be continuous meaning the x-axis ticks are independent of the point's argument.

String

This causes the chart to be discrete meaning the axis ticks are dependent on the point's arguments.

Text Format

See Point Label Text Format.

Label Rotation Angle

Enter a value to rotate the X or Y-axis labels. The default is 0 resulting in no label rotation.

Logarithmic

Setting this to True will force the axis values to follow a logarithmic pattern specified in the Logarithmic Base property below.

Logarithmic Base

Set the Logarithmic Base value. For example, logarithmic values with base 10 yields the axis values 0, 1, 10, 100, 1000, 10000, etc.

Use Automatic Range

If this is set to True, this sets the minimum and maximum values automatically depending on the points contained in the chart. In order to customize the minimum/maximum value, set this to False and specify the values in the properties below.

Minimum/Maximum Value

Specify the minimum and maximum range values.

Use Automatic Step

Set this to True in order to create the tick marks by certain interval values automatically. Set this to False to specify the specific intervals for the tick marks.

Step

Specify the intervals for the tick marks.

Reverse Order

By default, the order of values on a chart begins at the bottom and works its way to the top, so the first color in the legend is the bottom value's color. Setting this property to True will reverse that order. When the order is reversed, the values are displayed top to bottom, thus following the order of the legend.

Interlaced

Set this to True for the chart's domain color to alternate between the interlaced color. If this is set on the x-axis, the color will display horizontally. If this is set on the y-axis, the color will display vertically.

Interlaced Color

Enter the desired interlaced color.

Show Grid Lines

Set this to True to display grid lines at each data point on the axis.

Show Minor Grid Lines

Set this to True to display grid lines at each tick mark on the chart's axis.

Polar2D Types

PolarArea

The data series displays as a filled area on a circular diagram.

PolarLine

The data series displays in a continuous line on a circular diagram.

PolarPoint

The data series displays in a series of small circles on a circular diagram.

Radar2D Types

RadarArea
The data series displays as a filled area on a circular grid with multiple axes along which data can be plotted.

RadarLine
The data series displays as a line on a circular grid with multiple axes along which data can be plotted.

RadarPoint
The data series displays as a series of small circles on a circular grid with multiple axes along which data can be plotted.

Simple2D Types

Funnel
A funnel chart displays a wide area at the top, indicating the total points' value, while other areas are proportionally smaller.

NestedDonut
This is similar to a Doughnut Chart but compares the data with one doughnut nested in another one.

PieAndDonut
This is used to compare the percentage values of different data points in the same series.

XY2D Types

AreaRange
This displays the data series as filled areas on a diagram, with two data points that define minimum and maximum limits. This chart is used to illustrate the difference between start and end values.

BarRangeOverlapped/Waterfall

This displays either vertical or horizontal bars along the Y axis (the axis of values). Each bar represents a range of data with two values. This chart type is used to show activity from different data series one above another to compare.

BarRangeSideBySide/Waterfall

This displays either vertical or horizontal bars along the Y axis (the axis of values). Each bar represents a range of data with two values. This chart type is used to show activity from different data series grouped by their settings.

BarSideBySide

This displays the data series as individual bars where the height of each bar is determined by the data value.

BarSideBySideFull/SideStacked

This can stack different bars and combine them into groups displayed side-by-side across the same axis value.

LineScatter

This is a type of Line Chart where the data points are connected by a continuous line.

LineStep

This is used to display to what extent values have changed for different points in the same series.

Point

This is used to show points from two or more different series on the same chart plot.

Stock

This is used to show variation in stock prices over the course of time.

> **NOTE:** See Chart (Basic) under Content Components for all other XY2D Chart Types.

Model Display

Change the look of the series by choosing between three tiers (Basic, Moderate, Advanced) to determine which preset model the chart series will use. Selecting Moderate or Advanced provides a more enhanced view of the chart.

Show Markers

This is used in conjunction with Line Charts to display a circular marker at each end of the line. Set this to True to display line markers.

Marker Size

Enter a value for the marker size. Values are entered in pixels.

Bar Width

This is used in conjunction with Bar Charts. Enter a value to control the width of each bar on the chart.

Line Thickness

Enter a value measured in pixels to control the line thickness displayed in a Line Chart.

Line Style Type

This determines whether a Line Chart will display Solid or Dashed lines.

Pie Hole Radius Percent

This is used in conjunction with Pie and Donut Charts and controls how large the middle hole is. A 0 value will result in no hole.

Series 1-10 Color

Select a color for each Data Series. For example, if a chart is displaying data for an Actual and Budget Scenario, data displayed for Actual would be one data series and data displayed for Budget would be a second data series. In this example, a color can be entered into Series 1 Color and Series 2 Color. By default, the order of values on a chart begins at the bottom and works its way to the top, so the first color in the legend is the bottom value's color. See the Reverse Order property for details on changing this order.

Series 1-10 Point Colors

Enter a color for the data points within a specific data series. If there are several data points and each one needs to be a different color, enter a comma separated list of colors. If there are more data points in the series than specified colors, the data points will cycle through the colors and repeat as needed.

Data Series Source Type

Cube View

A Cube View from the chart's Data Adapter is being used as the data source.

Business Rule

A Dashboard DataSet Business Rule from the chart's Data Adapter is being used as the data source. Utilizing a Business Rule gives more control to the chart's data series and provides additional customized settings.

Suppress Zeros

If set to True, this removes all results that are zero.

Row List Type

This specifies the rows to include from the chart's Data Adapter.

All Rows

The chart will include all rows.

First Row

The chart will only include the first row.

Row Index List

The chart will include the list of rows specified in the Row Index List field.

Row Index List

Enter the list of rows to include in the chart.

Cube View Data Point Legend Type

Default
The legend text comes from the row headers.

Annotation
The legend text comes from the Annotation View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Assumptions
The legend text comes from the Assumptions View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Audit Comment
The legend text comes from the AuditComment View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Footnote
The legend text comes from the Footnote View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Variance Explanation
The legend text comes from the VarianceExplanation View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Check Box

This provides a small box that when selected is filled with a check mark. This component can be used with dashboards.

Formatting
See Button.

Action
See Button.

Cube View

Use this component to attach a Cube View to a dashboard.

Cube View

Click the and select the desired Cube View Name.

Show Header

Select True in order to display the Cube View's Page Caption, select False in order to hide it.

Show Toggle Size Button

Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Actions

For all Action settings and examples, see Button earlier in this section.

> **NOTE:** In order to enable Component Actions, the Cube View must have Navigation Link Parameter(s) configured otherwise, the actions will not function.

Combo Box

This displays a drop-down list of strings. This component can be used with dashboards.

Formatting

See Button.

Action

See Button.

Gantt View

This is a graphical illustration of a schedule over time and is used for relational or hierarchical data (non-Cube data). This component can use a SQL query to query the data, but in order to fill these components with data, a Dashboard Data Set Business Rule must be used within a Data Adapter. The Business Rule must use an XFGanttTaskCollection object that helps the user convert a normal data source into the object-based data source required to feed these new hierarchical components. See Gantt View Component Business Rule Example in Implementing Security for an example of the Dashboard Data Set Business Rule.

Action

See Button.

Show Header

When set to True, the grid Name/Description will display as the Component's header on the Dashboard, when set to False, it will be hidden.

Show Toggle Size Button

Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Grid View

This displays data from a Data Adapter in a grid. Similar to the SQL Table Editor, but data is not editable.

> **NOTE:** The Grid View dashboard component should not be used for large data sets as it is not a component that uses paging. In this instance a "SQL Table Editor" component should be used to display the data to the end user and it will use paging to return the full data set.

Grid View Multiselect

Allow Multiselect will generate the Bound Parameter as a comma delimited list.

Use one of the following selection methods:

- Select all rows with the Select All checkbox.
- Select multiple rows using the Shift key.
- Select individual rows with Ctrl-click or select each row individually.

Deselect each row individually or click the Deselect All toolbar button.

Show Column Settings Button

Shows (if set to True) or Hides (if set to False) the Column Setting Button which allows user to reorder and set visibility of columns.

Formatting

See Button.

Action

See Button.

Grid View

Table Name

The name of the table created in the data Parameter.

Show Header

When set to True, the grid Name/Description will display as the Component's header on the Dashboard, when set to False, it's hidden.

Show Column Settings Button

When set to True, Column Setting **will show on the Dashboard, if set to False, it's hidden.**

Allow Column Reorder

When set to True, you can drag columns and put them in a different order, if set to False, you can't drag them to put them in a different order.

Show Toggle Size Button

Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Show Group Panel

When set to True, the column grouping option will be shown on the Dashboard. Active users can drag and drop column headers to a group area to apply grouping on the data in the grid. If set to False, it's hidden.

Show Row Headers

When set to True, the grid row headers will be shown on the Dashboard, if set to False, it's hidden.

Show Column Headers

When set to True, the grid column headers will be shown on the Dashboard, if set to False, it's hidden.

Show Horizontal Grid Lines

When set to True, the horizontal grid lines will be shown on the Dashboard, if set to False, it's hidden.

Show Vertical Grid Lines

When set to True, the vertical grid lines will be shown on the Dashboard, if set to False, it's hidden.

Default for Columns are Visible

If set to True, all columns will be hidden unless overridden on an individual column.

Retain Table Column Order

Default value which allows reordering of the columns displayed. If this setting is True, then the column order can be changed via the Grid View Column Format section.

Column Name for Bound Parameter

Specify the name of the database column to be used to change the value of the Bound Parameter when a database row is selected. This is used when a SQL Table Editor is being used to affect the display of other Dashboard components. (e.g., when showing detailed information for the selected row.)

Save State

Enables User Preferences for Column: Order, Visibility, Filtering, Sorting and Width. If this setting is True, then the column activity can be changed and saved via the Grid View Column Format section.

Vary Save State By

Apply Vary Save State By to workflow profile settings and scenario. The current Save State elements will have the control state tied to the current Workflow Profile and Workflow Scenario settings. If you select **False**or **Use Default**, Vary By Save State will not apply.

When Vary Save State By is enabled, the related Dashboard Component will have the additional reset option of **Reset All States**, which can be used to clear the user's Save States across all the Vary Save State By parameters.

Grid View Column Format

Column Name

Column name from the table where the selected formatting is to be applied. Specified name must match the column name from the table exactly.

Description

Column description to be displayed. By default, the column name from the table is displayed.

Column Display Type

Standard
This displays the default data in the column.

OneStreamClientImage
This replaces the data in the column with the associated image.

WorkflowStatusImage
This replaces the data in the column with the associated Workflow status indicator image.

IsVisible
Setting to override the default Columns Visible setting. Settings are True, False, or Use Default.

ParameterName
The assigned Parameter name to be used to store the Parameter value from the specified column.

Background Color
Background color to be displayed on the selected column. Choose color from drop down options.

IsGroupable
This determines whether the column can be used in the Group panel on the grid. Settings are True or False.

IsFilterable
This setting, located at the top of the column, turns the filter button on and off. Settings are True or False.

ShowDistinctFilters
This setting turns the filter option on and off, which enables users to click check boxes to select filtered Members. Settings are True or False.

ShowFieldFilters
This enables and disables advanced filtering on the specified column. Settings are True or False.

IsSortable

This enables and disables sorting on the specified column. Settings are True or False.

IsMultilineText

This setting allows columns to display data on mulrows if the column is not wide enough to display the full value. Settings are True or False.

> **TIP:** The data will wrap on string spacing. If there are no blank spaces, the column data will not wrap. Additionally, keyed fields cannot be wrapped.

DataFormatString

Specify a number and date format to the data in the column. For example, mm/dd/yyyy will return the current Month/Day/Year using a slash. MM-dd-yyyy will return the Month-Day-Year using a dash. N0 will return a number without a decimal point, and #,###,0 will return a number without a decimal and a comma depicting the thousandth place. See Application Properties in Application Tools for more examples of number formats.

Width

Specifies the default column width to be displayed.

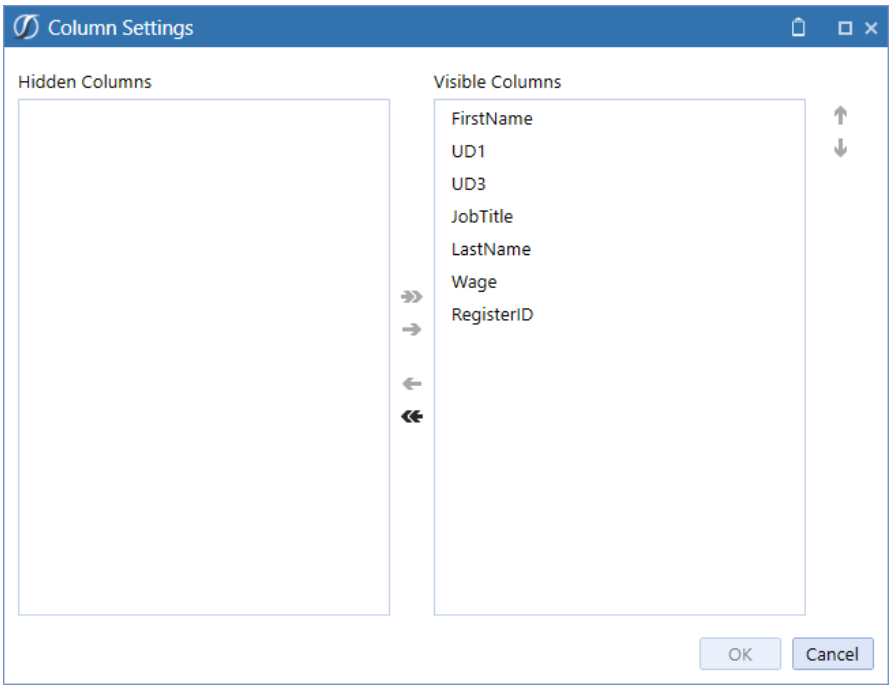
Drag and Drop

You can rearrange the columns, drag and drop the headers, and sort the columns.

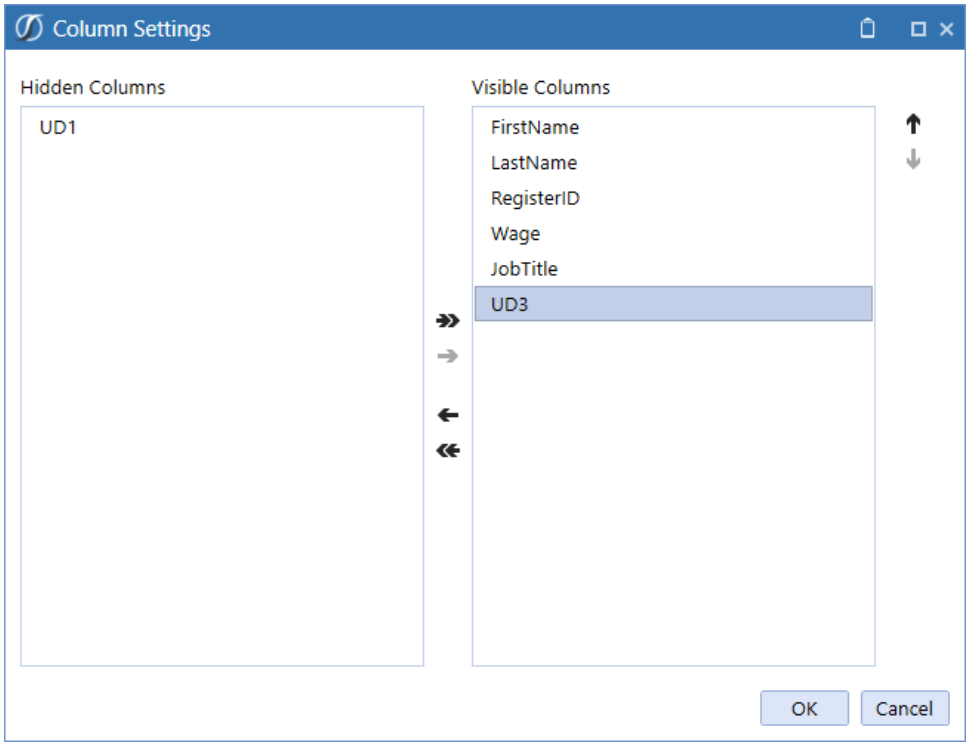
If Save State is True, changes to headers and columns ware saved.

Grid View	
Table Name	
Show Header	True
Show Column Settings Button	True
Allow Column Reorder	False
Show Toggle Size Button	True
Show Group Panel	True
Show Row Headers	True
Show Column Headers	True
Show Horizontal Grid Lines	True
Show Vertical Grid Lines	True
Default For Columns Are Visible	True
Retain Table Column Order	True
Column Name For Bound Parameter	
Save State	True

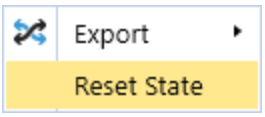
If Save State is False and you view the dashboard, you can't drag and drop headers. Click Column Settings to open the Column Settings window.



You can hide columns and change the order that the columns are shown on the dashboard.



Reset State

When you view a dashboard, you can right click Reset State 

to go back to the default settings of the grid view component.

Sort Column

When you click on a column header it will sort the column:

- first click, ascending order,
- second click, descending order,
- third click, no sorting order.

If Save State is set to True, the changes to the sort order will be saved.

Filter

When a new filter is applied to the grid, the filter icon is orange to show that it has been changed.

UD1	UD3	RegisterID	Wage	JobTitle	LastName	FirstName
Sales	Texas	E001022	67500.000000000	Administrative Assistant	Cooper	Jason
Advert	Texas	E001022	67500.000000000	Administrative Assistant	Cooper	Jason
Mkt	Texas	E001022	67500.000000000	Administrative Assistant	Cooper	Jason

List Box

This component displays a vertical list of predefined strings that are contained in a box and can be configured to allow single or multiple selections.

See Formatting and Action.

Map

The Map Component is used to display specific locations on a geographical map via a Dashboard DataSet Business Rule. Within each location, users can pass in Parameters in order to perform specific actions and display additional data. Map components can only display data via a Dashboard DataSet Business Rule using the XFMapItemCollection objects. Define the data set in the Business Rule, assign the rule to a Data Adapter, and the Data Adapter to the Map Component. See Map Dashboard Component Business Rule Examples in Implementing Security for examples on using this component.

Action

See Button.

Show Toggle Size Button

This controls whether the toggle icon displays on the dashboard at runtime.

Display Type

Humanitarian
This displays all countries/states/cities in English. The geographical location labels display smaller than Standard and Transport.

Standard
This displays all countries/states/cities in the respective local language. The map displays geographical location labels and road lines.

Transport
This displays all countries/states/cities in the respective local language. The map displays major cities with large labels.

Zoom Level
Enter the zoom value to which the map will open upon running the dashboard.

Center Latitude/Longitude
Enter the GPS coordinates to use upon launching the dashboard.

> **NOTE:** To define the exact GPS coordinates refer to a GPS coordinate website.

Member Tree

This component displays dimensional hierarchies represented in a tree view. The dimensional hierarchies are passed to the component by adding a Bound Parameter and are filterable based on the component's configuration using a Member Filter.

See Formatting and Action.

> **NOTE:** When adding a Bound Parameter, the only valid Parameter Type for the Member Tree is Member Dialog.

Radio Button Group

This displays a choice of mutually exclusive related options where users can only choose one. This component can be used with dashboards.

Formatting

See Button.

Action

See Button.

Supplied Parameter

This is a control that holds a Parameter value for use on a Dashboard, but it will not be displayed. This component can be used with dashboards.

Bound Parameter

See Button.

Text Box

This provides a box for entering text.

Formatting

See Button.

Action

See Button.

Allow Rich Text

Set this to True to enable special formatting options such as font, color, alignment, and more. Set this to False for a plain text box without formatting options.

Enable Spell Check

Set this to true to display spell check options in the toolbar when Allow Rich Text is set to True. Set the Text Box's Spell Check toolbar button to Not Used to not display red lines underneath misspelled words or the Spell Check right-click options. Enable Spell Check only works when English is selected in Windows for International users.

Multiline

Set this to True for word wrap or to click Enter and go to the next line. Set this to False for one string of text.

Tree View

This provides a graphical illustration of hierarchical data. This is used for relational or hierarchical data (non-Cube data).

Action

See Button.

Content Components

This section contains information for the following content components:

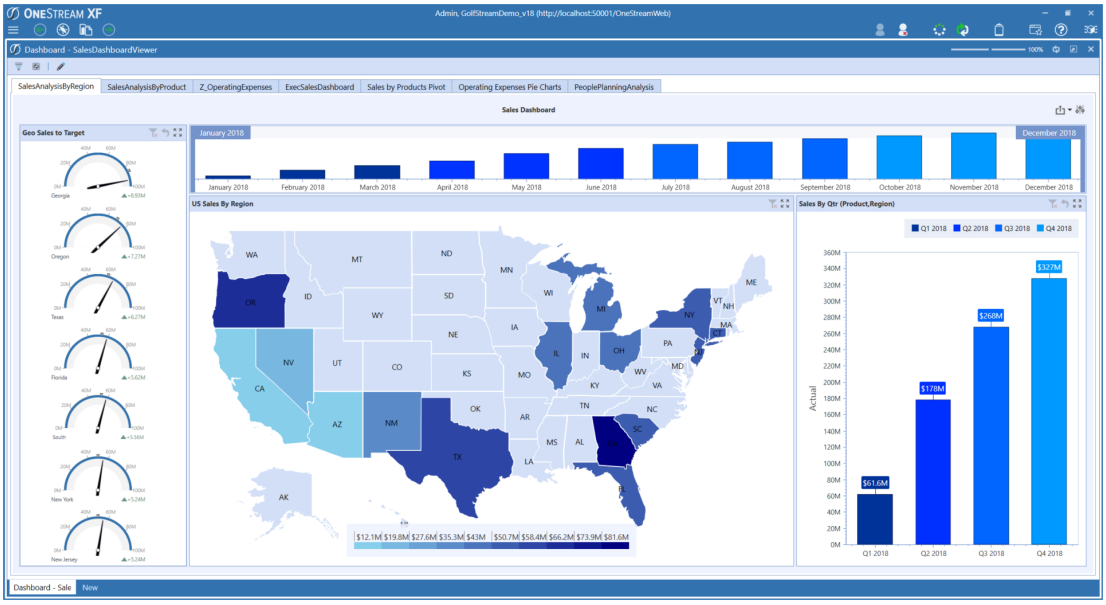
- BI Viewer
- Show or Hide Borders
- Charts, both Basic and Advanced
- Data Explorer
- Dynamic Grid
- File Viewer
- Image
- Label

- Password Box
- Pivot Grid
- Sankey Diagram
- Dashboard Filter Editor
- Text Editor
- Text Viewer
- Date Selector
- Report
- SQL Table Editor
- Embedded Components
- Text Box
- Menu Component

BI Viewer

This is an Interactive Dashboard Engine that is used to quickly create a Business Intelligence (BI) visualization of data from existing or new Data Adapters. The user can quickly create BI Dashboards that will allow the user to visualize and analyze data.

The BI Dashboards are a powerful tool and allow the user to dynamically add data and build Charts, Gauges, Maps, Grids, Cards. Etc. This BI component integrates well with the cube view MD Data adapter, allows the user at design time to; Add Calculated fields to the Data Source to create new fields and measures, design a path for Drill Down, Multi-Select filtering and Conditional formatting for dynamic viewing at runtime and provides the ability Customize Palette Colors.



Key Elements of the BI Viewer

- Compatible with existing dashboards
- Simple to build, share, and reuse components
- Fast drag and drop ad hoc data exploration
- Easily combine OneStream and non-OneStream data
- Bring trusted, validated OneStream data to the masse

Key Items for a Quick Start

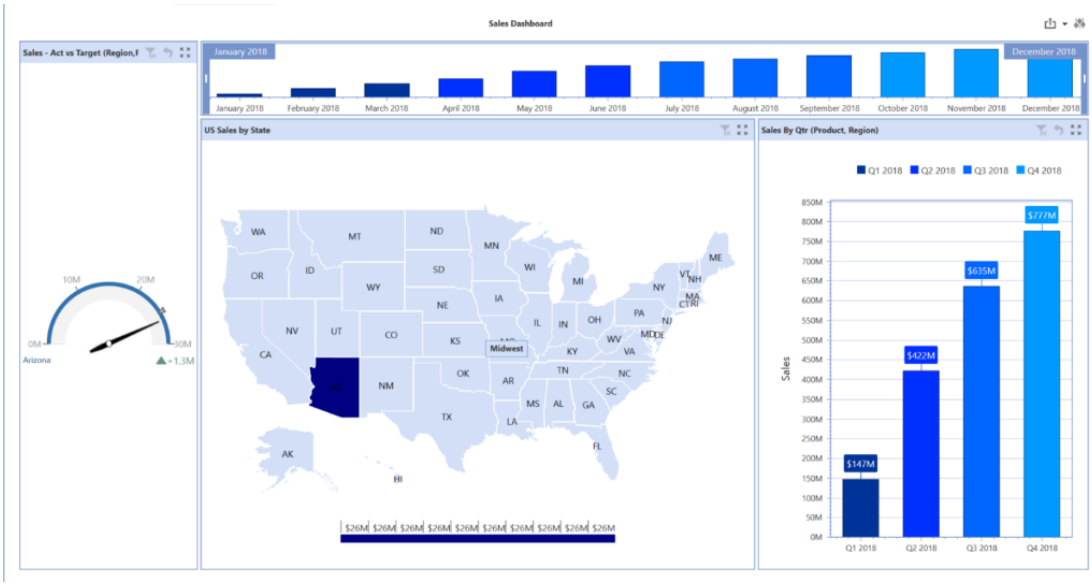
- Data Adapter – used to source the data and render the dataset
- BI Viewer Component – to design the dashboard
- Dashboard Component – to display the BI Dashboard
- Literal Parameter (optional) – to store the color palette

Show or Hide Borders

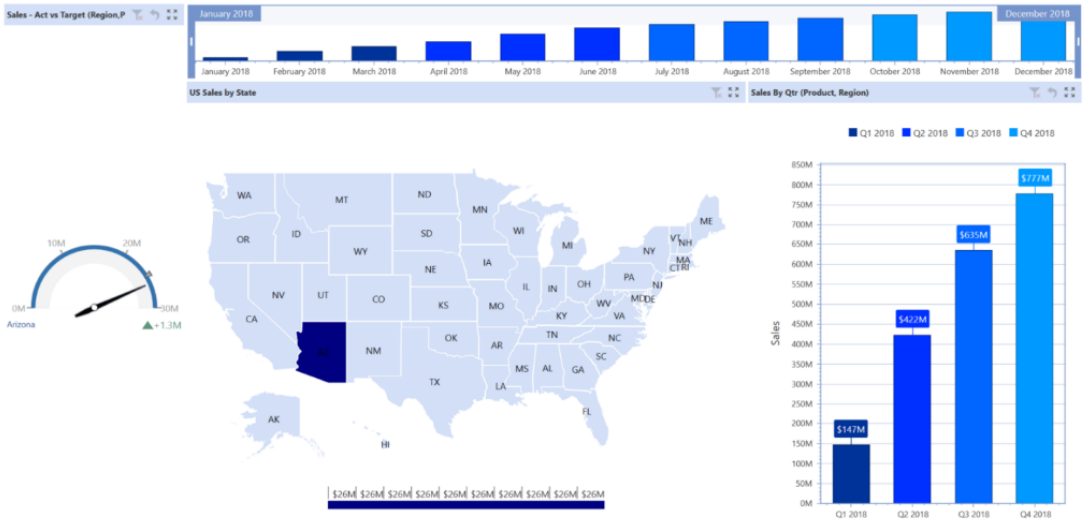
To use borders to separate areas on dashboards, select **Component Properties** > **BI Viewer** and set **Show Borders** to **True**.

BI Viewer	
Show Toggle Size Button	True
Action Name Value Pairs (e.g., Grid 1=Col1, ...)	
Parameter Name Value Pairs (e.g., Param1=Value1,	
Palette Colors	|!param_MapChartColors!|
Show Borders	True

For example:



If Show Borders is False, borders are not displayed. For example:



Additional information on the use and design elements of BI Viewer can be found in the BI Viewer Design and Reference Guide.

Book Viewer

This is used to display a Report Book in a Dashboard. This Component is not compatible with Mobile Dashboards.

File Source Type

Dashboard File

Display a Book stored in a Dashboard Maintenance Unit File section.

Application/System Database File

Display a Book stored in the Application or System Database Share.

File Share File

Display a Book from the File Share.

Full File Name

The name of the Book. Click the ellipsis and browse to the desired Book's file name.

Show Header

Select True in order to display a header derived from the Component's Name or Description.

Show Toggle Size Button

Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Initial Content Item

Enter a number to navigate to a specific page in the Report Book when the Dashboard runs.

Chart (Basic)

This is used to display a chart in a Dashboard. This Component can be used with Mobile and Application Dashboards.

> **NOTE:** OneStream has intentions to deprecate the Basic Chart Dashboard Component targeted for 2026 R3. We recommend all customers analyze their applications to determine if any dashboards are utilizing the Basic Chart. Any affected Dashboards should make use of the more versatile, and browser-compatible, Advanced Chart, or BI Viewer Components.

Chart Type

Area

An area chart or area graph displays graphically quantitative data and it is based on the line chart. The area between axis and line are commonly emphasized with colors, textures and hatchings. Commonly, two or more quantities are compared with an area chart. Area charts are used to represent cumulated totals using numbers or percentages over time. Use the area chart to show trends over time among related attributes.

Bar

A bar graph is a chart that uses vertical bars to show comparisons among categories. One axis of the chart shows the specific categories being compared, and the other axis represents a discrete value.

Bubble

A bubble chart is a variation of a scatter chart in which the data points are replaced with bubbles, and an additional Dimension of the data is represented in the size of the bubbles.

CandleStick

A candlestick chart is a combination of a line and bar chart, in that each bar represents the range of price movement over a given time interval.

Doughnut

A doughnut chart is functionally identical to a pie chart, except for a blank center and the ability to support multiple statistics as one.

Horizontal Bar

A horizontal bar graph is a chart that uses horizontal bars to show comparisons among categories. One axis of the chart shows the specific categories being compared, and the other axis represents a discrete value.

Horizontal Stacked Bar

A horizontal stacked bar chart stacks multiple data points in each bar on the chart instead of a single data point.

Line

A line chart displays information as a series of data points connected by straight line segments.

Pie

A pie chart is a circular chart divided into sectors, illustrating numerical proportion

Range

The range chart displays a range of data by plotting two Y values per data point, with each Y value being drawn as a line chart. The range between the Y values can then be filled with color.

Scatter

A scatter chart displays numerical values along the horizontal and the vertical axis, combining these values into single data points that are displayed in uneven intervals.

Spline

A spline chart is a specialized form of a line chart. Unlike conventional charts which connect data points with straight lines, a spline chart draws a fitted curve through the data points.

Spline Area

A spline area chart is a specialized form of an area chart. Unlike conventional charts which connect data points with straight lines, a spline chart draws a fitted curve through the data points.

Spline Range

The spline range chart displays a range of data by plotting two Y values per data point, with each Y value drawn as a line chart. The range between the Y values can be filled with color.

Stacked Area

The stacked area chart stacks two or more data series on top of one another.

Stacked Area 100

The stacked area 100 chart displays multiple series of data as stacked areas. The cumulative proportion of each stacked element is always 100% of the Y axis.

Stacked Bar

A stacked bar chart stacks multiple data points in each bar on the chart instead of a single data point.

Stacked Bar 100

A stacked bar 100 chart displays multiple data series as stacked bars where the cumulative proportion of each stacked element always totals 100%.

Stacked Line

A stacked line chart has lines that do not intersect because they are cumulative at each point.

Stacked Spline

A stacked spline chart has lines that do not intersect because they are cumulative at each point.

Stacked Spline Area

A stacked spline area chart is very similar to a spline chart. Data is displayed using different colors in the "area" below the line. Each series of points is represented with a different color.

Stacked Spline Area 100

The stacked spline area 100 chart is a variation of the spline area chart. The areas are stacked so that each series adjoins, but does not overlap the preceding series. This chart displays contributions for each data point to the category as a percentage that totals to 100%.

Stick

A stick chart is a combination of a line and bar chart.

Waterfall

A Waterfall chart is a visualization of the sequence of positive and negative values that arrive at a final value.

Show Toggle Size Button

Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Show Legend

If set to True, a legend containing a list of the variables will appear in the chart and an example of each appearance. This information allows the data from each variable to be identified in the chart.

Legend Title

The text to be displayed under the legend.

Legend Position

The location of the legend in relation to the chart. The options are Bottom, Left, Top, Right.

Chart X/Y Axis

Title
The title for the horizontal axis of the chart.

Label Rotation Angle
The angle to display the X Axis & Y Axis labels.

Use Automatic Range
If set to True, it automatically determines the range of values to display on the X & Y Axis based on the values being shown in the graph, if set to False, it needs to be set manually in the following fields.

Minimum Value
Manually set the X & Y Axis starting value. This can only be set if Use Automatic Range is False.

Maximum Value
Manually set the X & Y Axis maximum value. This can only be set if Use Automatic Range is False.

Step
Manually set the change in values displayed on the X & Y Axis. This can only be set if Use Automatic Range is False.

Chart Data

Data Series Source Type
The Data Source for the chart.

Cube View
A Cube View from the chart's Data Adapter is being used as the data source.

One Column Multiple Rows
One column and multiple rows from the chart's Data Adapter are being used for the data source.

One Row Multiple Columns

One row and multiple columns from the chart's Data Adapter are being used for the data source.

Suppress Zeroes

If set to True, this removes all results that are zero.

Row List Type

This specifies the rows to include from the chart's Data Adapter.

All Rows

The chart will include all rows.

First Row

The chart will only include the first row.

Row Index List

The chart will include the list of rows specified in the Row Index List field.

Row Index List

Enter the list of rows to include in the chart.

Column List Type for X Axis

This specifies the columns from the Data Adapter to include in the chart's X axis. When the Data Series Type is set to Cube View, all columns are included.

Column List

A specified list of columns used as X Axis types for the chart. These columns are entered in the Column List for X-Axis field.

All Columns

All columns are used as X-Axis Members.

All Numeric Columns

All columns with numeric values are used as X-Axis Members.

Column List for X Axis

The column names used as X-Axis Members. This field is only used when Column List is selected as the Column List Type for X Axis.

Use Column Headers for X Axis

If set to True, the column names are used as X-Axis Members, if set to False, the values/text in the named column are used as X-Axis Members.

Column List Type for Data

This specifies the columns from the Data Adapter to include in the chart. When the Data Series Type is set to Cube View, all columns are included.

Column List

A specified list of columns used as Data Sources for the chart. These columns are entered in the Column List for Data field.

All Columns

All columns are used as Data Sources.

All Numeric Columns

All columns with numeric values are used as Data Sources.

Column List for Data

The column name containing the data to display in the chart. This field is only used when the Column List Type field for Data is set to Column List.

Cube View Data Point Legend Type

Default

The legend text comes from the row headers.

Annotation

The legend text comes from the Annotation View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Assumptions

The legend text comes from the Assumptions View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Audit Comment

The legend text comes from the AuditComment View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Footnote

The legend text comes from the Footnote View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Variance Explanation

The legend text comes from the VarianceExplanation View Member in the Cube View instead of the default. The Data Adapter must include this information for this option.

Column for Legend Label

The legend text comes from the column name specified here.

Legend Labels List

Comma separated strings used in the legend to identify the data series.

Chart (Advanced)

This Component can be used with Mobile and Application Dashboards to display charts.

Series Type

Waterfall

A waterfall chart (also known as a Walk or Bridge chart) is a type of chart that provides a visual story of the net changes of values between two identified points, such as starting and ending values in income statements, balance sheets, and operational expenses. You can build a waterfall chart with data supplied in a particular format from a Data Adapter. You can also convert existing BarRangeSideBySideWaterfall charts to the new waterfall chart type.

Waterfall charts display net changes of a single value per bar. They provide positive variance values above the bar and negative variance values below the bar, as shown in the following image.

When you select the Waterfall type chart, the Waterfall Series Properties section provides configurations specific to the Waterfall chart:

- **Start Bar Color**: Sets the color of the first bar in the series.
- **Total Bar Color**: Sets the color of the bar that represents a Total.
- **Rising Bar Color**: Sets the color of all bars whose values are positive (but are not Start or Totals).
- **Failing Bar Color**: Sets the color of all bars whose values are negative (but are not Start or Totals).
- **Total Included in Series**: Indicates if the data source includes a total value (a sum). If set to TRUE, the chart identifies the last value in the data source as a Total bar on the chart. If set to FALSE, the chart auto-calculates a sum and adds the result as a Total bar on the chart.

- **Total Value Label**: Sets the label of the Total bar.
- **Include Subtotals**: Indicates auto-calculated subtotals added to the waterfall chart. If set to TRUE, you can indicate where you would like to place the subtotal bar(s). If FALSE, no subtotals are defined. Default is False.
- **Subtotal Indexes**: Enter a comma separated list of number(s) to indicate where you would like your subtotal bar(s) to be placed. For example, to display a subtotal after your first delta value, you would enter 1. You can also enter 2,4 for example, which would display a subtotal after the second delta value and also after the fourth delta value. These numbers count the delta values.
- **Subtotal Labels**: Enter a comma separated list of labels for each subtotal. Each label needs to have unique text. The number of labels must equal the number of subtotal indexes specified. If nothing is listed, defaults to “Subtotal 1,” “Subtotal 2,” “Subtotal 3,” and so forth.
- **Subtotal Bar Color**: Sets the color of bar(s) that represent a subtotal. Defaults to XFDarkBlueBackground.
- **Label Position**: Sets the position of the bar labels. Default is Auto. Options include:
    - **Auto**: Labels are placed above the bar for rising values and below the bar for falling values.
    - **Center**: Labels are placed in the center of the bars.
    - **InsideEnd**: Labels are placed on the inside bottom of the bar for falling values and on the inside top of the bar for rising values.
    - **InsideStart**: Labels are placed on the inside bottom of the bar for rising values and on the inside top of the bar for falling values.

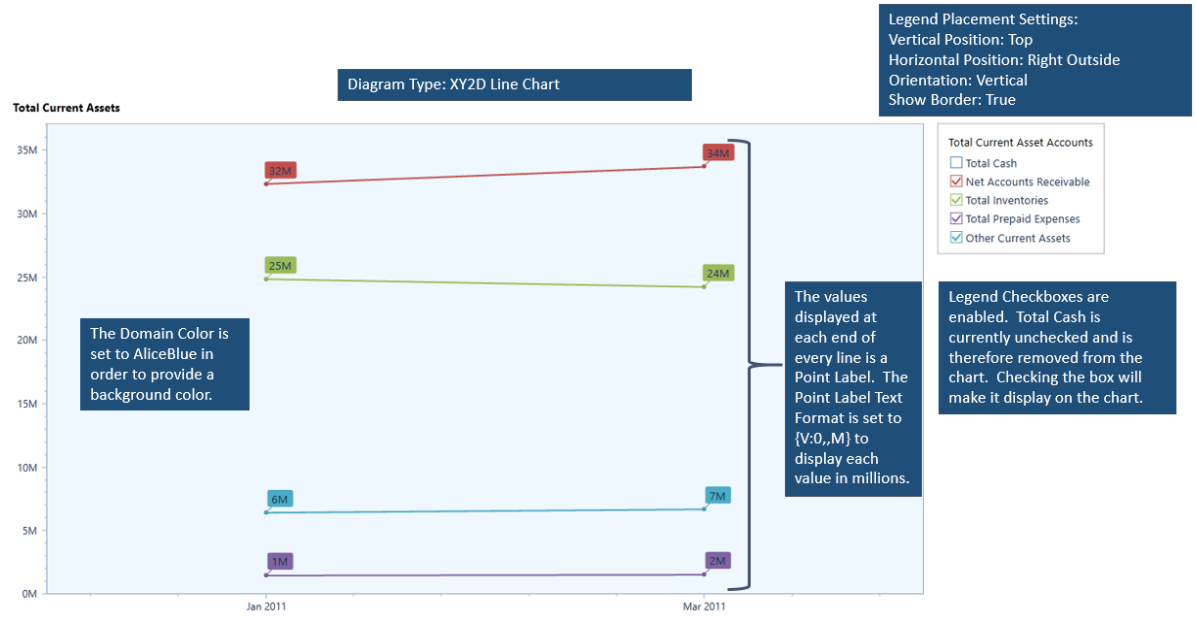
Create a Waterfall Chart

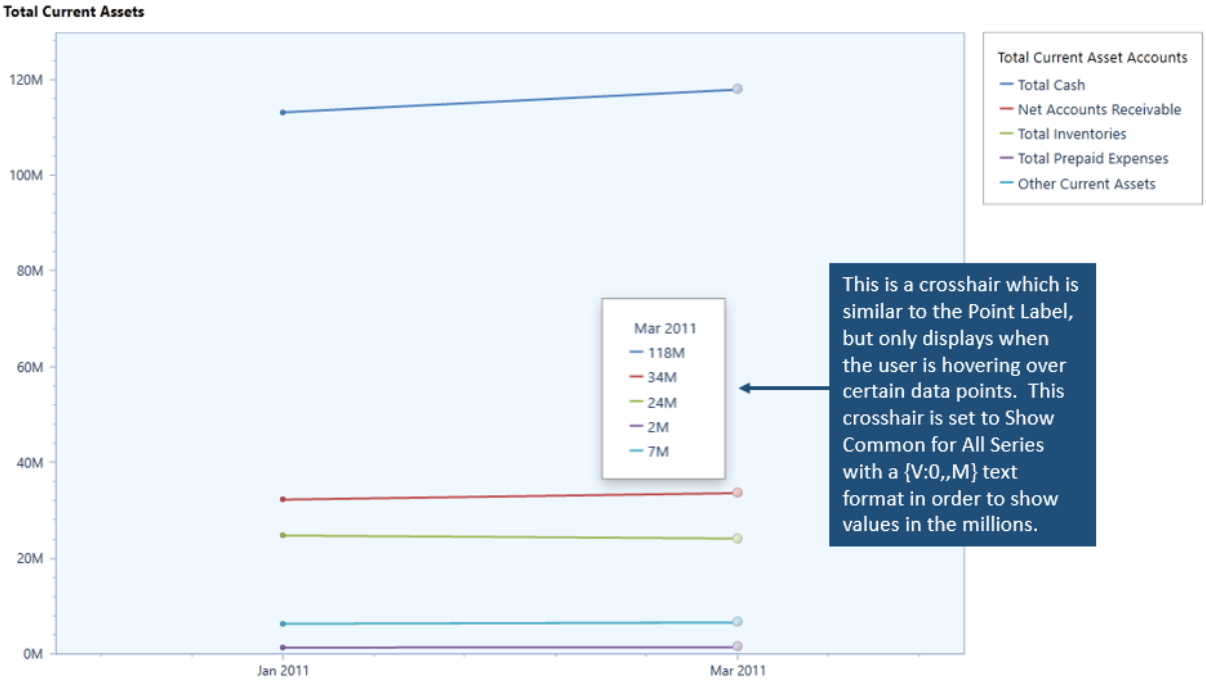
1. From the My Company Name, LLC Application tab, click **Dashboards**.
2. Expand **Dashboard Maintenance Units** and then expand the appropriate maintenance unit.
3. Click the **Components** label and then, from the toolbar, click **Create Dashboard Component**.
4. In the **Create Dashboard Component** dialog, click **Chart (Advanced)** and then click **OK**.
5. In the Name field, type a name for the new waterfall chart.
6. Scroll down to the Series Properties section and select **Waterfall** in the Type field.
7. Click **Save**.

If you have an existing BarRangeSideBySideWaterfall chart type, you can change it to waterfall by selecting **Waterfall** in the Type field.

Advanced Chart Examples

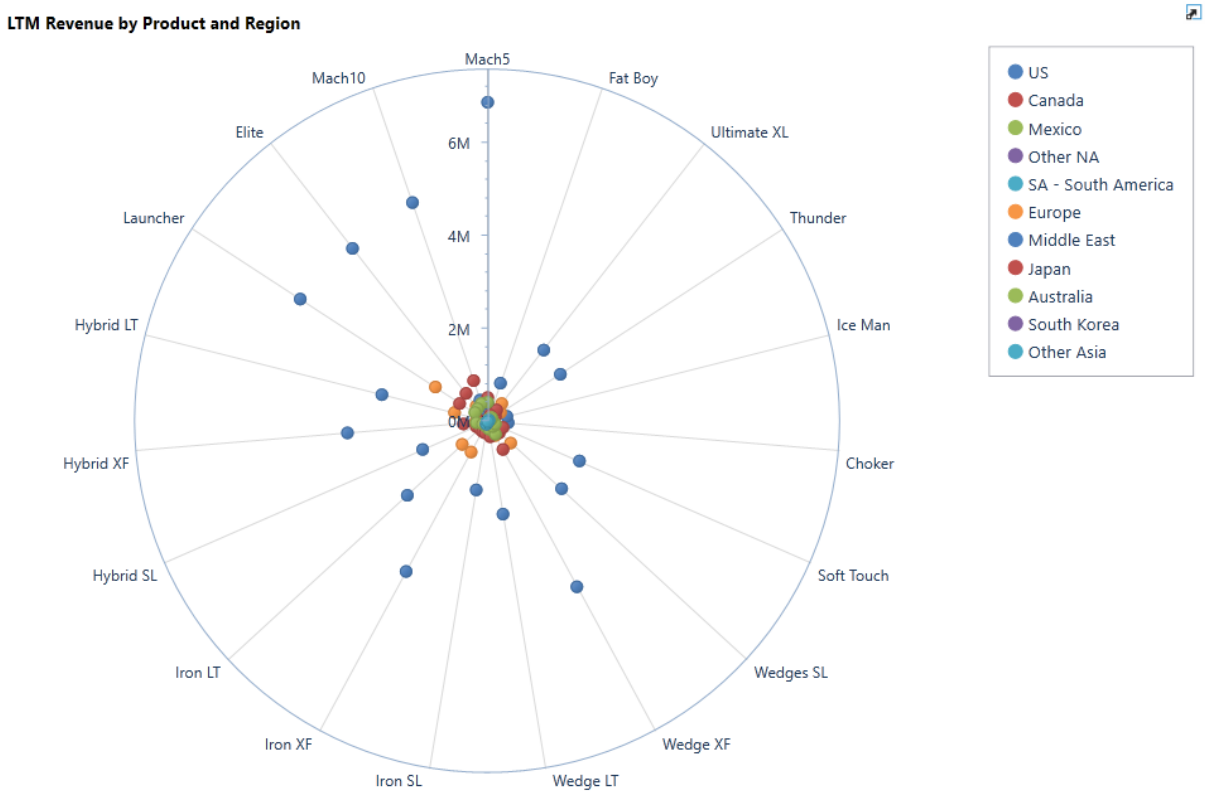
XY2D Line Chart Examples





Radar2D Radar Point Example

The chart below is displaying product revenue by region. This data is driven by a Cube View where the Cube View Rows display the region and the Cube View Columns display the products. Only the Y axis can be customized with this chart type which is driven by the Cube View data. The Legend is driven by the Members on the Cube View Rows and the column Members display multiple axes organized in a circle.



Data Explorer

This component is linked to a cube view and displays in the standard Data Explorer view which is seen when a cube view is launched from the Analysis area on the OnePlace tab. To use this component, specify a name and add a Data Adapter linked to a cube view. You cannot use this component with mobile dashboards.

Data Explorer Report

This is a component linked to a cube view and displayed in Report Viewer as if generated on the fly from a Data Explorer window. To use this, name the component and add a Data Adapter linked to a cube view. The component can be used with mobile and application dashboards. See Report in the Cube Views section.

Dynamic Grid

The Dynamic Grid Component displays data in a tabular format from a Workspace Assembly-driven data source. Each column represents a field, and each row represents a record. This grid enables you to read data from various sources, transform that data, and write it back to any specified target. This offers a flexible and high-performance feature for managing large datasets. It supports paging, dynamic columns, custom formatting, and other customizations. This component maintains the core functionality of the SQL Table Editor Component while enhancing the flexibility of data presentation and end-user interaction.

The key features of the Dynamic Grid Component include the following benefits:

- **Flexible data transformation**: Read, write, and transform data from various sources using Workspace Assemblies.
- **High performance**: Manage large datasets efficiently with paging.
- **Dynamic columns**: Add, remove, and customize columns.
- **Custom formatting**: Apply custom formatting to data for better visualization.

This component is created in the Windows Application and can be displayed in the Modern Browser Experience and the Windows Application.

To use the Dynamic Grid Component, follow these steps:

1. Review the Prerequisites.
2. Create a Dynamic Grid Component.
3. Update Configuration Settings.
4. Add the Dynamic Grid Component to a Dashboard.
5. Create a Workspace Assembly Dynamic Grid Service.

Review the Prerequisites

Before you create a Dynamic Grid, review these developer and setting prerequisites.

Developer

- Primarily, the Dynamic Grid will be created by a developer. Users will interact with the grid.
- The Workspace Assembly required can be complex and is designed to be used by a developer.
- The developer should have in-depth knowledge of OneStream Workspace Assemblies, source data, and target data.

Settings

- Create a Workspace and Maintenance Unit. See Workspaces Setup.
- Create a Dashboard Group and Dashboard. See Dashboards.

Create a Dynamic Grid Component

1. Go to **Application** > **Presentation** > **Workspaces**.
2. Click the arrow to the left of the Workspace to expand the navigation tree.
3. Click the arrow to the left of the Maintenance Unit to expand the navigation tree.

   > **NOTE:** Do not select the Default Maintenance Unit if you have also selected the Default Workspace because you will not be able to add a Dashboard Component.

4. Click **Components**.
5. In the toolbar, click the **Create Dashboard Component** button.
6. In the **Create Dashboard Component** dialog box, select **Dynamic Grid** and then click the **OK** button.
7. Name the Dynamic Grid.

   > **NOTE:** The Dynamic Grid name must be unique within its Workspace.

8. Update the configuration settings. See Update Configuration Settings.
9. Click the **Save** button.

Update Configuration Settings

Configure the available settings for the Dynamic Grid Component.

Property	Description
General (Component)	
Name	Enter a name for the Dynamic Grid Component. The name must be unique within its Workspace.

Property	Description
Description	Enter a description for the Dynamic Grid Component. The description will display as the title text in the title header row.
Formatting	
Display Format	Click the ellipsis button to select an option for how the colors display in the background. • BackgroundColor: Select this option to apply one background color to all rows in the grid. • AlternateRowBackgroundColor: Select this option to apply a specified color to every other row in the grid, creating a striped effect. This option can improve readability by visually distinguishing between adjacent rows. Display Format settings in the Assembly code will override the option selected in this property.
Processing	
Template Parameter Values	Enter a comma-separated list of name-value pairs to specify Template Parameter Values. These values are used as the default settings for Template Parameters referenced using the ~!tParam!~ syntax related to Dynamic Workspace processing (for example, “tParam1=Value1, tParam2=Value2”).

Property	Description
Text 1	Enter arbitrary text that can be accessed by Business Rules or Dynamic Workspace processing. For example, this property can be used to specify a parameter name or a functional name as a hint when implementing Selection Changed actions. See User Interface Action and Navigation Action in Action. It also supports template variables (for example, ~!tParam!~) for Dynamic Workspace processing.
Text 2	Enter arbitrary text that can be accessed by Business Rules or Dynamic Workspace processing. For example, this property can be used to specify a parameter name or a functional name as a hint when implementing Selection Changed actions. See User Interface Action and Navigation Action in Action. It also supports template variables (for example, ~!tParam1!~) for Dynamic Workspace processing.
Action	
See Action.	
Dynamic Grid	

Property	Description
Filter Mode	Configure to filter data from column headings using options Popup and Filter Row. • Popup: Select to filter by clicking the funnel button in the column heading, which opens a popup window. This is the default. • Filter Row: Select to filter directly in the column heading. This setting is the same as the Filter Mode setting for the SQL Table Editor Component. See SQL Table Editor Filtering.
Default For Columns Are Visible	Select True to display all columns in the grid (unless overridden at the individual column level). Select False to hide all columns in the grid (unless overridden at the individual column level).
Show Title Header	Configure to show (True) or hide (False) the title header row at the top of the grid. **NOTE:** If set to False, the Default For Allow Column Updates, Process Selection Changed For Inserts, Allow Inserts, and Allow Deletes options remain enabled.
Default For Allow Column Updates	Select True to allow the data in the existing table to be updated. Select False to turn off this option.

Property	Description
Show Title	Configure to show (True) or hide (False) the title text in the title header row.
Show Column Settings Button	Configure to show (True) or hide (False) the Column Settings button, which enables users to reorder the columns and set the column visibility.
Show Deselect All Button	Configure to show (True) or hide (False) the Deselect All button, which enables users to clear the checkboxes for all selected rows.
Process Selection Changed For Inserts	Select True, and set the Allow Inserts option to True to allow the Insert button to process the selection changed event, which is often configured to refresh a portion of the dashboard. Select False to turn off this option.
Allow Inserts	Select True to allow rows of data to be added to the table. Select False to turn off this option.
Allow Deletes	Select True to allow rows of data to be deleted from the table. Select False to turn off this option.
Retain Table Column Order	Select True to allow the column order to be changed using the Assembly service. Select False to turn off this option.

Property	Description
Read Only Text Color	Select (Use Default) to set the default text color to standard black text. Update to use any text color, including a system color (for example, XFReadOnlyText).
Column Name for Bound Parameter	Enter the name of the column to use to change the value of the bound parameter when a database row is selected. This is used when a Dynamic Grid is being used to affect the display of other Dashboard Components (for example, when showing detailed information for the selected row).
Default For Show Search	Select True to enable search functionality within the drop-down menu for all columns that are bound to a list parameter, unless specified otherwise for individual columns. This setting is the same as the Default For Show Search setting for the SQL Table Editor Component. See SQL Table Editor Searchable Drop-Down List. Select False to turn off this option.

Property	Description
Allow Multiselect	Select True to allow checkboxes to be selected for multiple rows. The active selected items will be passed to the defined Bound Parameter field as a comma-delimited list. The Bound Parameter format will be: Item1, Item2, Item3. If three values, A, B, C,D, are selected (where C,D is one value), the resulting Bound Parameter string is: A, B, "C,D". This setting is the same as the Allow Multiselect setting for the SQL Table Editor Component. See SQL Table Editor Multiselect. Select False to turn off this option and only pass a single Bound Parameter at a time. The checkboxes will not display in the user interface.
Rows Per Page	Enter a numeric value between 1 and 3000 to specify the number of rows to display before a page break. The default is -1, which turns off the property and uses your defined user preference property Grid Rows Per Page, which is located in the Security settings on the user profile. If you enter a value greater than 3000, the maximum number of results per page is still 3000.
Save State Settings	

Property	Description
Save State	Select True to retain the user settings on the component. User preferences for columns saved will be for Order, Visibility, Filtering, Freeze Bar, Sorting, and Widths. This setting is the same as the Save State setting for the SQL Table Editor Component. See SQL Table Editor Save State User Preferences. Select False to turn off this option. **NOTE:** To reset the Save State to default, right-click on the dashboard to enable the Reset State to return to its Component Properties settings.
Vary Save State By	Select True to save the Save State settings by Workflow Profile and Workflow Scenario. The related Dashboard Component will have the option Reset All States, which can be used to clear the Save States for a user across all the Vary Save State By parameters. Select False or (Use Default) to turn off this option.
Data Manipulation Buttons	
Show Data Manipulation Buttons	Configure to show (True) or hide (False) the Add Row, Remove Row, Cancel, and Save buttons.
Show Add Row Button	Configure to show (True) or hide (False) the Add Row button.

Property	Description
Show Remove Row Button	Configure to show (True) or hide (False) the Remove Row button.
Show Cancel Button	Configure to show (True) or hide (False) the Cancel button.
Show Save Button	Configure to show (True) or hide (False) the Save button.

Add the Dynamic Grid Component to a Dashboard

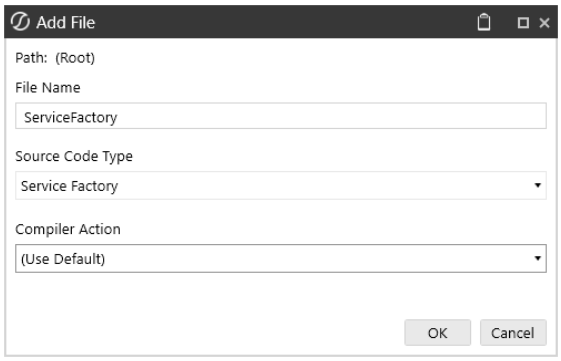
1. In the same Maintenance Unit as the Dynamic Grid Component, click the arrow to the left of **Dashboard Groups** to expand the navigation tree.
2. Select a dashboard.
3. Click the **Dashboard Components** tab.
4. In the toolbar, click the **Add Dashboard Component** button.
5. In the **Add Dashboard Component** dialog box, select the Dynamic Grid Component and then click the **OK** button.
6. Click the **Save** button.

Create a Workspace Assembly Dynamic Grid Service

1. In the same Maintenance Unit as the Dynamic Grid Component, click **Assemblies**.
2. In the toolbar, click the **Create Assembly** button.
3. On the **Assembly Properties** tab, in the **Name** field, name the Assembly.

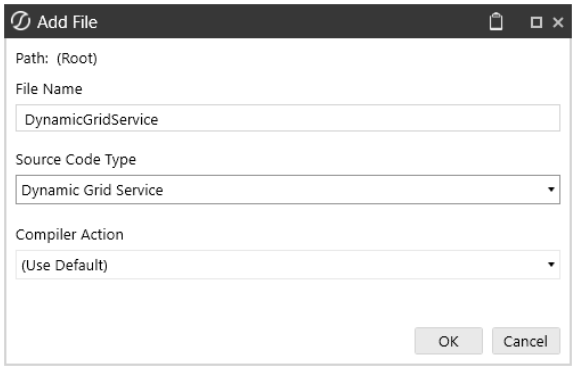
> **NOTE:** The Assembly name must be unique within its Workspace.

4. In the **Compiler Language** drop-down menu, select a language.
5. In the toolbar, click the **Save** button.
6. Create files for the Service Factory and the Dynamic Grid Service.
    a. On the **Assemblies Files** tab, right-click on **Files** and select **Add File**. In the **Add File** dialog box:
        i. In the **File Name** field, enter a name for the file (for example, ServiceFactory).
        ii. In the **Source Code Type** drop-down menu, select **Service Factory**.
        iii. In the **Compiler Action** drop-down menu, select **(Use Default)**.
        iv. Click the **OK** button.



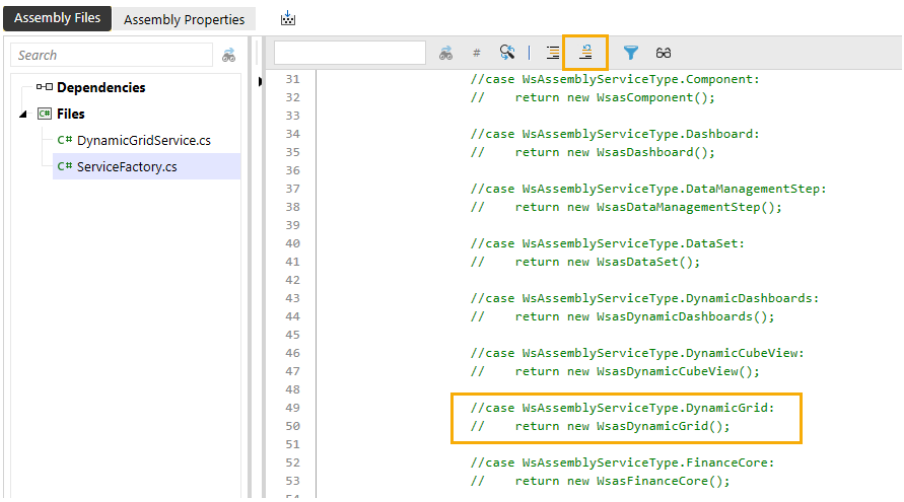
    b. On the **Assemblies Files** tab, right-click on **Files** and select **Add File**. In the **Add File** dialog box:
        i. In the **File Name** field, enter a name for the file (for example, DynamicGridService).
        ii. In the **Source Code Type** drop-down menu, select **Dynamic Grid Service**.
        iii. In the **Compiler Action** drop-down menu, select **(Use Default)**.

iv. Click the **OK** button.

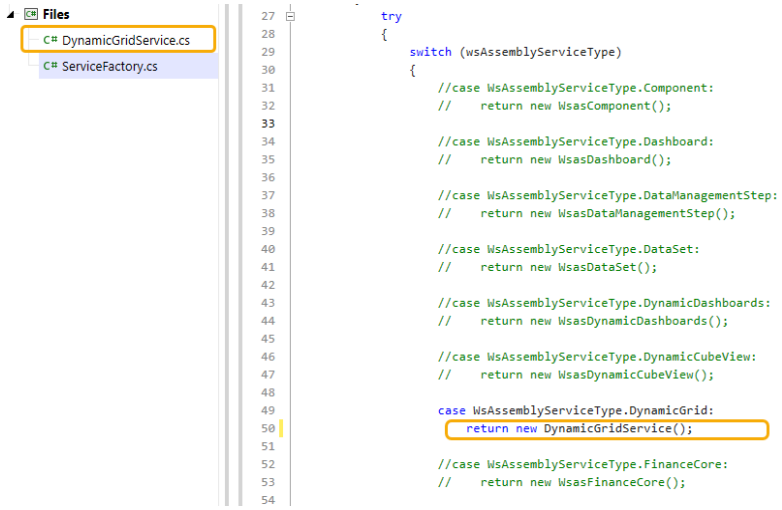


7. By default, Dynamic Grid methods for Get and Save are included. See Dynamic Grid Service. Modify the code as needed.

8. To enable the Dynamic Grid, in the **Service Factory** Assembly File, highlight the lines for the Dynamic Grid, and click the **Uncomment the selected lines.** button.



> **NOTE:** Ensure the object name matches the name of the Dynamic Grid Service file.



9. In the toolbar, click the **Save** button.

10. Select the Maintenance Unit. In the **Workspace Assembly Service** field, enter the name of the Assembly and the Service Factory Assembly File in this format: <Assemblyname>.<ServiceFactoryAssemblyFilename>

> **Example:** If the Assembly name is DynamicGridService and the Service Factory Assembly File name is ServiceFactory, the Workspace Assembly Service would be: DynamicGridService.ServiceFactory

⊟ General (Maintenance Unit)	
Name	SBMaintenanceUnit
Workspace	SBDynamicGrid
Description	
Is Mobile	False
⊟ Assemblies	
Workspace Assembly Service	DynamicGridService.ServiceFactory
⊟ Security	
Access Group	Everyone
Maintenance Group	Everyone

11. In the toolbar, click the **Save** button.
12. Select the Assembly, and click the **Compile Assembly to check syntax** button to ensure the Workspace Assembly files compile.

File Viewer

This is used to present files such as PDF's stored on the OneStream file share. This Component can be used with Mobile and Application Dashboards.

There are only three situations where users can see these files:

1. They must have the ManageFileShare system role
2. The files must be stored in a folder under the Incoming folder.

   ```
   Example:
   \\MyServerName\OneStreamShare\Applications\MyApplicationName\Incoming\MyDashboardFiles
   ```

3. The file is placed in one of the folders under \\MyServerName\OneStreamShare\Applications\MyApplicationName\Groups and then in a named folder with the same name as a User Group. The file must be in the folder with same name as the User Group and the user must have permissions to that Group.

File Source Type

URL
Display a file from an internal or external web page.

Dashboard File
Display a file stored in a Dashboard Maintenance Unit File section.

Application/System Database File
Display a file stored in the Application or System Database Share.

File Share File
Display a file from the File Share.

URL or Full File Name

The URL or name of the file being used. Enter the full URL name, or click the ellipsis and browse to the desired file.

Show Header

Select True in order to display a header derived from the Component's Name or Description.

Show Toggle Size Button

Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Show PDF Toolbar

Select True to display the PDF Toolbar while viewing the Component on a Dashboard, select False to hide it.

Auto Play (audio and video files)

If set to True, the video/audio files shown in the Dashboard Component will automatically start.

Process Extensible Document

If set to True, the File Viewer Component will run and process the attached Extensible Document, if set to False the unprocessed file will display which is mainly used for testing purposes.

> **NOTE:** An Extensible Document is a Text, Word, PowerPoint, or Excel file that uses Parameters in its content. The file name must contain .xfDoc before the extension.

```
Example: StatusReport.xfDoc.docx
```

See Extensible Document Framework in Presenting Data With Extensible Documents for more details on this feature.

Image

This is used to display an image on a Dashboard. This Component can be used Application Dashboards.

Formatting
See Label

Image

File Source Type
The Image (if any) to display.

URL
The image displayed is located on a URL.

Workspace File
The image displayed is located in a file stored in the Workspace Maintenance Unit File Section.

Application Database File
The image displayed is located in a file in the Application Database Share.

System Database File
The image displayed is located in a file in the System Database Share.

File Share File
The image displayed is located in the File Share.

URL or Full Name File
The URL, file path or file name for the displayed image.

URL Syntax
Access a built-in toolbar image, or use an image from any web server

Built-in OneStream Image Example
/OneStream.ClientImages;component/Misc/ToolbarImage.png

Web Server Example

http://www.onestreamsoftware.com/img/onestream-logo.png

Database File System Example

Enter the path and file name Documents/Public/TestImage.png

Workspace File Syntax

Enter the name of the file resource TestImage.png

The following Excel properties are used to specify an Excel file from either an Extensible Document or Report Book in order to display it as an Image or Button on a Dashboard.

Excel Page Number

Enter the page number of the Excel document to display.

Excel Sheet

Enter the name of the Excel worksheet to display.

Excel Named Range

Enter the named range from Excel to display. The named range must be the specified Excel sheet in the Excel Sheet property, or the Excel Sheet property must be blank in order to use a named range.

See Image under Component Display Format Properties to position and align the Excel image as needed.

Label

This is used to display text strings on a Dashboard. This Component can be used with Mobile and Application Dashboards.

Formatting

Text (use {1}, {2} for Data Table Cells) (Label Component Only)

The text to display in the Label. Use {1}, {2} to reference cells from the associated Data Adapter's Data Table.

Example: Sales and Profit for |!EntityParam!| are {1} and {2}

Tool Tip

The text to display when a user's mouse hovers over the Component.

Display Format

The formatting assigned to the control. See Component Display Format Properties

Data Table Cells from Adapter

Number of Data Table Cells

Specify the Data Table Cells available for display if the Label Component is being used, or to determine which State to display if the State Indicator Component is being used.

The following properties become available if a number is specified in the Data Table Cells field.

Data Row Selection Type

This specifies how the data cell row from the attached Data Adapter is identified and passed into the Label placeholder.

First Row

The specified row is the first row of data from the Data Adapter.

Row Index (0-based)

The row is specified by entering the row number in the Key Column Value or Row Index field. This selection uses 0-base row numbering where the first row is identified as row 0. (e.g., if there are six rows returned in the Data Adapter, enter 5 to get the last row)

Key Column Name and Value

The row is specified by querying the attached Data Adapter. Enter the column name to search in the Key Column Name field and enter the string/value to search in the Key Column Value or Row Index field. The row where the value is found will be the specified row.

Key Column Name

This property is only available when Key Column and Value is selected for the Data Row Selection Type. Enter the column name to search.

Key Column Value or Row Index

This property is only available when Row Index (0-based) or Key Column Name and Value is selected for the Data Row Selection Type. Enter the Key Column value/string to search or enter the Row Index Number.

Result Column Name

Enter the column name to return to the Label placeholder.

Number Format

The formatting applied to a value returned to the label placeholder if a string is not returned.

Logo

This is used to display a logo on a Dashboard. This Component can be used with Mobile and Application Dashboards.

Formatting

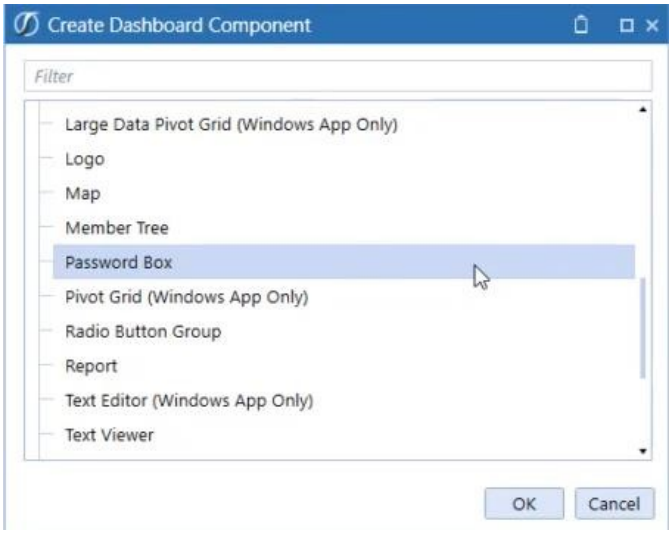
See Label

Password Box

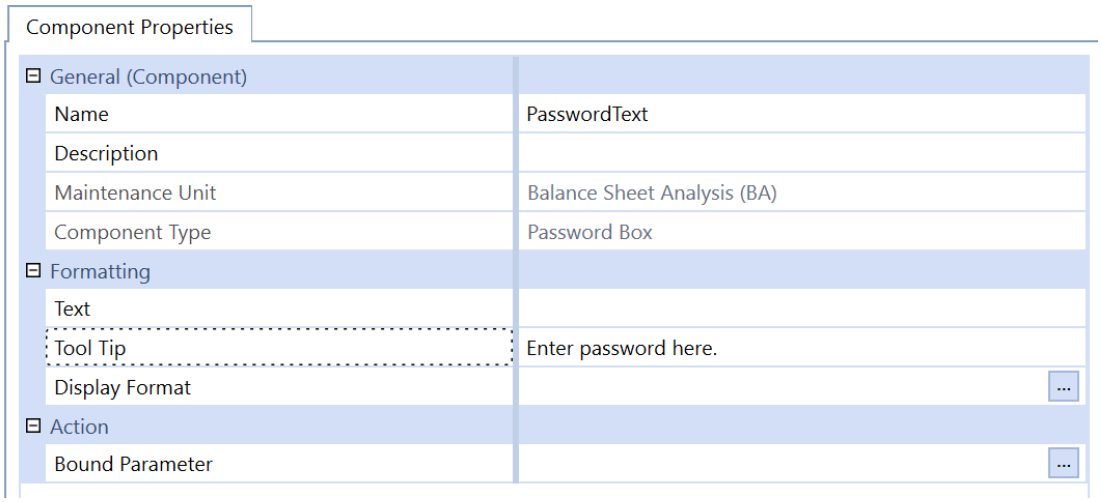
The Password Box displays a text box in which characters entered are hidden, so you can securely enter a password. When you create a Password Box, you can enter tool tip text.

Create a Password Box

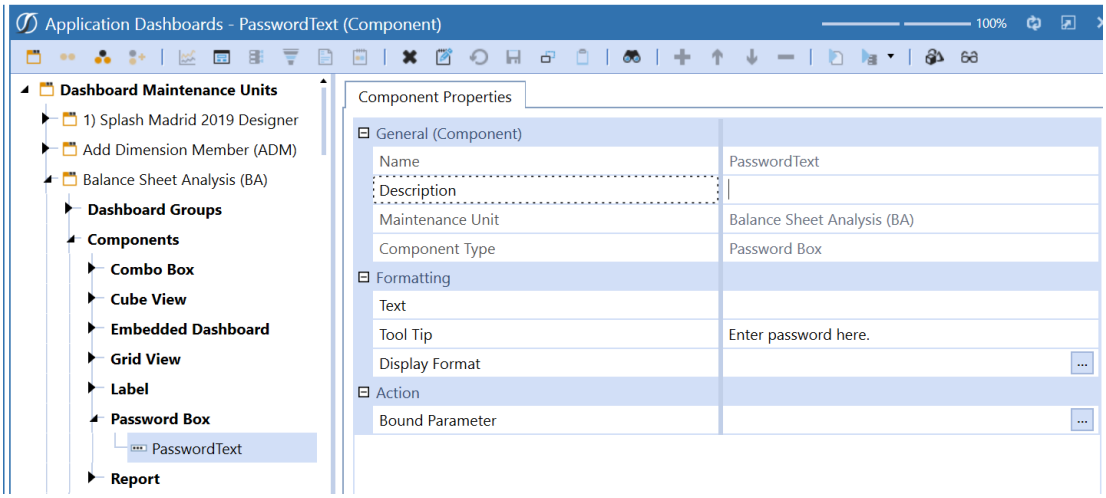
1. Go to **Presentation > Dashboards**.
2. Expand a **Dashboard Maintenance Unit** and then **Components**.
3. Click **Create Dashboard Component**. The Create Dashboard Component window opens.
4. Select **Password Box** and click **OK**.



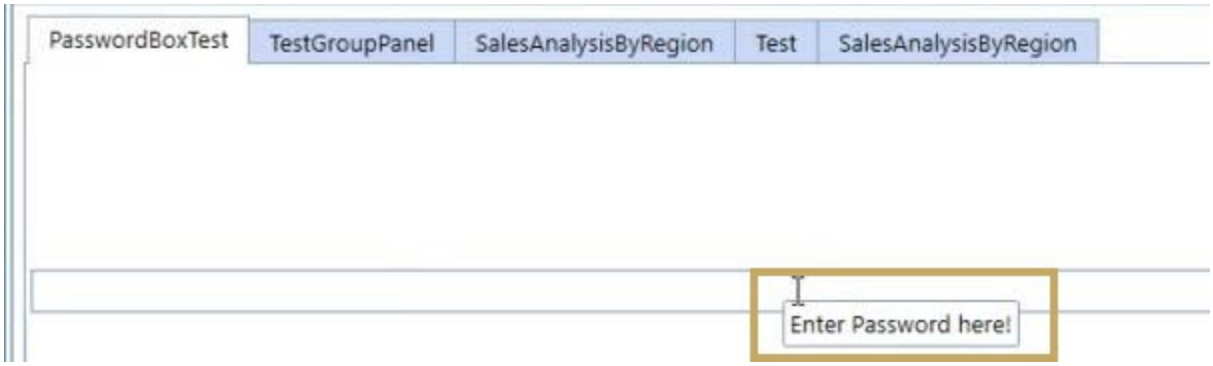
5. In **Component Properties**, enter a name. Optionally, enter text that displays as a tool tip.



6. Click **Save**. The password box component appears in the Components list.



8. Add the password box component to the dashboard and then run the dashboard.

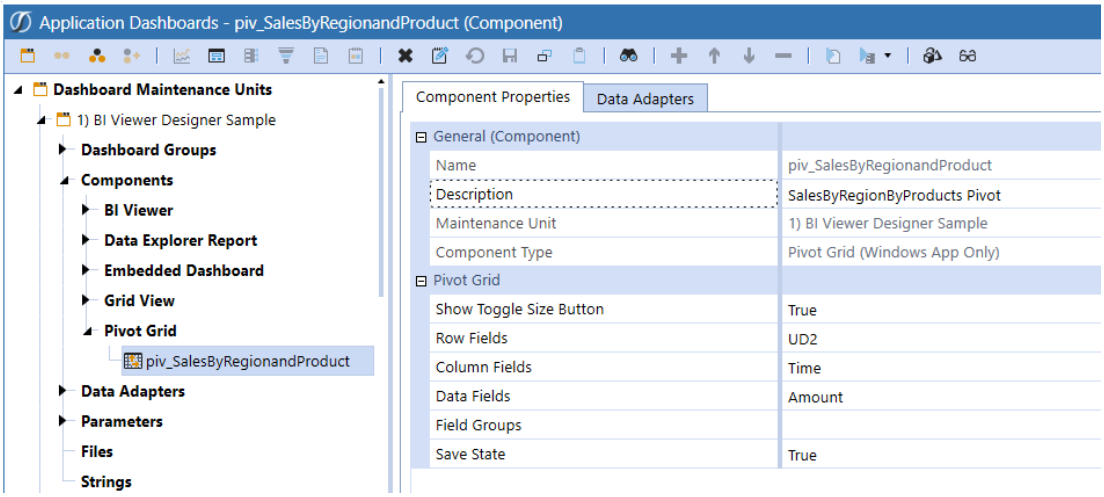


Pivot Grid

The Pivot Grid is a component within the Dashboard Maintenance Unit that allows you to create a pivot table, utilizing data from any existing and new Data Adapters to perform multi-dimensional analysis. At runtime, the user can customize the layout of the report using simple drag-and-drop operations and conditional formatting based on their analysis requirements. In the Pivot Grid, data can be summarized, calculated fields (text and decimal) can be added and displayed in a cross-tabular format that can be sorted, grouped and filtered. The Pivot Grid also supports drill-down (to view the underlying Data Adapter Results Table). The resulting output can be printed or exported to various file formats (such as PDF, XLS and XLSX).

All Data Adapters are supported when using the Pivot Grid. Utilizing Cube View MD (Multi-Dimensional) as the source of a Data Adapter will return the selected Cube View as a Multi-Dimensional Fact Table and provide an easy to use source table for the Pivot Grid. The results of the Cube View MD are Dimensions (Entity, Consolidation, Scenario, time, View, Account, Flow, Origin, IC, UD1-UD8) as columns.

Component Properties



Name

The name of the Pivot Grid.

Description

A quick description of the Pivot Grid.

Maintenance Unit

The Maintenance Unit to which the Component belongs.

Component Type

The type of Component

Pivot Grid

Show Toggle Size Button

Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Row Fields
Enter a comma separated list of Column Names (from the Results Table) to be placed in the Row Area of the Pivot Grid by default.

Column Fields
Enter a comma separated list of Column Names (from the Results Table) to be placed in the Column Area of the Pivot Grid by default.

Data Fields
Add a Measure (from the Results Table) to be assigned as the default Measure. Multiple measures can be added using a comma "," to separate the column names.

Field Groups
Enter a comma separated list of Column Names (from the Results Table) to be grouped together as a default.

Save State
Enables the Save Button on the Pivot Grid when set to True and Disables the Save button when set to False.

> **NOTE:** Layout button on Pivot Grid when copying Dashboards:

When copying a Dashboard that contains a Pivot Grid or Large Data Pivot Grid, the Pivot Grid and Large Data Pivot Grid component layouts are saved per dashboard location, not by component. Next time each dashboard is run the Pivot Grids should have different saved layouts.

Additional information on the use and design elements of the Pivot Grid can be found in the Pivot Grid.

Report

This Component can be used with Mobile and Application Dashboards.

Auto Fit to Page Width
If set to True, this will auto fit the Report to the area allocated in the Dashboard.

Text Editor

This is used to display documents created with the Text Editor feature in a Dashboard. It also can process Extensible Documents when embedded in a Dashboard. This Dashboard Component is only compatible with the OneStream Windows App. If using any version other than the OneStream Windows App to see files created with the Text Editor feature within a Dashboard, information will not display, and an error message appears telling the user that Text Editor documents are not available in the OneStream browser version. See Text Editor in Application Tools for more information.

Show Toggle Size Button
Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

File Source Type

Dashboard File
Display a file stored in a Dashboard Maintenance Unit File section.

Application/System Database File
Display a file stored in the Application or System Database Share.

File Share File
Display a file from the File Share.

Allow Create File
If set to True, this allows users to create text files.

Allow Open File
If set to True, this allows users to open text files.

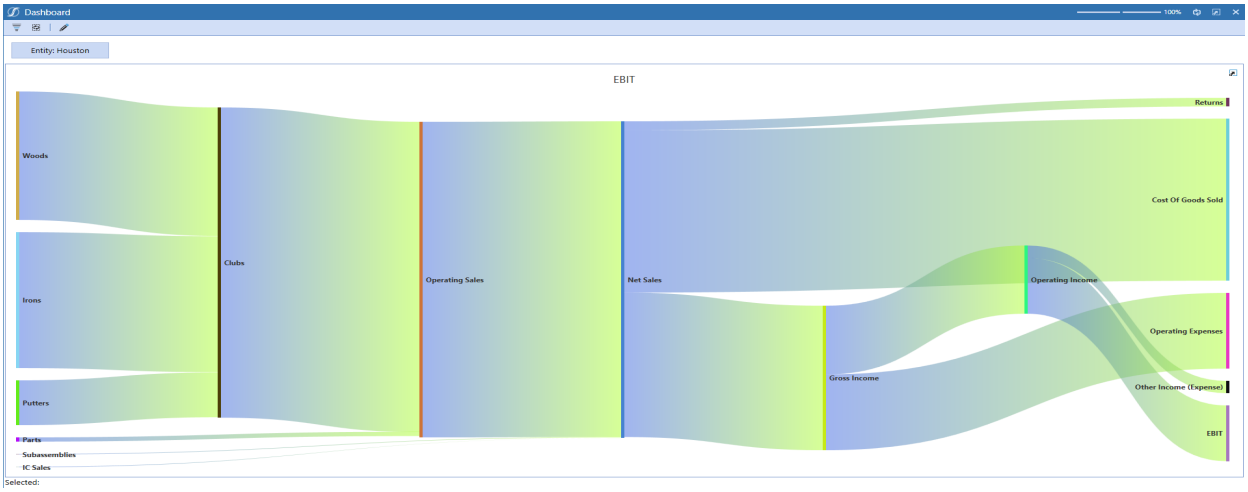
Allow Save File
If set to True, this allows users to save text files.

Show Ribbon

Select True to enable users to see the Ribbon below the Menu Bar, select False to hide the Ribbon.

Sankey Diagram

This component can be used with the Windows Application dashboards to display Sankey diagrams.



Sankey diagrams enhance the visualization of connected data sources and entities using the method command type business rules or single-row cubes views with multiple columns.

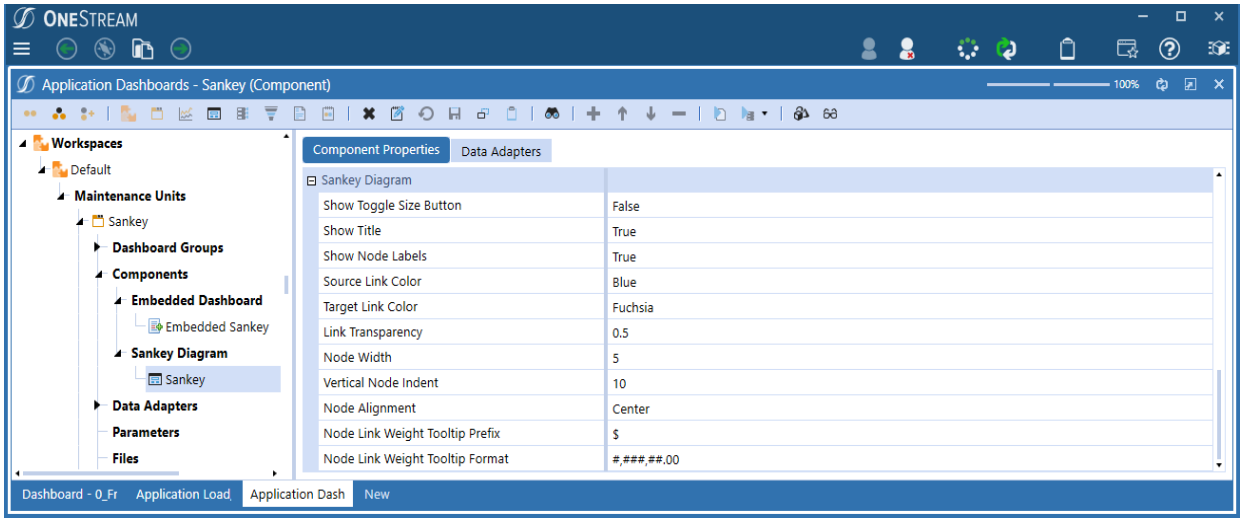
Diagrams can be exported as .png files by right-clicking outside of the links then selecting PNG.

Create a Sankey Diagram

1. From the OneStream Application tab, click **Dashboards**.
2. Expand **Dashboard Maintenance Units** and then expand the appropriate maintenance unit under **Workspaces**.

3. Click the **Components** label and then, from the toolbar, click **Create Dashboard Component**.
4. In the **Create Dashboard Component** dialog, click **Sankey Diagram (Windows App Only)** and then click **OK**.
5. In the Name field, type a name for your Sankey diagram.
6. Click **Save**.

Sankey Diagram Component Properties



Enhance the data presentation by adjusting the component properties. Navigate to the components: **Dashboards** > **Workspaces** > **Default** > **Maintenance Units** > **Sankey** > **Components** > **Sankey Diagram**

Show Toggle Size Button:When True, the toggle size button is available. When False, the toggle size button is unavailable.

Show Title: When True, the title is centered above the diagram. When False, no title is shown in the diagram.

Show Node Labels: When True, node labels display on the diagram. When False, node labels do not display on the diagram.

Source Link Color: When Default, source link colors automatically change with each refresh. Set a custom or preloaded color to maintain the same color.

Target Link Color: When Default, target link colors automatically change with each refresh. Set a custom or preloaded color to maintain the same color.

Link Transparency: Specify the level of transparency displayed on the diagram by selecting a value between 0 and 1. 0 is opaque and 1 is transparent. The default is 0.5.

Node Width: Specify the width of the bar for the node. Increase this number for a wider bar.

Vertical Node Indent: Specify the vertical space between nodes. Increase this number for more white space.

Node Alignment:Specify the vertical alignment to the top, bottom, or center of the diagram.

Node Link Weight Tooltip Prefix: Set the tooltip prefix as any keyboard character (!@#$%^&*()<>?) or alphanumeric text. This lets you input a currency culture like $ or €.

The example below sets prefix as $ if the user is set to English:

```
if (args.FunctionName.XFEqualsIgnoreCase("GetUserCultureCurrency"))
{
UserInfo userInfo = EngineUsers.GetUser(si);
CultureInfo culture = new CultureInfo(userInfo.UserPreferences.Culture);
return culture.NumberFormat.CurrencySymbol;
}
```

XFBR can be changed for culture. To get another user culture, replace GetUser(si) with GetUser (si, "nameOfUser").

Node Link Weight Tooltip Format:Specify the tooltip formating.

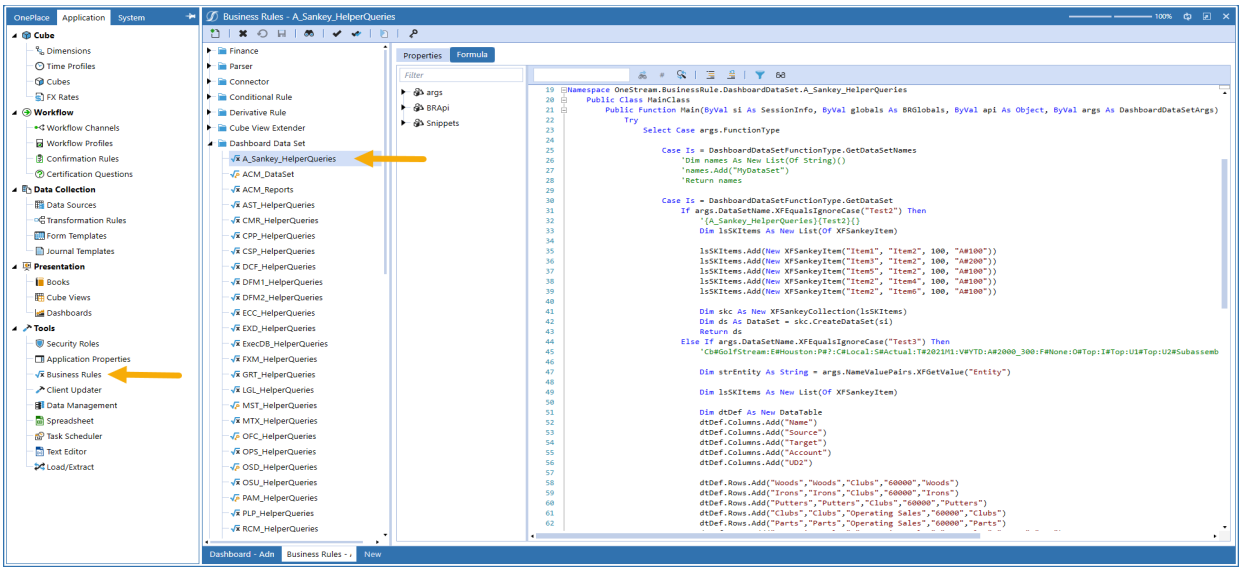
> **Example:** 0.00; #.##; #,###,###.00; 000.#'string'

See Standard numeric format strings.

Business Rules and Sankey Diagrams

When using a method command type business rule, in addition to defining the Sankey items, at least 4 columns must be defined from the data table:

- Name
- Source
- Target
- Value



> **NOTE:** Negative values are not recognized by Sankey diagrams. Links between items will not flow when a negative value is present.

Text Editor

This is used to view the Text Editor component in a dashboard. This is available in the Windows Application and can be displayed in the Modern Browser Experience.

Text Editor

Show Toggle Size Button

Defaults to True, which enables the toggle button to allow users to toggle the size of the component at run time. If set to False, the toggle button will not display in the component.

File Source Type

Workspace File

Displays a file stored in a Workspace Maintenance Unit's File section.

Application or System Database File

Displays a file stored in the Application or System Database.

File Share File

Displays a file from the File Share.

> **NOTE:** Application Database, System Database, and File Share exist in the File Explorer.

Url Or Full File Name

Specifies the URL or full name of the file being used.

Process Extensible Document

Set to True, which will process Extensible Documents when embedded in a dashboard.

Allow Create File

Defaults to True. If set to True, enables you to create files.

Allow Open File

Defaults to True. If set to True, enables you to open text files.

Allow Save File
Defaults to True. If set to True, enables you to save files.

Allow Refresh
Defaults to True. If set to True, the Refresh Document toolbar button will appear under the OneStream ribbon in the Text Editor component. If set to False, the Refresh Document button will not display in Text Editor Component under the OneStream Ribbon.

Show Ribbon
Defaults to True. If set to True, enables you to view the Ribbon below the Menu Bar. If set to False, hides the Ribbon.

Minimize Ribbon
Defaults to False. If set to True, minimizes the Ribbon below the Menu Bar. If set to False, the Ribbon displays as normal.

Text Viewer

This is used to view text and rich text documents similar to those created in Microsoft Word. This is available in the OneStream Windows App version. Documents cannot be created or edited using this tool.

Spreadsheet

This is used to display spreadsheet files created with the Spreadsheet feature in a Dashboard. See Spreadsheet in Application Tools for more information.

> **NOTE:** There are some configuration changes necessary for the Spreadsheet referenced in a Dashboard if functions such as XFGetCell or XFSetCell are included in that XLSX file that reference a custom Parameter that is driven by the Dashboard.
>
> In this case, the function available to Excel Add-in and Spreadsheet called XFGetDashboardParameterValue should be used. If that function is used within an XLSX file that is using a function like XFGetCell or XFSetCell (or similar) where these are referencing a custom parameter value (e.g. ParamEntity) that is on the Dashboard that references this Spreadsheet from within it as a Component.
>
> The practice to get this Custom Parameter value is to use XFGetDashboardParameterValue to fetch the text from that Parameter or its default value and place it in a cell on the Spreadsheet (e.g. B1). Then the cell that is using a retrieve function such as XFGetCell would reference this other cell (i.e. B1).

Spreadsheet

Show Toggle Size Button
Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

File Source Type

Dashboard File
Display a file stored in a Dashboard Maintenance Unit File section.

Application/System Database File
Display a file stored in the Application or System Database Share.

File Share File
Display a file from the File Share.

Url or Full File Name

Displays the URL or Full name of the file being used.

Process Extensible Document

Defaults to True. It will process Extensible Documents when embedded in a dashboard.

Refresh Spreadsheet When Opened

If set to True, will automatically refresh the spreadsheet file when it is opened.

Allow Create File

If set to True, allows users to create spreadsheet files.

Allow Open File

If set to True, allows users to open spreadsheet text files.

Allow Save File

If set to True, allows users to save spreadsheet files.

Allow Submit Data

If set to True, the Submit button on the Ribbon is enabled and users can load data to the forms channel.

Allow Calculate Data

If set to True, the Calculate button on the Ribbon is enabled and users can call calculate options using it.

Show Ribbon

Select True to enable users to see the Ribbon below the Menu Bar, select False to hide the Ribbon.

State Indicator

This is used to indicate a specific status on a Dashboard. This Component is not compatible with Mobile Dashboards.

Formatting
See Label

Data Table Cells from Adapter
See Label

States

State Indicator Type
The type of image used for indicating the status. Valid states start at index 0 and include the following:

Lamp
Off, Green, Yellow, Red

Arrow
Up, Down, Left, Right Arrow is displayed

Smile
Very Happy, Happy, Neutral, Sad

Traffic Lights
Off, Green, Yellow, Red

Default State Index
The default state is 0-4.

State Selection Type (Image = Off)
Specify the State selection type and then define the Minimum Amount, Maximum Amount and/or a list of comma-separated text items that the value from the first Data Table Cell must satisfy in order for the State to be selected.

(Not Used)
The State is not used.

Minimum Amount
The minimum amount for the State to be selected.

Minimum Amount Inclusive
The minimum amount including the specified value for the State to be selected.

Maximum Amount
The minimum amount for the State to be selected.

Maximum Amount Inclusive
The maximum amount including the specified value for the State to be selected.

Range
The amount between two values for the State to be selected.

Range Minimum Amount Inclusive
The amount between two values including the minimum amount for the State to be selected.

Range Maximum Amount Inclusive
The amount between two values including the maximum amount for the State to be selected.

Range Minimum and Maximum Amount Inclusive
The amount between two values including both the minimum and the maximum amount for the State to be selected.

Equals Text Item
If the string equals the string specified in the Text Items field, it should be selected for the specified State.

Starts with Text Item
If the value starts with the string specified in the Text Items field, it should be selected for the specified State.

Ends with Text Item

If the value ends with the string specified in the Text Items field, it should be selected for the specified State.

Contains Text Item

If the value is listed in the Text Items field, it should be selected for the specified State.

Minimum Amount

The value used in State tests for minimum amount.

Maximum Amount

The value used in State tests for maximum amount.

Text Items

The text values used in the types: Equals Text Item, Starts with Text Item, Ends with Text Item & Contains Text Item.

Web Content

This is used to display a URL or file embedded in a OneStream Dashboard. It can be used with Mobile and Application Dashboards.

> **NOTE:** This component uses *Microsoft Edge WebView2 Runtime* to embed the web content. It must be installed on the client device. If the client device does not have it installed, you will receive an error when accessing a dashboard that is configured with a Web Content component.

Web Content Type

URL

The URL to a web page. This can be internal or external.

Dashboard File

The image displayed on the button is based on a file stored in the Dashboard Maintenance Unit File Section.

Application Database File
Display a file from the Application Database share.

System Database File
Display a file from the System Database share.

File Share File
Display a file from the file share.

URL or Item Name
Item to display (URL or file) in the control.

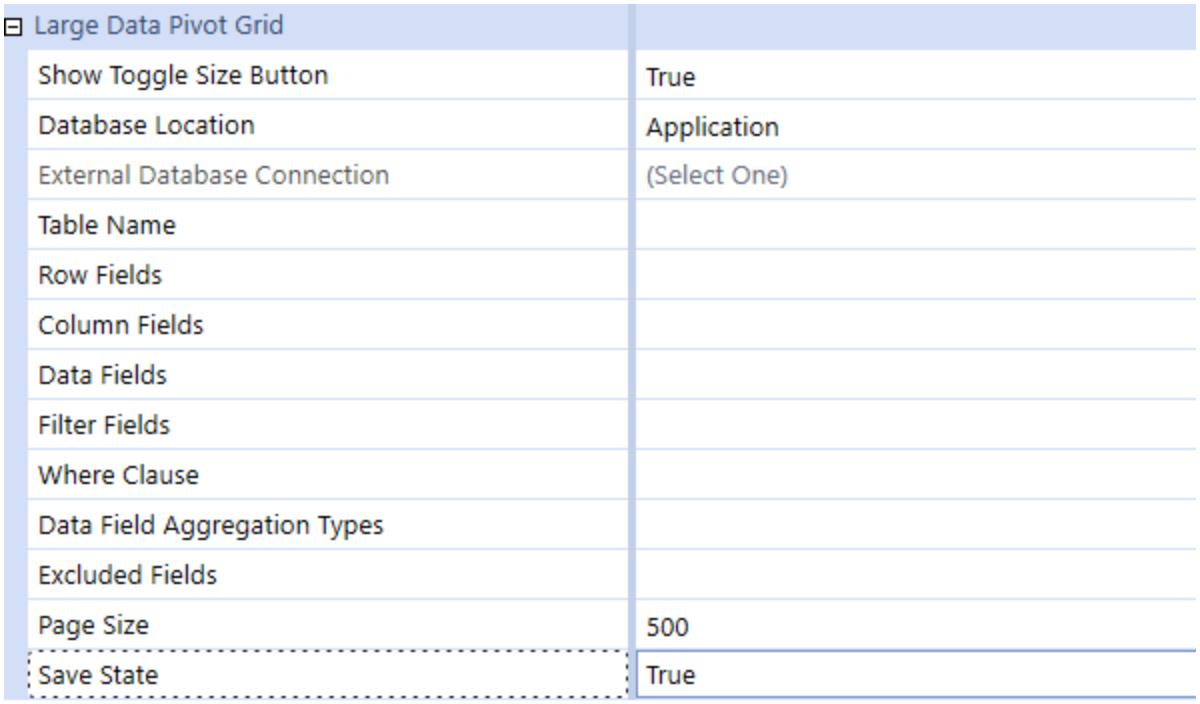
Show Borders
Select False to remove the border around the Web Content dashboard component. This property defaults to True.

Show Header
Select True in order to display a header derived from the Component's Name or Description.

Show Toggle Size Button
Select True to enable the toggle button and allow users to toggle the size of the Component at run time, select False to hide the toggle button.

Large Data Pivot Grid

The Large Data pivot grid is a dashboard component that lets you connect to external tables or large database tables for "pivot" style analytic reporting. The grid helps designers seamlessly integrate data from external tables into a OneStream environment through a dashboard for analytic reporting. The grid's paging and server based processing features support very large data sets/tables. In some cases, this component requires server configuration settings so users can access specific database tables.



You can specify the following properties:

Property	Setting
Show Toggle Size Button	True/False, to activate resize button.
Database Location	Identify the type of table location, External, Application or Framework
External Database Connection	Database Connection defined in the Application Server Configuration

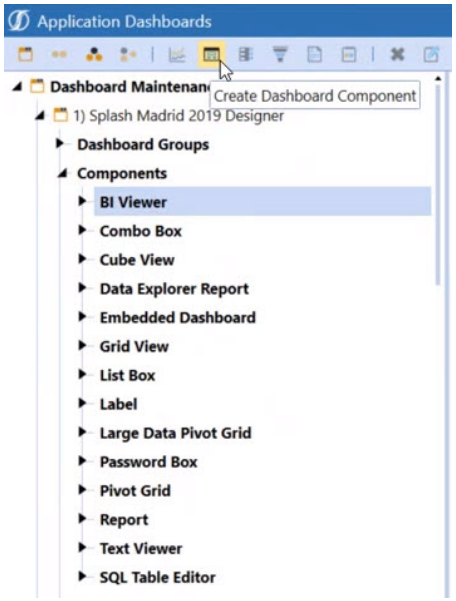
Property	Setting
Table Name	Table Name to be retrieved
Row Field	Comma-separated database column names to be placed in Row area by default
Column Fields	Comma-separated database column names to be placed in Column area by default
Data Fields	Comma-separated database column names to be placed in Data area by default
Filter Fields	Assign Dimensions as a default filter and assign filtering
Where Clause	Used to assign global filters to focus the returned results from the source table
Data Field Aggregation Types	List of comma-separated key value pairs that specify one aggregate function type per data field. Supported Sum, Average, Min and Max.
Excluded Dimensions	Allows Database Column Dimensions to be excluded from the Pivot
Page Size	Number of records returned in a page. Defaults to 500 with a maximum value of 3000.

Property	Setting
Save State	Enables the Save Button on the Large Data Pivot Grid when set to True and Disables the Save button when set to False

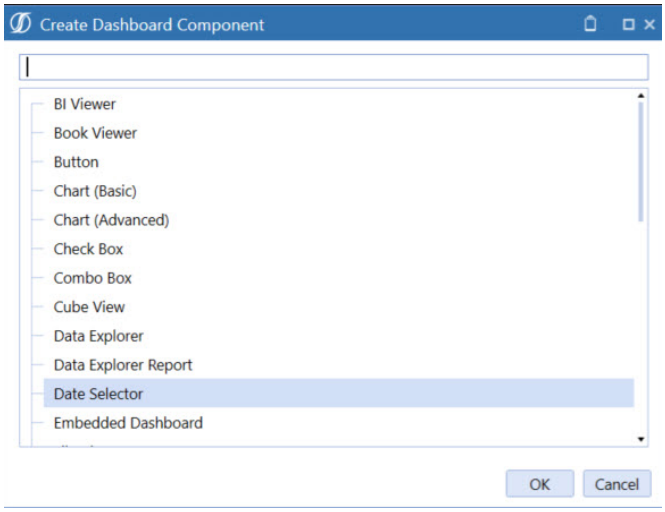
Date Selector

You can set a date in a dashboard by creating a date selector component.

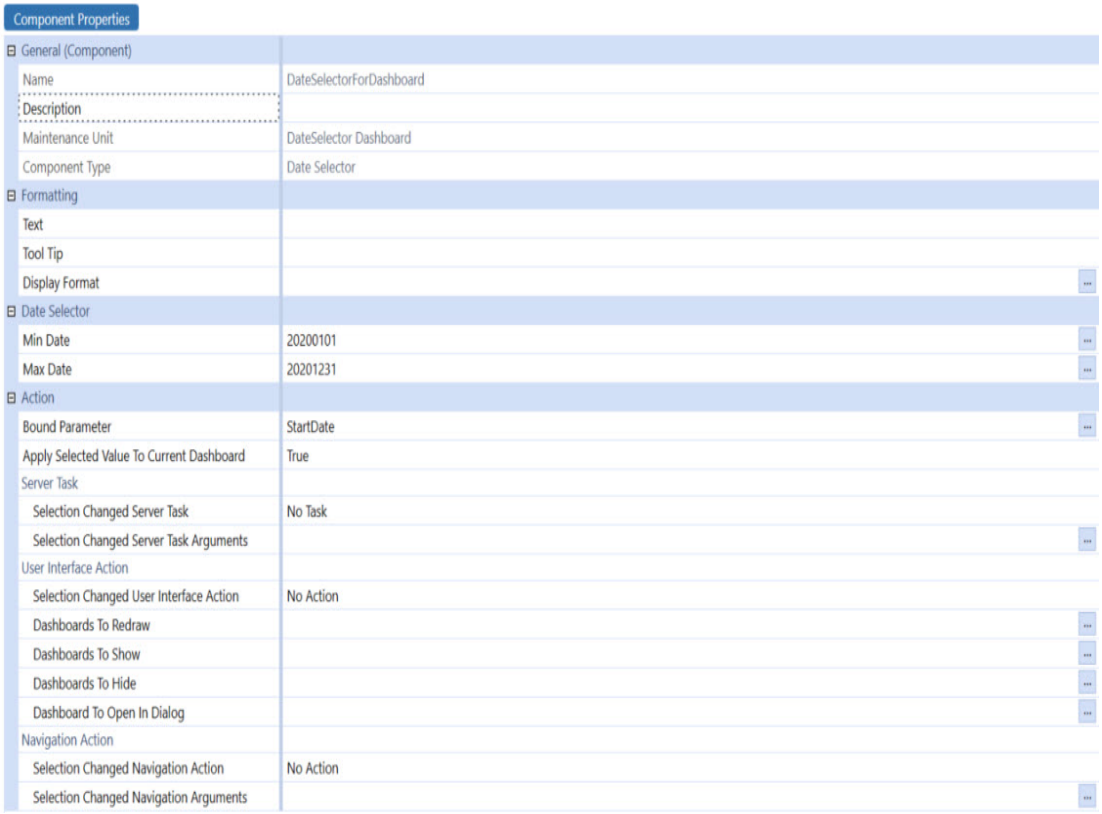
1. Go to **Presentation** > **Dashboards.**
2. Select a dashboard under **Dashboard Maintenance Units.**
3. Click **Components**.
4. Select a component and click **Create Dashboard Component**.



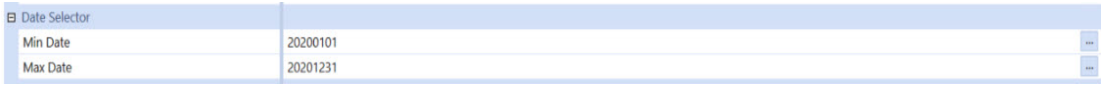
5. **Click Date Selector**



6. Enter the properties.

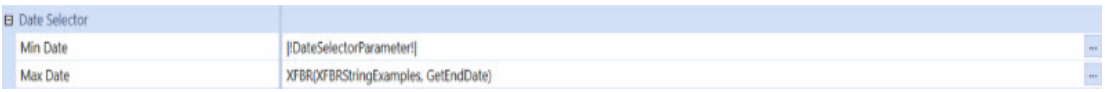


7. When you set the properties at design time, you set a minimum and maximum date range that is viewed at run time. For example, **Min Date** to **20200101** and the **Max Date** to **20201231**, others can only select dates within that range.



if you click the ellipsis, you can choose an existing parameter which will automatically fill in the**|!...!|**.

You can enter and use a Dashboard **XFBR** string.

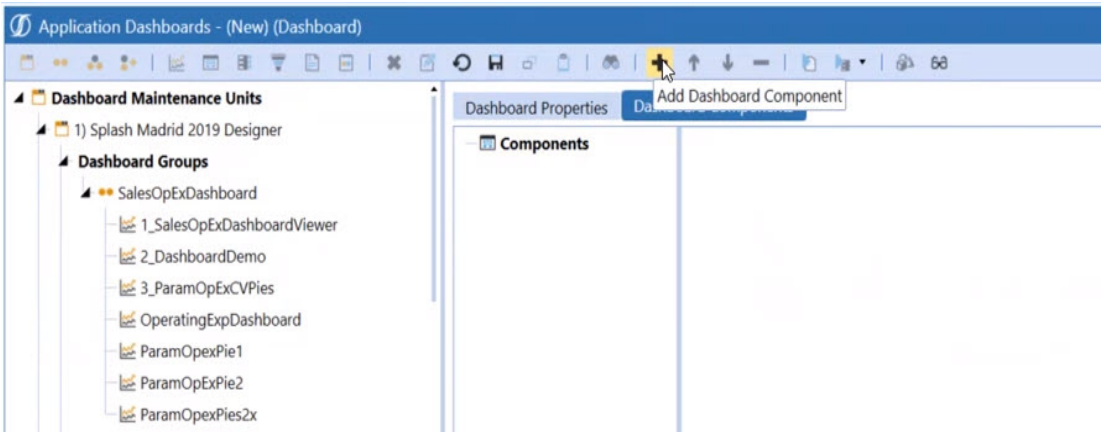


> **NOTE:** Min and Max Date values should be entered in the following format: yyyyMMdd. This value is converted at run time based on your application culture setting.

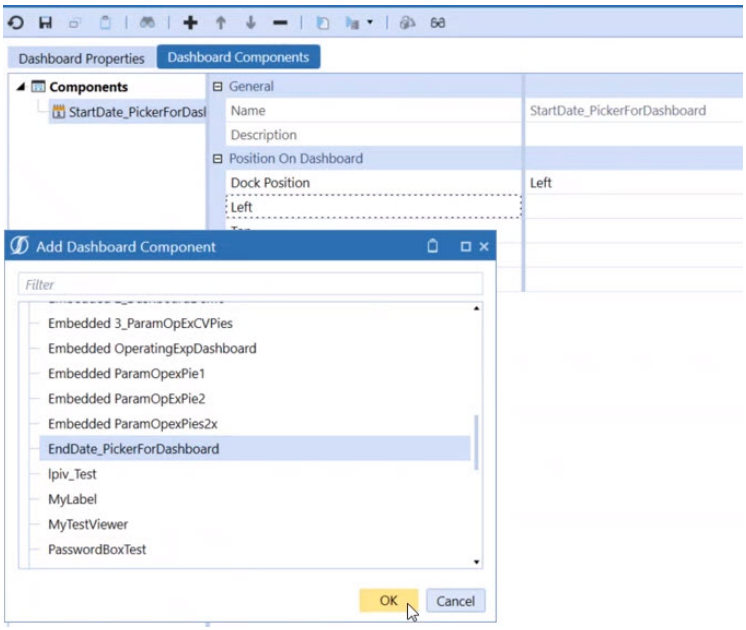
Add the Date Selector Component to a Dashboard

Once the date selector dashboard component has been created, you can add it to a dashboard.

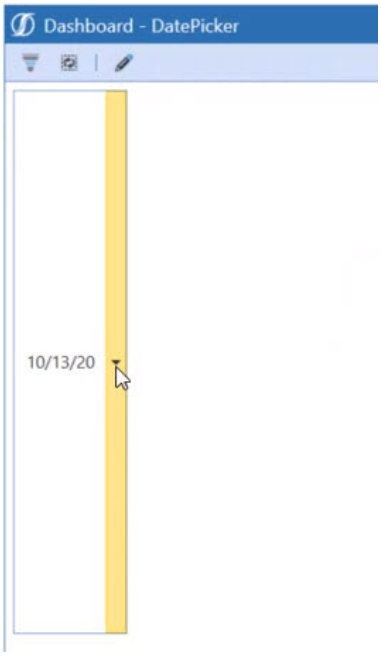
1. Select the dashboard and click **Add Dashboard Component.**



2. Select the date selector component and click **OK**.



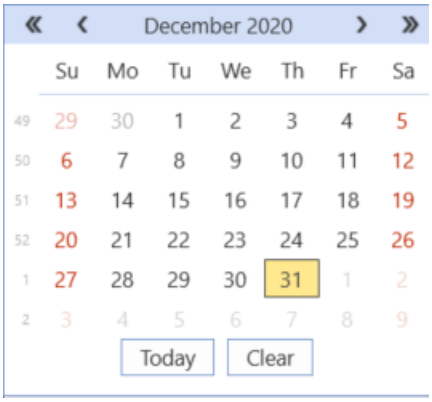
3. Click **Save** and **View Dashboard**.



4. Click the arrow to show the calendar.

Use the double arrows >> to show the max date and << to show the min date.

> **NOTE:** If no min date is specified, the min date is set to 1/1/1900. If no max date is specified, the max date is 12/31/9999.



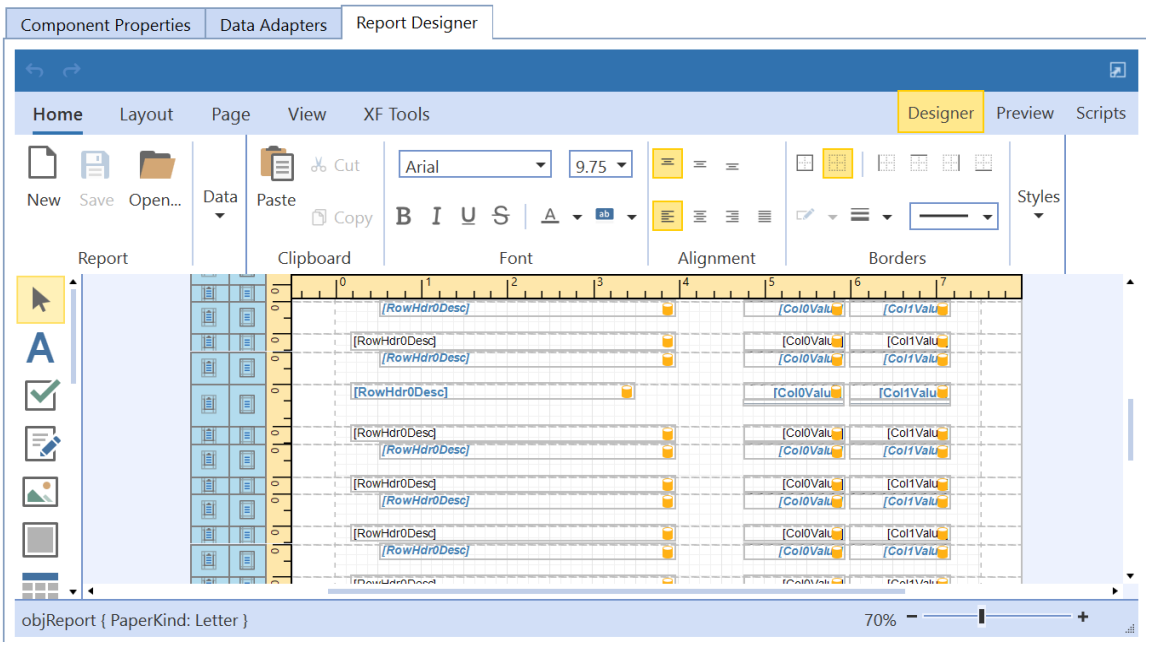
Report

The Report Designer is built into the dashboard report component, which allows you to edit reports on the report itself.

> **NOTE:** The `OneStreamClientApi.dll` is now included with the Windows application install. The OneStream Windows application includes the Client API installer.

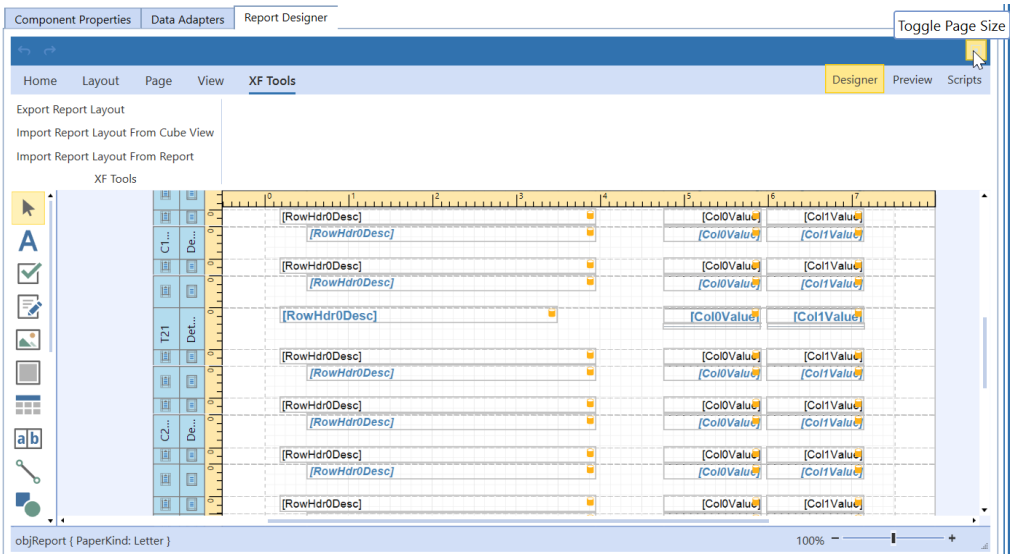
Access the Report Designer

1. Go to the Application tab, then click **Presentation**> **Dashboards**.
2. Scroll to a maintenance unit and select **Components**.
3. Expand **Report** and open a specific report.
4. Click the **Report Designer** tab.

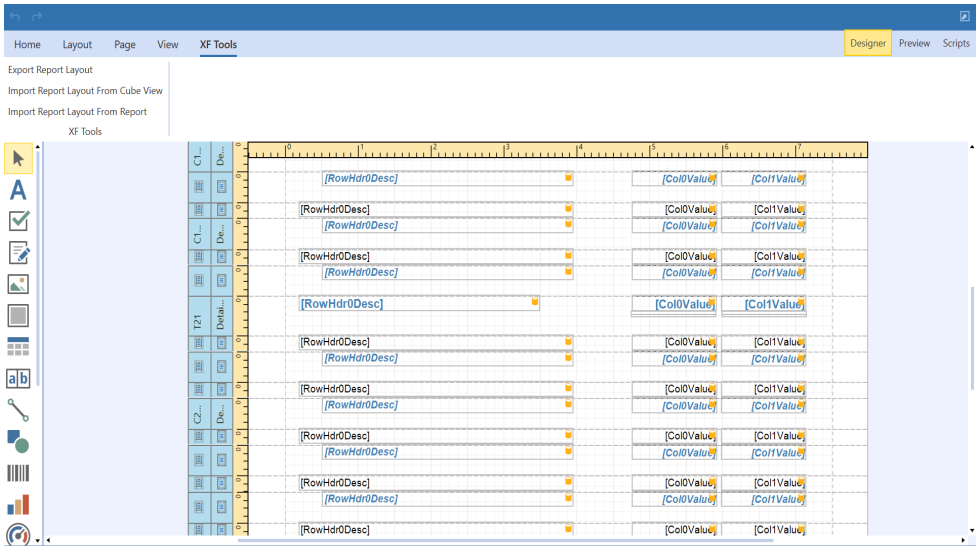


Toggle Page Size

1. Click **Toggle Page Size** in the upper right corner of the Report Designer to maximize the Report Designer as full screen.
2. Setting the display **Zoom** slider will assign the display percentage to the Maximize view.



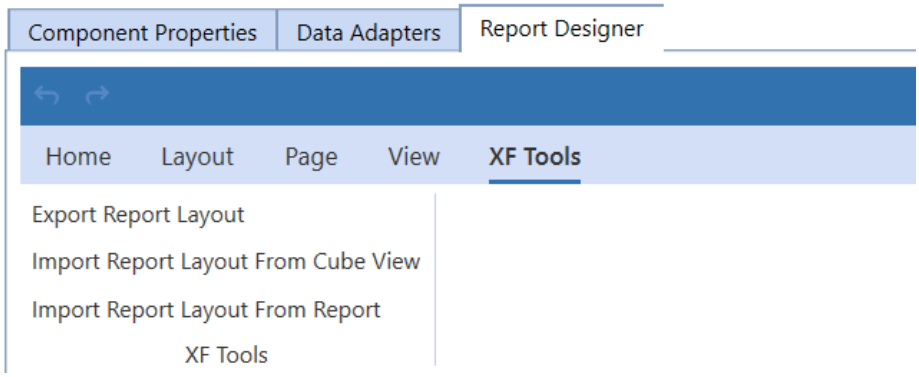
The report changes to full screen.



Export Report Layout

Export the report `*.repx` file.

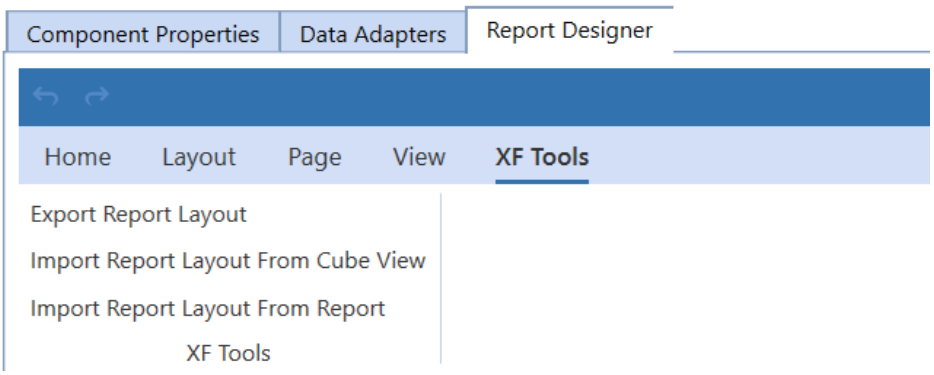
1. Click the **XF Tools** tab and then click **Export Report Layout**. The **Save As** dialog box opens.



2. Browse to a location to save the report and click **Save**.

Import Report Layout from Cube View

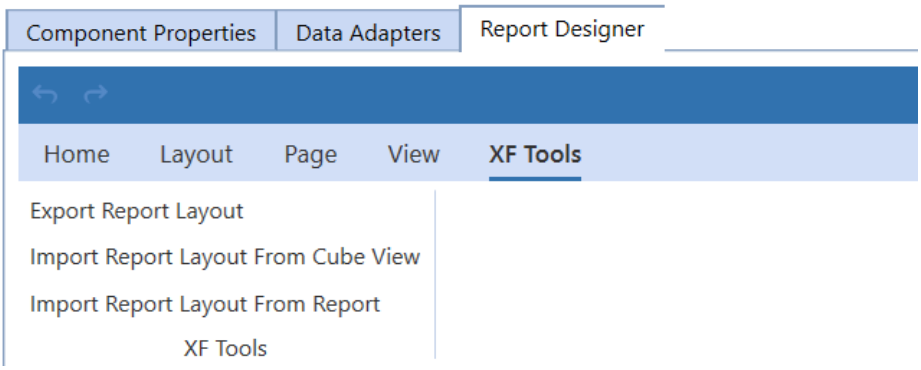
1. Click the **XF Tools** tab and then click **Import Report Layout from Cube View**. The **Object Lookup** dialog box opens.



2. Browse to a cube view and click **OK**.
3. The report layout changes to the formatting of the selected cube view.
4. Click **Save** to save the new report layout or click **Cancel All Changes Since Last Save** to revert to the previous layout.

Import Report Layout from Report

1. Click the **XF Tools** tab and then click **Import Report Layout from Report**. The **Object Lookup** dialog box opens.

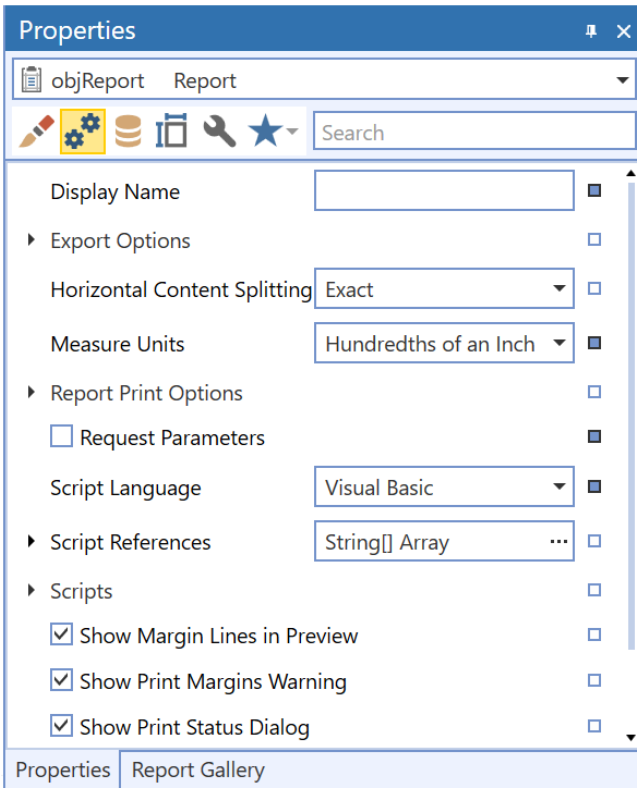


2. Browse to a dashboard report component and then click **OK**.
3. The report layout changes to the formatting of the dashboard report component you selected.
4. Click **Save** to save the new report layout or click **Cancel All Changes Since Last Save** to revert to the previous layout.

Functionality

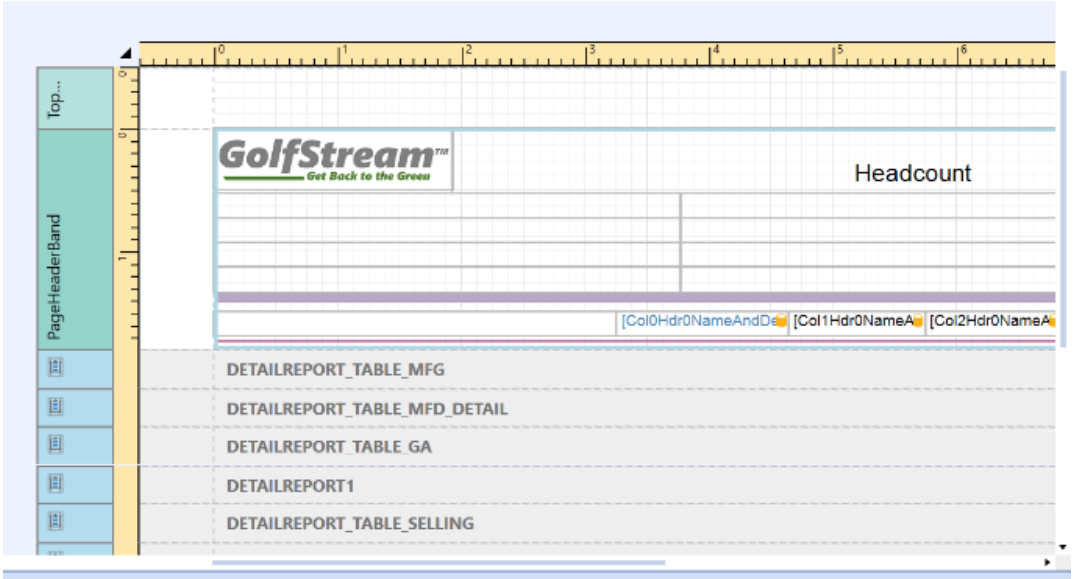
For most right-click functionality in Report Designer, use the **Properties** dialog box.

> **NOTE:** When editing sub reports, you can access the Generate Own Pages option from the Properties menu.



Display Band Details

To expand any of the bands, double-click the icon in the left column.



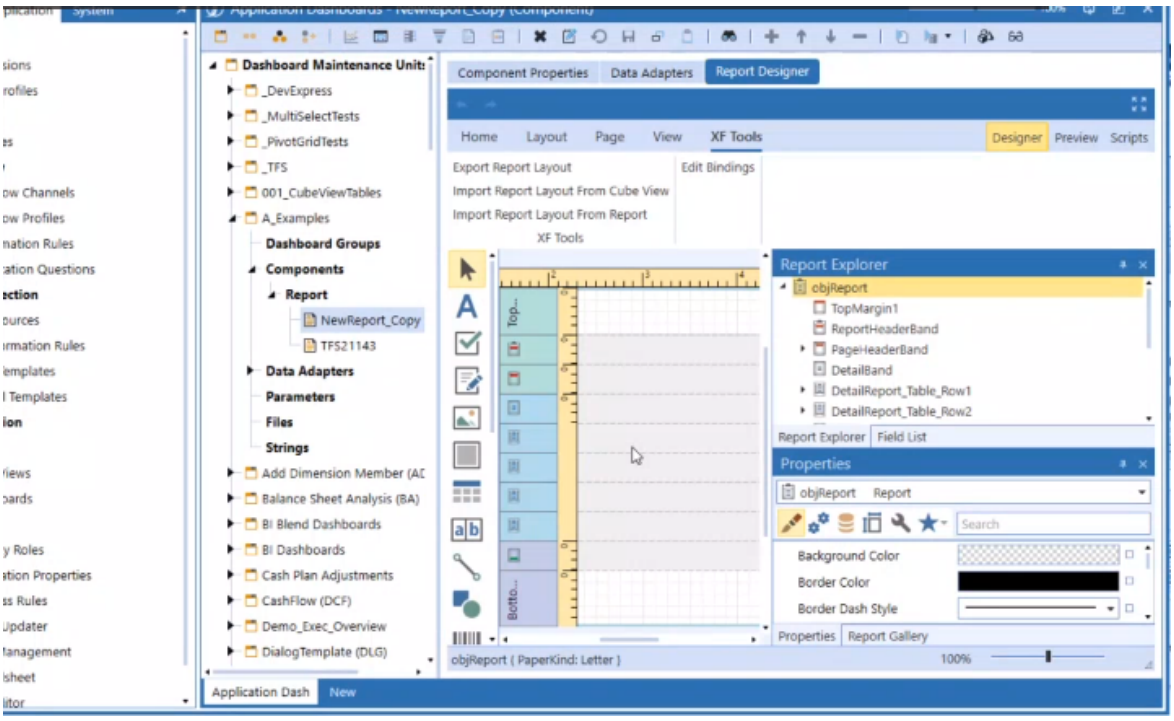
Edit Bindings by Report

Edit bindings directly and evaluate all the controls with the built-in component in the Report Designer. The Edit Bindings dialog box lists all of the controls, property type and data binding. It displays invalid status and allows you to apply corrections.

The Property type contains type definitions.

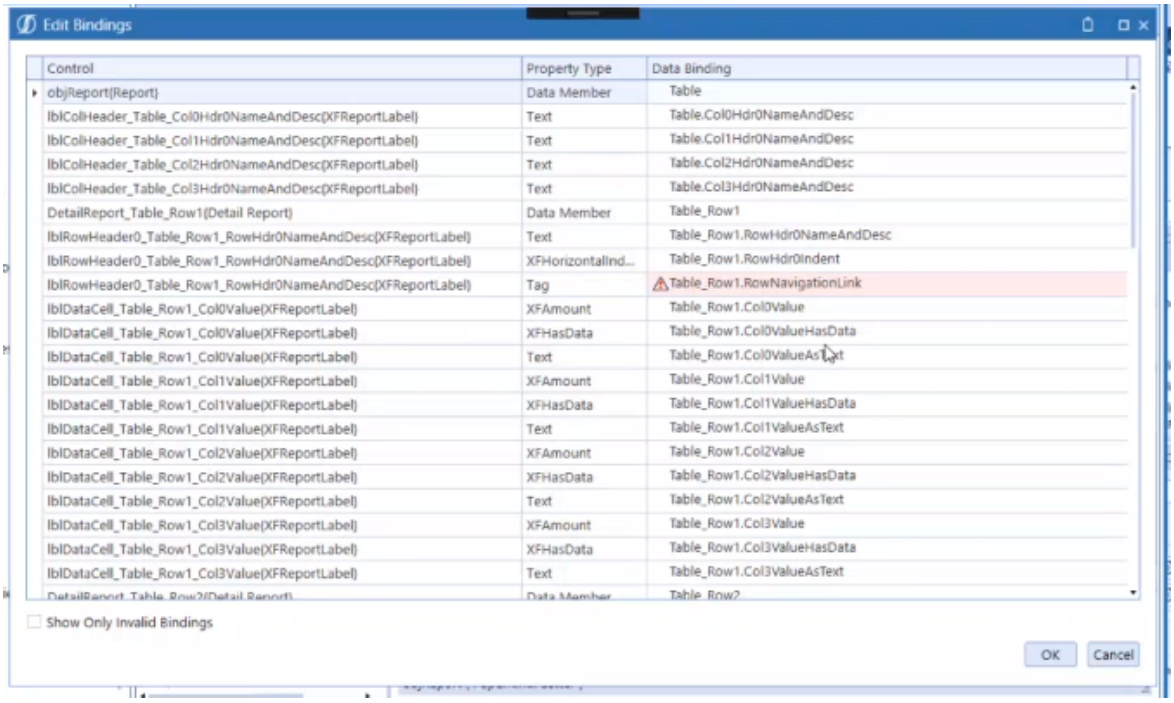
Access Edit Bindings

1. From the Report Designer tab, select **XF Tools**, then **Edit Bindings**.



The **Edit Bindings** dialog box opens.

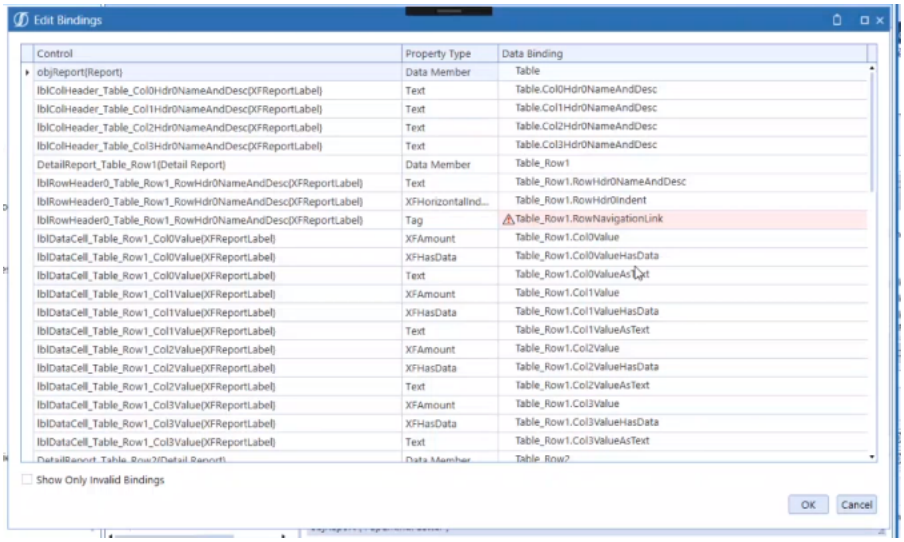
2. Click any of the column headings to sort the data in the column.



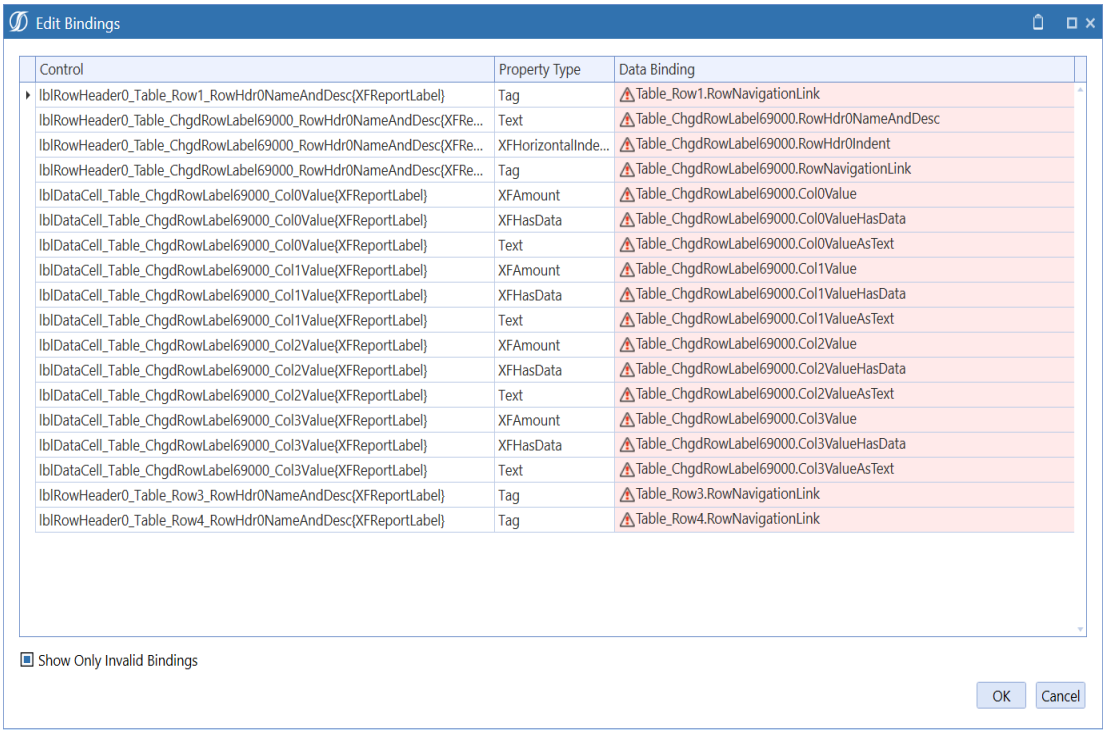
Show Invalid Bindings

The **Edit Bindings** dialog box lists invalid bindings. The source binding is set to a path that does not exist in your data set.

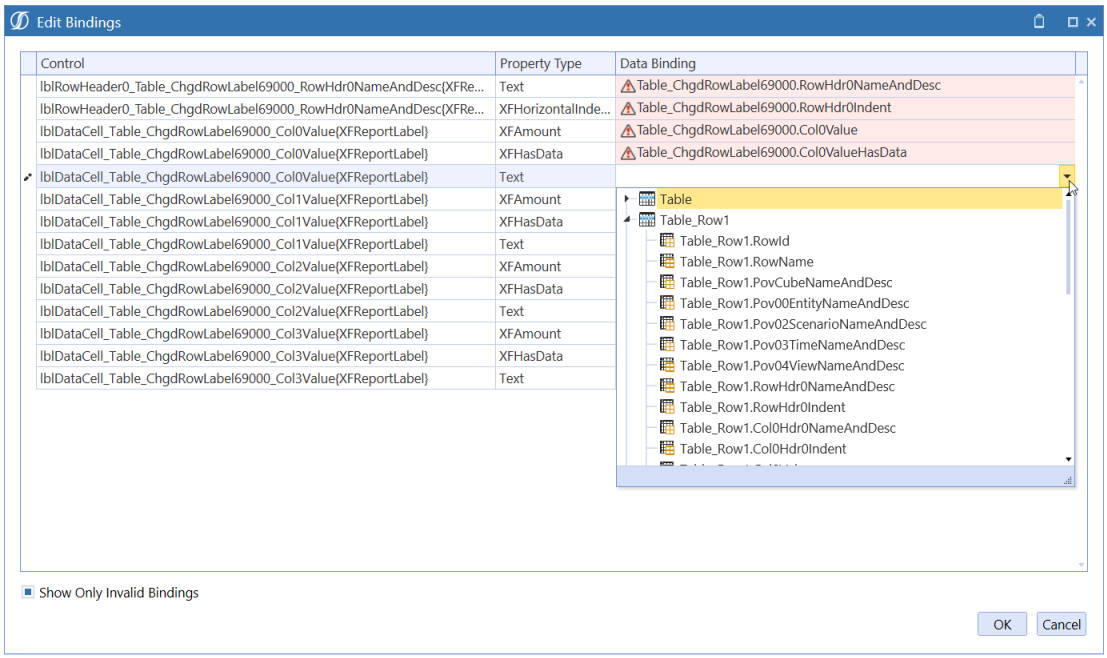
1. From the **Edit Bindings** dialog box, click **Show only invalid bindings**.



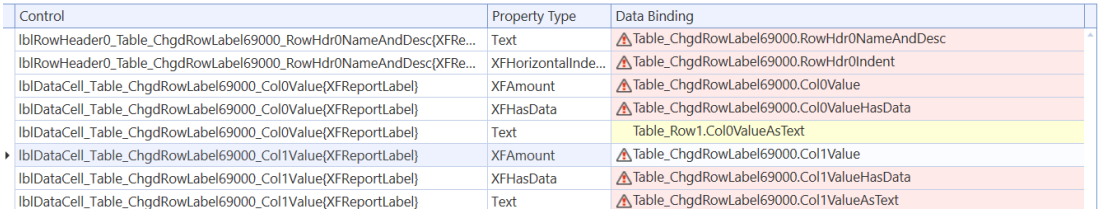
A list of invalid bindings opens. The invalid bindings have a red background.



2. Select one of the invalid bindings from the Data Binding column and click the down arrow.



3. From the drop-down list select the correct data to a path that exists in your data set.
4. After you click off of that row, the color of the row changes to yellow to identify that it is modified.



> **NOTE:** Even if you modify an invalid binding to make it valid, it still appears in the invalid list.

5. Click **OK** to close the **Edit Bindings** dialog box.

6. Click **Save** in the Report ribbon to save your changes to the report and database. Or, click **Undo** to revert the changes without saving.

SQL Table Editor

Displays a grid linked to a SQL table and the contents of the table are shown in the grid. Settings in the object determine whether the grid allows changes to the table data. This Component is not compatible with Mobile Dashboards.

Action
See Button

SQL Table Editor
Database Location

Application
Select if the table to be displayed is located in the application database.

Framework
Select if the table to be displayed is located in the framework database.

External
Select if the table to be displayed is located in an external database.

External Database Connection
Connection information to enable connecting to an external database. This is only needed if the Database location is set to External.

Table Name
The name of the table being displayed in the control.

Where Clause
The SQL string where clause used to pull data from the table.

Table Column Names

Optional. A comma-separated list of database table column names you can use in the SQL Select statement. Only the entered columns are displayed in the control.

> **NOTE:** If editing existing data or adding new rows is enabled, any columns that exist in the underlying database table are excluded from the list. This property must have default constraints set in the database table.

Order by Clause

The SQL string to order the results of the table.

> **NOTE:**
> The Order by Clause property does not allow SQL Functions, like YEAR(effectiveDate). When run inside a dashboard, the Table Editor's columns allow clicks to change the order to ascending, descending, or no sort. SQL functions can not be allowed in the ordering.

Filter Mode

The SQL Table Editor lets you filter results using Filter Mode. This filtering mode is built into the header cell of each filterable column. It is simpler to use than Popup mode (an alternative filtering mode) and uses only one field editor to filter. All filters start as empty until a value is entered into the editor.

Default for Columns are Visible

If set to True, all columns will be hidden unless overridden on an individual column. Settings are True or False.

Default for Allow Column Updates

If set to True, the data in the existing table can be modified, if set to False, it cannot be modified.

Show Title Header

Show or Hide entire bar including the items within the bar. If you set it to False, the Default For Allow Column Updates, Process Selection Changed For Inserts, Allow Inserts, and Allow Deletes remain enabled.

Show Title

Show or Hide the text in the Title Header.

Show Column Settings Button

Shows (if set to True) or Hides (if set to False) the Column Setting Button which allows user to reorder and set visibility of columns.

Show Deselect All Button

Shows or Hides Deselect All Button which allows the user to deselect all selected rows.

Process Selection Changed For Inserts

If set to True, (and the “Allow Inserts” property is also set to True), it will allow the Insert button to process the selection changed event which is often configured to refresh a portion of the Dashboard. If set to False, it will not allow the Insert button to process the selection changed event.

Allow Inserts

If set to True, new rows of data can be added to existing data in the table. If set to False, new rows cannot be added.

Allow Deletes

If set to True, rows of data can be deleted from the existing data in the table. If set to False, rows cannot be deleted.

Retain Table Column Order

If set to True, the column order can be changed via the SQL Table Editor Column Format section. If set to False, the order cannot be changed.

Read Only Text Color

This field sets the default color to standard black text. The setting can be changed to any color including a system color (e.g., XFReadOnlyText).

Column Name for Bound Parameter

Specify the name of the database column to use to change the value of the Bound Parameter when a database row is selected. This is used when a SQL Table Editor is being used to affect the display of other Dashboard Components (e.g., when showing detailed information for the selected row.)

Allow Multiselect

If set to True, the multiple selection of records will be enabled by item selection or using a selection check box. The active selected items will be passed to the defined Bound Parameter field as a comma delimited list. The Bound Parameter format will be as: item1, Item2, Item3. If the 3 values A, B, C,D are selected (where C,D is one value) the resulting bound parameter string is A, B, "C,D". If set to False, only a single Bound Parameter can be passed at a time and the selection boxes will be deactivated from the user interface.

Rows Per Page

Specify how many rows to display before a page break. This is a numeric value between 1 and 3000. By default, this property is set to -1, which turns OFF the property and uses your defined user preference property Grid Rows Per Page. Any value greater than 3000 results in a maximum value of 3000.

Save State

If set to True, user settings on the Component are retained. User preferences for Columns saved will be for: Order, Visibility, Filtering, Freeze Bar, Sorting and Widths.

> **NOTE:** Reset Save State Back To Default: Right-click on the Dashboard to enable the “Reset State” to return back to its Component Properties settings.

Vary Save State By

Apply Vary Save State By to workflow profile settings and scenario. The current Save State elements will have the control state tied to the current Workflow Profile and Workflow Scenario settings. If you select **False**or **Use Default**, Vary By Save State will not apply.

When Vary Save State By is enabled, the related Dashboard Component will have the additional reset option of **Reset All States**, which can be used to clear the user's Save States across all the Vary Save State By parameters.

Display Format

Sets the SQL Table Editor formatting options such as row color. Set row **BackgroundColor** property to a color, and set **AlternateRowBackgroundColor** to a different color.

Data Manipulation Buttons

Show Data Manipulation Buttons

Hide or Show Insert Row(s), Delete Row(s), Cancel All Changes Since Last Save and Save. Settings are True or False.

Show Insert Row(s) Button

Hide or Show Add Row Button. Settings are True or False.

Show Delete Row(s) Button

Hide or Show Remove Row Button. Settings are True or False.

Show Cancel All Changes Since Last Save Button

Hide or Show Cancel Button. Settings are True or False.

Show Save Button

Hide or Show Save Button. Settings are True or False.

Server Task

Create Table if Necessary

The option to create the table if it does not already exist. Settings are True or False.

Table Creation Script
The SQL script to create the new table.

Save Data Server Task

No Task
There is not a performed task for a SQL table save data event.

Execute Dashboard Extender Business Rule
This will use a Dashboard Extender Business Rule to perform a save data event.

Save Data Server Task Arguments
Enter the arguments required by the server task.

Dashboard Extender Business Rule Example (each enclosed in curly braces)
{Business Rule Name}{Function Name}{Optional Name-Value Pairs}

> **TIP:** The data will wrap on string spacing. If there are no blank spaces, the column data will not wrap. Additionally, keyed fields cannot be wrapped.

DataFormatString
Specify a number/date format to the data in the column. For example, mm/dd/yyyy will return the current Month/Day/Year using a slash. MM-dd-yyyy will return the Month-Day-Year using a dash. N0 will return a number without a decimal point, and #,###,0 will return a number without a decimal and a comma depicting the thousandth place. See Application Properties in Application Tools for more examples of number formats.

Width
This specifies the default column width to be displayed.

SQL Table Editor Column Format

ColumnName
Column name from the table where the selected formatting is to be applied. Specified name must match the column name from the table exactly.

Description
Column description to be displayed. By default, the Column name from the table is displayed.

Column Display Type
See Grid View for Column Format

IsVisible
Setting to override the default Columns Visible setting. Settings are True, False, or Use Default

AllowUpdates
This is a setting to override the Default For Allow Column Updates setting for each individual column. If set to True, users can modify data displayed in the grid. The Use Default setting uses the Default For Allow Column Updates setting.

ParameterName
The assigned Parameter name to be used to store the Parameter value from the specified column.

DefaultValue
The default value to be entered when allowing new records to be added. This helps ensure that invalid blank cells are not created.

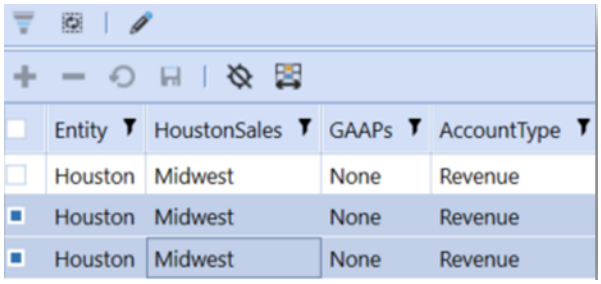
IsMultilineText
If set to True, this will allow columns to display data on multiple rows if the column is not wide enough to display the full value.

SQL Table Editor Multiselect

Enabling the Allow Multiselect on the SQL Table Editor Component will expand the functionality of the Component for use in interactive dashboards.

- Allow Multiselect will generate the Bound Parameter as a comma delimited list.
- Selection Methods can be performed with the Select All check box, selecting a range using Shift key, and by item using Ctrl-Click.
- Deselecting can be done by item or using the Deselect All toolbar button.
- Applying a Column Filter after selections are activated is not allowed. Applying a column filter will clear any existing selected records. The column must be first filtered, once filtered, selections can be performed.
- Defer Refresh button can be used to manually control the execution of tasks defined in the Actions properties of the SQL Table Editor Component. This user-controlled execution of actions may provide better performance within complex dashboards.

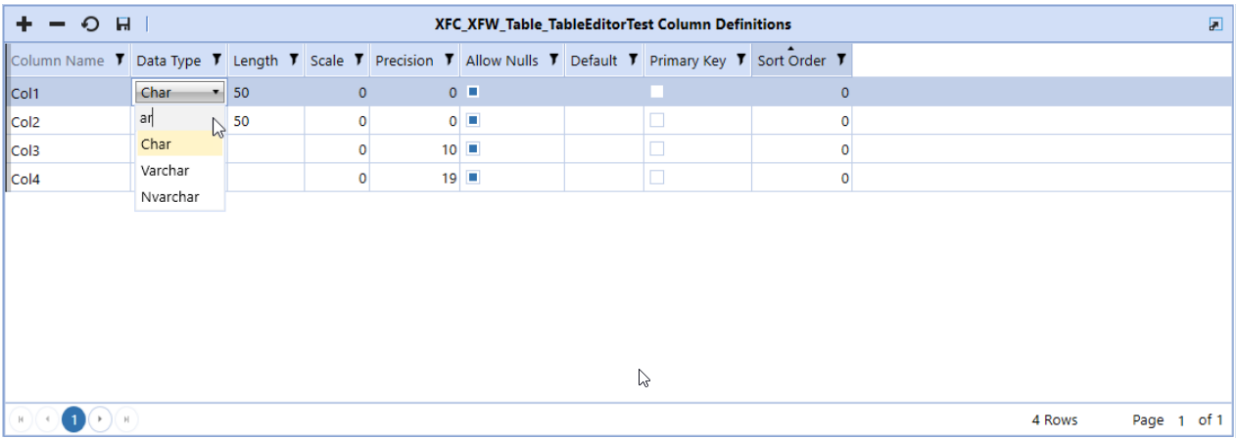
> **NOTE:** Defer Refresh remains checked after the Refresh button is clicked in SQL Table Editor.



> **NOTE:** Multi-select retains selected items after column filtering and sorting by using a table's primary key columns. If the referenced table does not contain a primary key column, the selection is not retained across pages or after filtering or sorting columns.

SQL Table Editor Searchable Drop-Down List

When drop-down lists have many items to choose from, finding a suitable item in the list can be cumbersome. You can make the drop-down list searchable to improve the efficiency of finding the appropriate item. As you type, search options are narrowed down to those that contain the typed text and you no longer have to scroll through a long list.



> **NOTE:** To use this search feature, the table editor must be configured so that you can edit existing data or add new rows. Columns must be configured with a parameter of type Delimited List, Member List, or Bound List, which is what configures a column as a standard drop-down list.

Enable the Search

1. From the Application tab, select **Dashboards**, and expand the appropriate **Workspace**, **Maintenance Unit**, and **Component** until you reach the SQL Table Editor component you want to modify.
2. Select the **SQL Table Editor** to view its properties.
3. Enable the search feature.

- For all columns that are configured with a list parameter:
    a. In the SQL Table Editor section, set **Default for Show Search** to TRUE.
    b. Click **Save**.
- For specific columns only:
    a. In the SQL Table Editor Column Format section, locate the column where you want to enable the search feature and click the **ellipses (...)**.
    b. In the dialog box, set the **ShowSearch** property to TRUE.
    c. Click **OK**.
    d. Click **Save**.

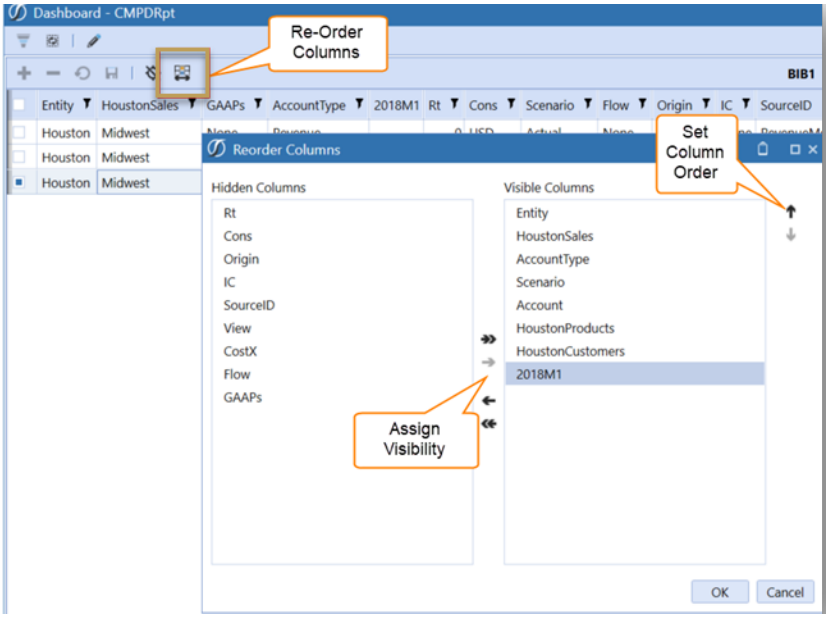
Use the Search

1. Run the dashboard that contains the SQL Table Editor component.
2. Double-click in a cell to expand the drop-down menu.
3. Start typing in the search field. The drop-down list narrows down to items that contain the typed text.

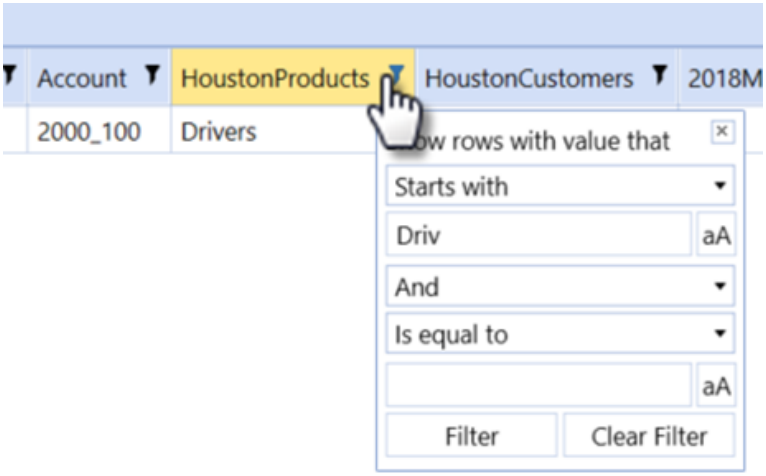
SQL Table Editor Save State User Preferences

Use Save State to retain your user preferences for the SQL Table Editor display. Your preferences for the following are saved if you set Saved State to True:

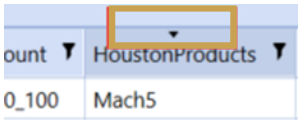
- Column order
- Visible or hidden columns



You can set column filters Column Filters can be set by the user and will be preserved if the Saved State is set to True.



However, the Column Sort will not be preserved once the SQL Pivot Grid is deselected or the dashboard is closed.



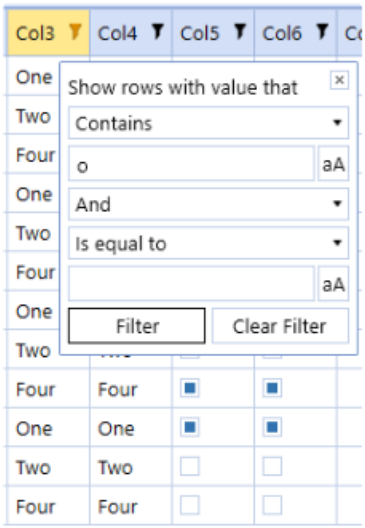
Save State for Freeze Bar

When Save State is set to **True** the Freeze bar will retain its position if you drag it to a certain column.

SQL Table Editor Filtering

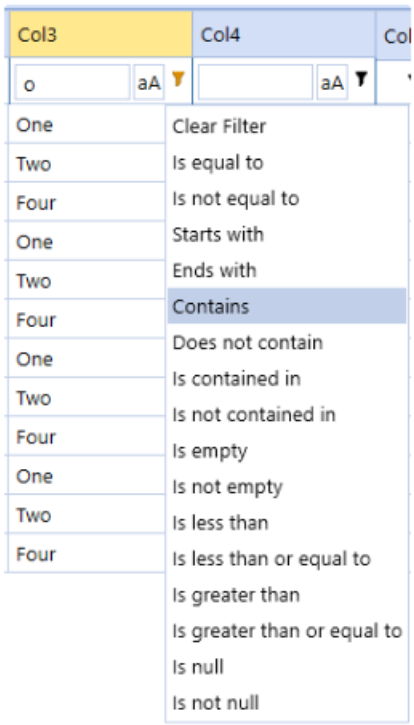
The SQL Table Editor lets you filter results using the following two methods, or modes:

- **Popup Mode**: This is the default filtering mode and has been available for use in all prior releases of the OneStream software application. Using this mode, you can activate filtering by clicking the funnel icon in the column header, which launches a popup window.



The popup contains two field filters that let you create filtering criteria such as "value is greater than 18 and is less than 60." Field filters consist of two parts: an upper part, which is a combo box, and a lower part called the field filter editor. In the upper part (combo box), you can use filter operators like **Is Equal To** or **Is Less Than,** which change based on the data type of the column. The lower part (field filter editor) provides filtering criteria based on the data type of the column. For example, if you have a date column, you can use a date/time picker control as the filtering criteria.

- **FilterRow Mode**: This filtering mode is built into the header cell of each filterable column. It is simpler to use than Popup mode and uses only one field editor to filter. All filters start as empty until a value is entered into the editor. Click the filter icon to display a drop-down list of all operators available to the data type of the column. If you select **Clear Filter** from the list, it will clear and reset the column filter.



You can configure the Filter Mode property of the SQL Table Editor to use either of the modes. To configure the SQL Table Editor:

1. Navigate to **Application** tab > **Workspaces** > **Maintenance Units** > **Components** > **SQL Table Editor**.
2. In the **SQL Table Editor** section settings, set **Filter Mode** to either **Popup** (default) or **Filter Row**.

> **NOTE:** Consolidated column filtering is not supported on the following data types. Applying filters on these datatypes will not yield consolidated filtered results. Unsupported data types: binary, char, date, datetime2, datetimeoffset, float, geography, geometry, hierarchyid, image, filestream, money, nchar, ntext, numeric, nvarchar, real, rowversion, smalldatetime, smallmoney, sql_variant, text, time, timestamp, tinyint, varbinary, xml.

Filtering Columns Based on Values from Another Column

You can base a column's header filter on values from another column. This includes any column that is in the data set, even if the column is hidden. This is useful for name-value pair parameter columns as long as the values and names are pulled from individual columns.

To set up and configure the SQL Table Editor component:

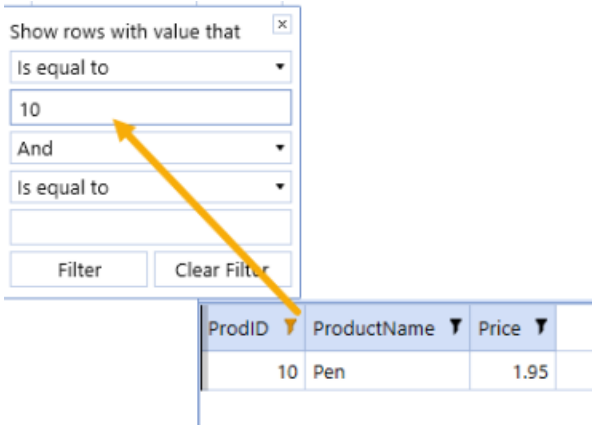
1. Inside the appropriate workspace, create and configure an SQL Table Editor component according to your needs.
2. Identify the column that you want to filter along with the column the filter is based upon.
3. In the SQL Table Editor component, configure the column format for the column you want to filter by typing the name of the column into the **FilterByColumnName** property.

Example

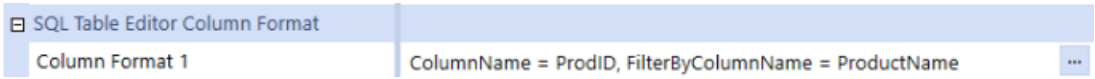
Begin with an SQL Table Editor grid. The data set can come from an SQL view or an SQL table.

ProdID	ProductName	Price
10	Pen	1.95
20	Mug	5.99
30	Tshirt	12.00
40	Speaker	35.99

When filtering data using a column header filter, the default behavior is to filter by the value stored in that column. For example, filtering the **ProdID** column on a value of 10 returns one row where **ProdID** equals 10.



If you would like to change the **ProdID** column filter and base it on **ProductName** instead, configure the column format for **ProdID** by setting **FilterByColumnName** to **ProductName**.



Column filtering for column **ProdID** is now based on values from the **ProductName** column.

SQL Table Editor Modify Table Data Using a View

You can modify data of an underlying base table through an SQL View. An SQL View is a virtual table where column and row contents are defined by an SQL query. The SQL Table Editor displays data from the virtual table, but you can change the data of the underlying base table. This provides more flexibility and control over how you present and manipulate data in the SQL Table Editor.

Use the following rules when you want to modify data through a View:

- Use Views to create a view of the data in one or more tables in the database. You can add calculated columns to your View.
- You can only update data in a single underlying base table. For example, if you create a View that combines data from four tables, you can only update data from one of those tables.
- Depending on what you are trying to do, you need the correct permissions to the underlying base table.

Configuration

Set up and configure the SQL Table Editor component:

1. Inside the appropriate workspace, create and configure an SQL Table Editor component according to your needs.
2. In the **Table Name** property, specify the underlying base table that contains the editable data.
3. In the **View Name** property, type a name for the view.
4. Identify columns in the view that do not exist in the underlying table. These columns are excluded from the Insert and Update statements that the SQL Table Editor generates when updating the base table.

5. Configure a column format for each column in the view that does not exist in the base table.
6. Set the **IsFromTable** property to False.

Example

You begin with two tables. The first is a Products table.

ProdID	ProductName	Price
10	Pen	1.95
20	Mug	5.99
30	Tshirt	12.00
40	Speaker	35.99

The second is an Orders table.

OrderID	prodID	Quantity
7800	10	5
7800	30	1
7800	40	2
9991	20	4

The two tables are linked by the **ProdID** value, as shown in the previous images.

Next, create a simple view called **OrderDetails** to show products specific to each order.

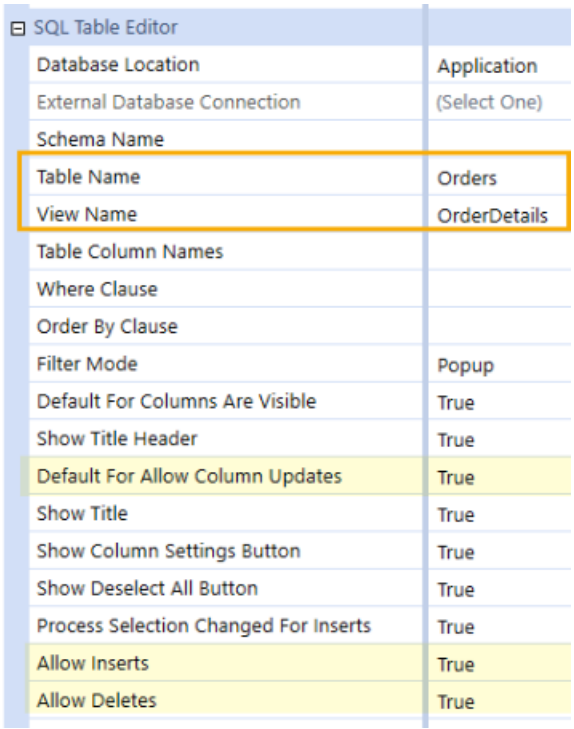
```
SELECT dbo.Orders.OrderID, dbo.Orders.prodID, dbo.Orders.Quantity, dbo.Products.productName,
    dbo.Products.Price
FROM dbo.Orders
INNER JOIN dbo.Products
ON dbo.Orders.prodID = dbo.Products.prodID
```

This produces the following view:

OrderID	prodID	Quantity	productName	Price
7800	10	5	Pen	1.95
7800	30	15	Tshirt	12.00
7800	40	2	Speaker	35.99
9991	20	4	Mug	5.99

At this point, you might want to expose the view to users so that they can update order quantities or add additional orders. This data is stored in the underlying base table, Orders.

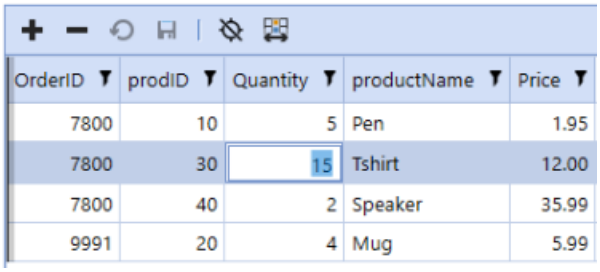
In the SQL Table Editor, you can set Orders as the **Table Name** (the base table) and OrderDetails as the **View Name**.



You now need to identify columns in the view that do not exist in the underlying base table. In this example, that would be productName and Price. For both columns, configure a column format where **IsFromTable** is set to False. You can also make the columns read-only by setting **AllowUpdates** to False.

SQL Table Editor Column Format	
Column Format 1	ColumnName = productName, IsFromTable = False, AllowUpdates = False
Column Format 2	ColumnName = Price, IsFromTable = False, AllowUpdates = False

You can now see all columns of the view while also modifying orders.

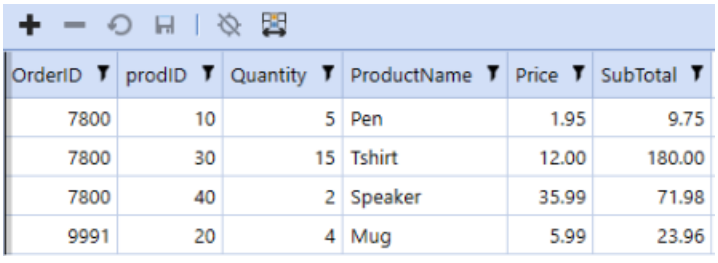


OrderID	prodID	Quantity	productName	Price
7800	10	5	Pen	1.95
7800	30	15	Tshirt	12.00
7800	40	2	Speaker	35.99
9991	20	4	Mug	5.99

You can also add calculated columns to views. These columns are virtual and do not exist in the database They derive their values dynamically when the view is executed.

For example, perhaps we would like to add a column that multiplies Quantity times Price. You could update the view with an expression. You can give the expression any name, such as SubTotal in the following example.

```
SELECT dbo.Orders.OrderID, dbo.Orders.prodID, dbo.Orders.Quantity, dbo.Products.productName,
    dbo.Products.Price,
(dbo.Orders.Quantity*dbo.Products.Price) AS SubTotal
FROM dbo.Orders
INNER JOIN dbo.Products
ON dbo.Orders.prodID = dbo.Products.prodID
```



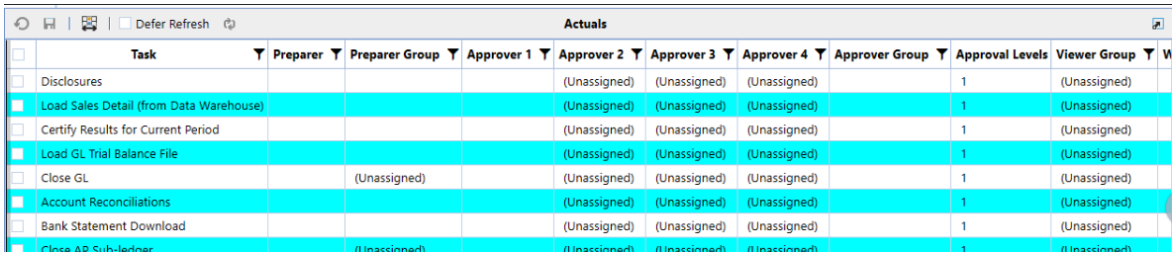
OrderID	prodID	Quantity	ProductName	Price	SubTotal
7800	10	5	Pen	1.95	9.75
7800	30	15	Tshirt	12.00	180.00
7800	40	2	Speaker	35.99	71.98
9991	20	4	Mug	5.99	23.96

SQL Table Editor Row Background Colors

You can apply a background color to all rows or alternate rows using the BackgroundColor property and the AlternateRowBackgroundColor property. This makes it easier to distinguish between rows of data at a glance, and can significantly reduce eye strain.

To configure these properties:

1. Navigate to **Application** tab > **Workspaces**, and expand the workspace.
2. Within that workspace, expand a maintenance unit, then expand **Components**.
3. Expand **SQL Table Editor** and select a component.
4. In the **Formatting** section, at the far right of the **Display Format** property, click the **Edit** (ellipses) button.
5. In the **Display Format** dialog box, in the **BackgroundColor** field, click the **down-arrow** and select a color.
6. In the **AlternateRowBackgroundColor** field, click the **down-arrow** and select a color.
7. Click **OK**.



Text Box

This is used to display a text box. You can also add formatted rich text to embedded refreshable content. This content can be used for narrative and can be edited and published into a report.

- Rich Text Properties
- Special Formatting Changes
- Insert Rich Text
- Modern Browser Experience with Rich Text

Rich Text Properties

Instead of making changes outside of OneStream or then having to continuously copy and paste into the application, you can add formatted text using the Allow Rich Text property in the Text Box component. This allows you to create formatted text using the formatting toolbar in OneStream in the Text Box Dashboard component.

Allow Rich Text

This property allows you to format font, color, alignment and more using rich text. The property is located in the Text Box component properties and has a default value of False.

- When set to False, this defaults to plain text and the component will not contain formatting options.
- When set to True, this allows you to add rich text formatting to your content.

> **NOTE:** When set to True, the Multiline field will not be enabled if Allow Rich Text is set to True.

Enable Spell Check

When Allow Rich Text is set to True, this property displays the spell check options in the toolbar. When Allow Rich Text is set to False, this property is disabled. As an Implementer in the Windows Application, you can set the Enable Spell Check field in the text box component to True or False. This property is under the Text Box Component > Component Properties > Text Box section.

> **NOTE:** This field property cannot be updated through Parameters or Business Rules.

When Enable Spell Check is set to True, these spell check icons are available in the Text Box ribbon:

- **Proofing**: Spelling and Language
- **Spell Check**: English (United States)

The default for Spell Check is English. Spell Check will only work when English is selected in Windows for International users. When set to Not Used, misspelled words will not display red lines underneath the text.

> **IMPORTANT:** Existing Text Box components will have Spell Check disabled and the default will be set to False.

IsReadOnly

As an Implementer in the Windows Application, you can set the rich text functionality in the text box component to be read-only or editable for select users through the **IsReadOnly** property.

This property is located in the Text Box Dashboard Component > Component Properties > Display Format field's Edit button > Display Format dialog > General section > Position section. You can also type "IsReadOnly=" to equal either True or False in the Display Format field. The default value is (Use Default). This property affects existing text boxes and rich text boxes.

- When set to **True**, you cannot edit the Text Box content and this property is read-only.
- When set to **False** or (**Use Default**), you can edit the Text Box content.

Set IsReadOnly Property

As an Implementer in the Windows Application, you can use a Business Rule or Custom Parameters to set the IsReadOnly parameter to True or False.

Here are a few scenarios to set this property:

Scenario 1

You can set a Dashboard String Business Rule in a Workspace Assembly with the IsReadOnly property. You can set the IsReadOnly property using Workspace Assemblies by creating a Dashboard String Function Business Rule which returns "IsReadOnly = True" or "IsReadOnly = False". This will not be automatically added to the Assembly File code.

For example, in the Rich Text Component > Display Format field, type the following string: XFBR (Workspace.Current.MyAssembly.MyStrings, IsReadOnly).

In the Workspace Assembly > Assembly File service, update the XFBR string to return either "IsReadOnly = True" or "IsReadOnly = False"" inside the Assembly File service. Refresh your dashboard after updating the XFBR string to display the updated state. If IsReadOnly is set to False, users can edit.

Scenario 2

You can reference a Custom Parameter in the Text Box Component > Display Format field. For example, if the Custom Parameter was named "TextBoxReadOnly", enter "|!TextBoxReadOnly!|" in the Text Box Component > Display Format field. Then in the Custom Parameter, write "IsReadOnly = True". If the Text Box Component > Display Format does not reference IsReadOnly, but the Display Format field references the Custom Parameter, which contains IsReadOnly = True, the Dashboard Text Box would be IsReadOnly and users will not be able to edit.

Scenario 3

You can set a Dashboard XFBR String Business Rule with the IsReadOnly property. You can set the IsReadOnly property using Business Rules by creating a Dashboard XFBR String Function Business Rule which returns "IsReadOnly = True" or "IsReadOnly = False". This will not be automatically added to the args line of code.

In a Dashboard XFBR String Business Rule, update the XFBR string to return either "IsReadOnly = True" or "IsReadOnly = False" inside the args lines of code. Refresh your dashboard after updating the XFBR string to display the updated state. If IsReadOnly is set to False, users can edit.

Special Formatting Changes

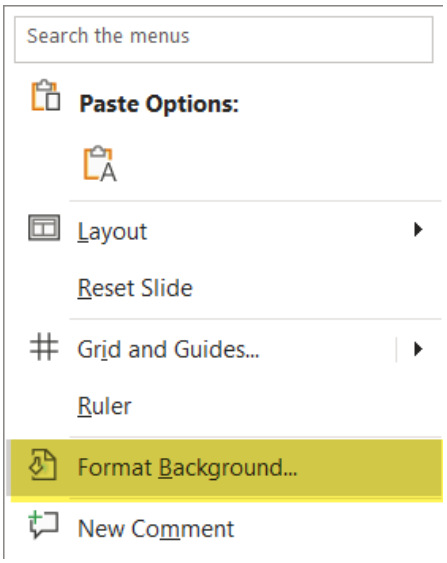
Background Color

You can include the optional **BackgroundColor** setting to specify the background color of the Rich Text image. If the BackgroundColor setting is blank or not valid, it will default to “#FFFFFF” (white). The value for this setting must be in Hexadecimal format (#RRGGBB or #AARRGGBB) and can be prefixed with or without #.

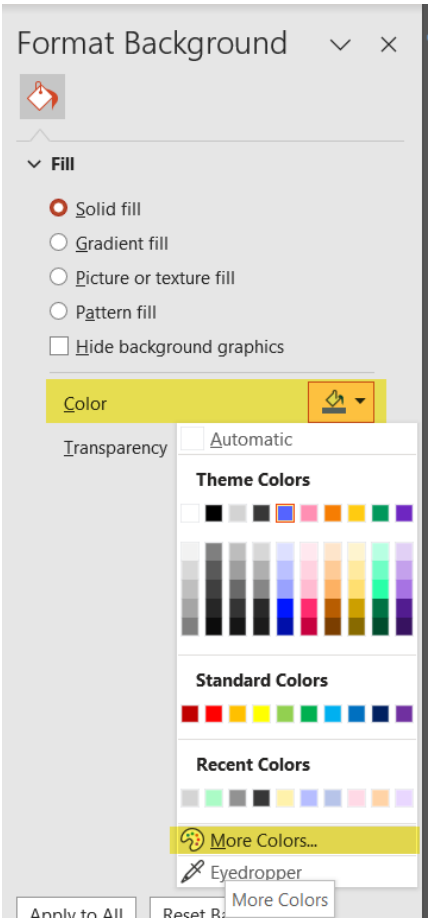
> **NOTE:** The background cannot be transparent. If transparency values are included in the BackgroundColor value, such as "AA" IN #AARRGGBB, it will be ignored and treated as #RRGGBB (no transparency) and remain opaque.

When using Microsoft PowerPoint to insert your rich text, you can find a Hex code on the slide by following these steps:

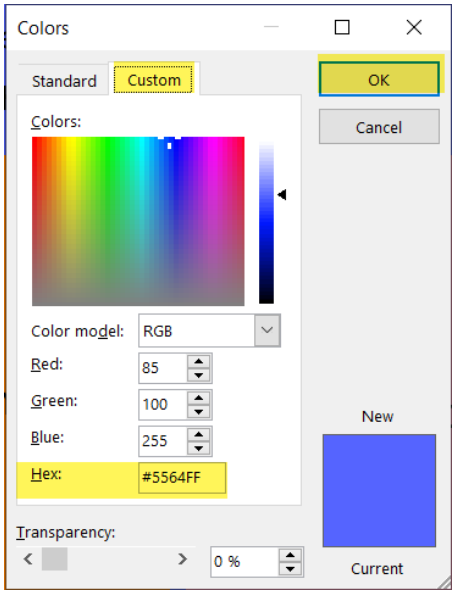
1. Right-click on the slide and click **Format Background**.



2. Next to **Color**, select the **Fill Color (paint icon)** and click **More Colors**.



3. Click the **Custom** tab. The Hex code is located under **Hex**. Enter the code and then click **OK**. The hexadecimal code will be listed in the Hex field.



Measurement Unit

You can include the optional **MeasurementUnit** setting to specify the measurement unit used by the PageWidth and PageHeight settings. If the MeasurementUnit setting is blank or not valid, it will default to Inch.

The following values are allowed for MeasurementUnit:

- **Document**: A document unit corresponds to 1/300th of an inch.
- **Centimeter**
- **Inch**
- **Millimeter**
- **Point**: A point unit corresponds to 1/72nd of an inch.

Page Settings

You can include the optional PageWidth, PageHeight, and PageNumber settings.

- **PageWidth**: This affects the width of the image that is generated and wraps the text to fit within the page width. If the PageWidth setting is blank or not valid, it will default to 8.5 inches.
- **PageHeight**: This affects the height of the image that is generated. If the PageHeight setting is blank or not valid, it will default to 11 inches.
- **PageNumber**: If the PageNumber setting is blank or not valid, it defaults to the first page.

Borders

You can include the optional **IncludeBorders** setting to display borders in the resulting image. If the IncludeBorders setting is blank or not valid, it will default to False.

Insert Rich Text

You can add formatted rich text to embed refreshable content in a document that can be published in various areas.

> **IMPORTANT:** Copy all syntax needed to insert rich text using Document Variable arguments directly from the Object Lookup dialog box. See Extensible Document Settings.

Microsoft PowerPoint

1. Navigate to the **Object Lookup** dialog box in OneStream. This dialog box provides all the syntax needed to insert content into an extensible document. This icon can be found on the following pages under the **Application** tab:
    - Form Templates
    - Books
    - Cube Views

- Workspaces
- Data Management

2. In the dialog box, select **Extensible Document Settings**. Expand **Insert Content Using Office Image** and then expand **Rich Text**. Select the line of syntax, and click the **Copy to Clipboard** button to copy the first string.
3. In Microsoft PowerPoint, open a new or existing presentation, and insert an image.
4. Right-click on the image, and select the **View Alt Text** option.
5. In the **Alt Text** field, paste the string. The string `{XF}{Application}{RichText}{RichTextContent}` displays in the field. You must update the "RichTextContent" placeholder text to a specific rich text format or a parameter containing rich text to appear.

   Optional: Return to the **Object Lookup** > **Insert Content Using Office Image** > Rich Text node and then copy the second string. The second string includes additional formatting options, see Rich Text in Insert Content Using Office Image.
6. Enter your special formatting changes to use with the Rich Text. See Special Formatting Changes and Extensible Document Settings.
7. Select **File and Save As**. Select a file name, and add **.xfdoc** extension to the name. The file name is appended with ".xfdoc.pptx".
8. Click **Save** the save the file.
9. Navigate back to OneStream, and open **File Explorer**. Select your folder of choice and then upload the extensible file.
10. Right-click on the file and select **"Process and Open"**. You may need to enter or select a parameter and then click **OK**. Microsoft PowerPoint will display the Rich Text content in place of the original image.

Text Box Component

1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. Create a Workspace and Maintenance Unit or select an existing one. See Create a Workspace.
3. Navigate to Components, and click the **Create Dashboard Component** toolbar button.
4. In the **Create Dashboard Component** dialog box, click Text Box, then **OK**.

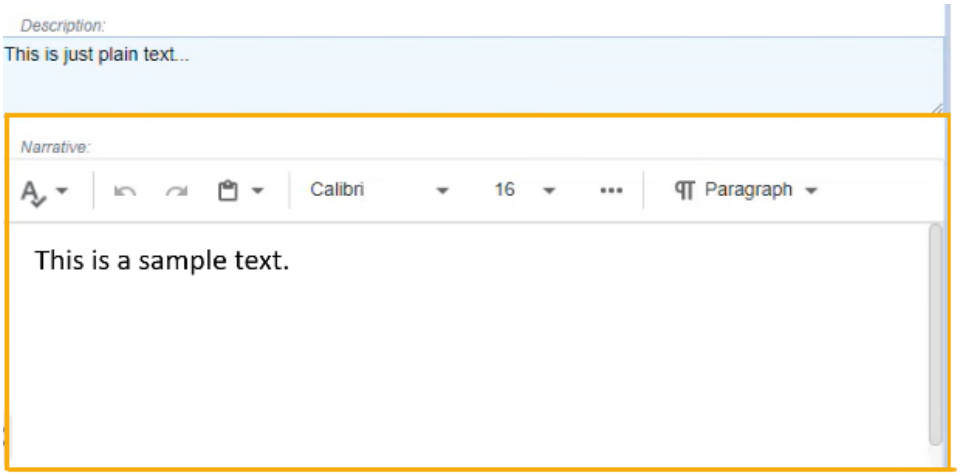
> **NOTE:** Under the Text Box section, you can set the Rich Text Properties. For example, you can set Allow Rich text to True. You can also set Enable Spell Check and IsReadOnly in this section. See Rich Text Properties.

5. Set your Rich Text Properties, and name the Text Box. Click **Save** to save the changes.
6. Add the Rich Text Box component to a new or existing dashboard within the same Workspace.
7. While the Dashboard node is selected, click the **View Dashboard** toolbar button to run your dashboard with the Rich Text Box component.
8. If IsReadOnly is set to (Use Default) or False, you can begin to edit the rich text box.

Modern Browser Experience with Rich Text

You can add formatted rich text to embedded refreshable content. This content can be used for narrative and can be edited and published into a report in the Modern Browser Experience. This works alongside the existing OneStream Windows Application to give users more options to use rich text formatting options across a wide range of devices.

You can add formatted text using the Allow Rich Text property in the Text Box component within the OneStream Windows Application. This allows you to create formatted text using the formatting toolbar in the Text Box dashboard component in a Modern Browser Experience.



See Extensible Documents with Rich Text Overview. These properties must be set up in the Windows Application to be available in Modern Browser Experience when run through a Dashboard.

> **NOTE:** The Find and Replace formatting options are not available in Modern Browser Experience.

Supported Rich Text Functionality

Allow Rich Text

Set the Allow Rich Text field to True to add extra formatting options. This allows you to make special formatting changes using rich text, such as font, color, alignment, and more.

> **IMPORTANT:** You may need to alter the design of your dashboard properties or individual components to allow the text box component to display properly in your dashboard layout.

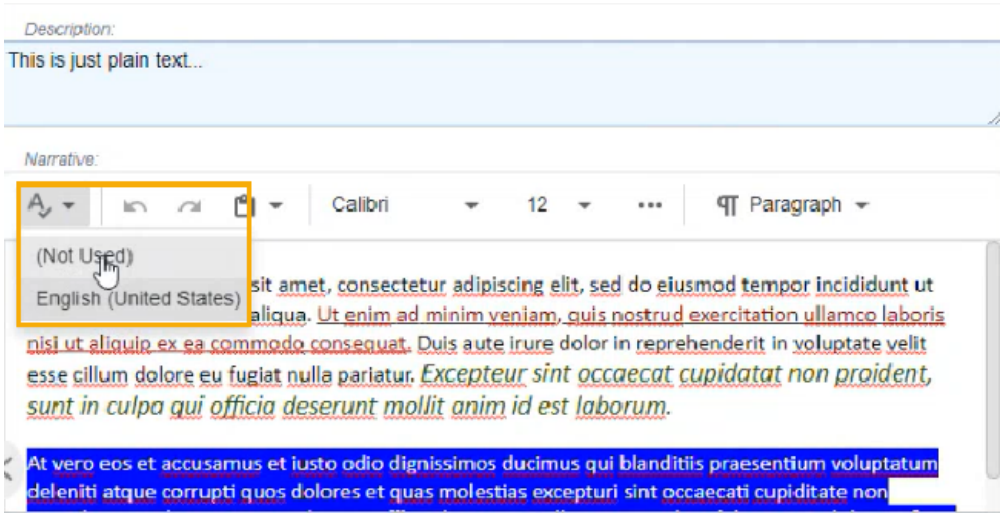
Spell Check

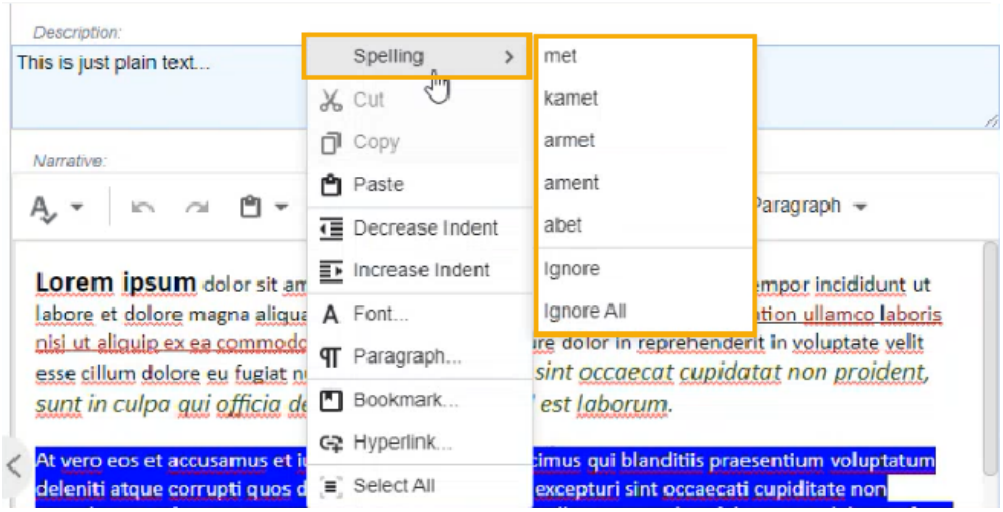
You can perform spell checks with rich text in the Text Box ribbon in Modern Browser Experience. When spell check is enabled, the Spell Check icon is available with the following options:

- (Not Used)
- English (United States)

The default for this property is Use English. English Spell Check is only available. If you select (Not Used), it will ignore any misspelled words and not include any red underlining. When spell check is enabled, you can right-click on a word in Modern Browser Experience and select different Spelling Check options.

> **NOTE:** The limit of word suggestions is seven.



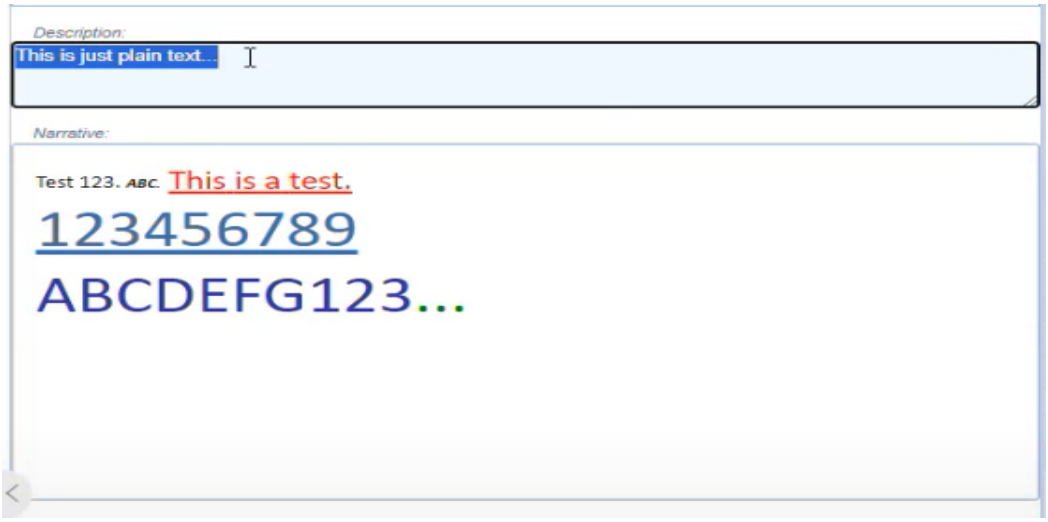


IsReadOnly

As an Implementer, you can enable the new rich text functionality within the text box component and make it read-only or editable for select users. Set and use the IsReadOnly property located in Text Box Component Display Format under Position to True to make rich text available for select users. You can set this property to the following options:

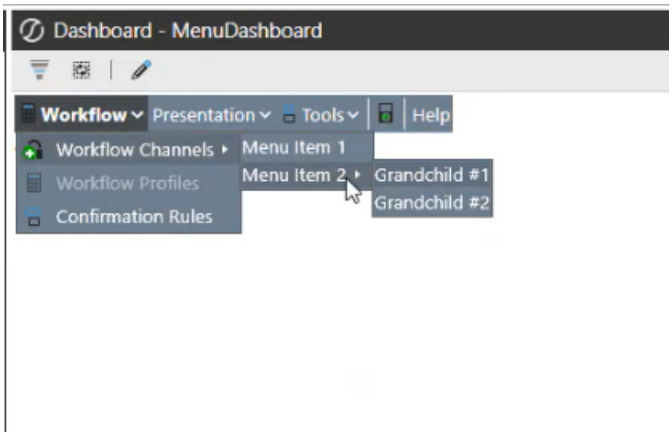
- When set to **True**, you cannot edit and this property is read-only.
- When set to **False** or (**Use Default**), you can edit.

> **NOTE:** In the following example, IsReadOnly is set to True and no formatting options are available in the rich text box.



Menu Component Overview

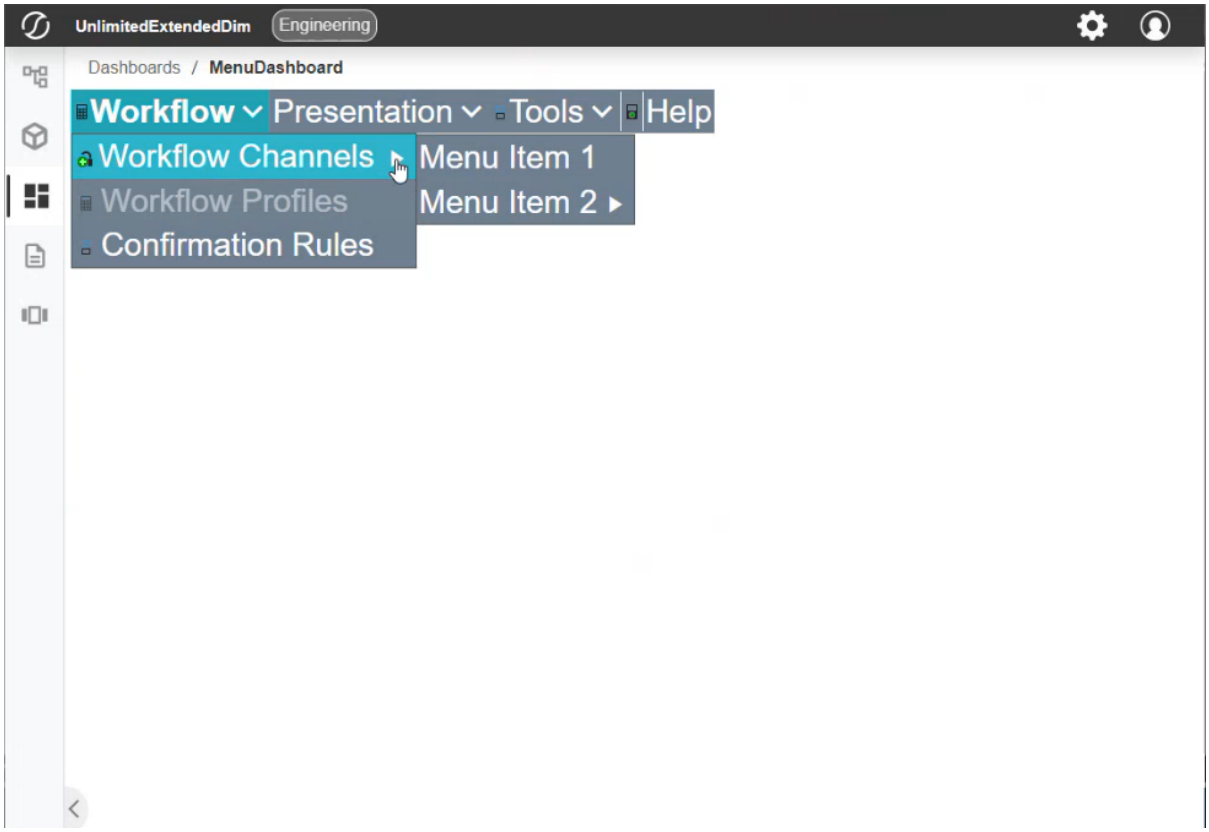
Developers can add navigational menus to their solutions. With the addition of navigational menus, users can easily navigate throughout the solutions and get to where they want to go. You can hover over or select a menu item to view a drop-down menu with nested options. Then, explore additional menu items by selecting those options to navigate seamlessly to your desired page.



- Modern Browser Experience and Menu Component
- Menu Component and Items
- Data Adapter and Menu Component
- Set Up Menu Component
- Customization
- Sample Customization

Modern Browser Experience and Menu Component

This component is created in the Windows Application and can be displayed in the Modern Browser Experience on a dashboard. The Modern Browser Experience works alongside the existing OneStream Windows Application to give users more options to display dashboards containing Menu Components across a wide range of devices. See Menu Component Overview.



Set Up and Customization

These properties must be set up and customized in the Windows Application to be available in the Modern Browser Experience when run through a dashboard. See Set Up Menu Component and Customization.

> **IMPORTANT:** For a dashboard to display in the Modern Browser Experience and on specific devices, the Dashboard Group containing the dashboard must be added to a Dashboard Profile that includes Web, Tablet, and/or Phone for its Client/Device Visibility property.

> **NOTE:** If the dashboard still isn't appearing in the Modern Browser Experience, verify that the Dashboard Profile's Visibility property is set to Always or OnePlace.

Menu Component and Items

The Menu Component has multiple options for Menu Items that you can integrate and use within your component. All Menu Items must be created and set up in the Windows Application to display in the Modern Browser Experience.

As the developer, you can set the Menu Items through the Workspace Assembly or Business Rule code. The Workspace Assembly or Business Rule code containing Menu Items can then be used to populate the Data Table within a Data Adapter. You can configure specific Display Format settings for the main Menu Component through the Display Format Properties dialog. A Custom Parameter can also be included in the Display Format Properties dialog to configure the main Menu Component.

See Menu Component and Items.

Consider the following when using Menu Component and Items in the Modern Browser Experience:

- **Menu Hover Color and Auto Open**: When using tablets, the Menu Hover Color will appear after tapping to select. To Auto Open, you must tap to open the Menu Item.

Data Adapter

Developers can write a Workspace Assembly or Business Rule to include data and content that a Data Adapter can use. The Data Adapter can then be used to populate the content of a Menu Component in the Windows Application. See Data Adapter and Menu Component.

Considerations

See Considerations for a list of all items to consider when using the Menu Component in the Modern Browser Experience and Windows Application.

Menu Component and Items

The Menu Component has the following options you can integrate and use within your navigational menu. As the developer, you can set the Menu Items through the Workspace Assembly or Business Rule code. The Workspace Assembly or Business Rule code containing Menu Items can then be used to populate the Data Table within a Data Adapter. You can configure specific Display Format settings for the main Menu Components through the Display Format Properties dialog. A Custom Parameter can also be included in the Display Format Properties dialog to configure the main Menu Component.

Menu Component Properties

Auto Open

Defaults to False. When Auto Open is set to False, you must click each Menu Component Item to expand its nested children. When Auto Open is set to True, the Menu Component Menu Items will automatically expand its nested children when you hover the cursor over the item.

Show Down Arrow

Defaults to False. When Auto Open is set to False, it will not display an arrow icon on the top-level Menu Items even if they contain nested children. When Show Down Arrow is set to True, it will display the arrow icon on the top-level Menu Items containing nested children.

Display Format Properties

General

- **Custom Parameters**: Use this property to use parameters from the following locations: Default Workspace and Non-Default Maintenance Unit or Non-Default Workspace and Maintenance Unit. You can enter the name of a Parameter that contains a string specifying Display Format settings into this property.

Position

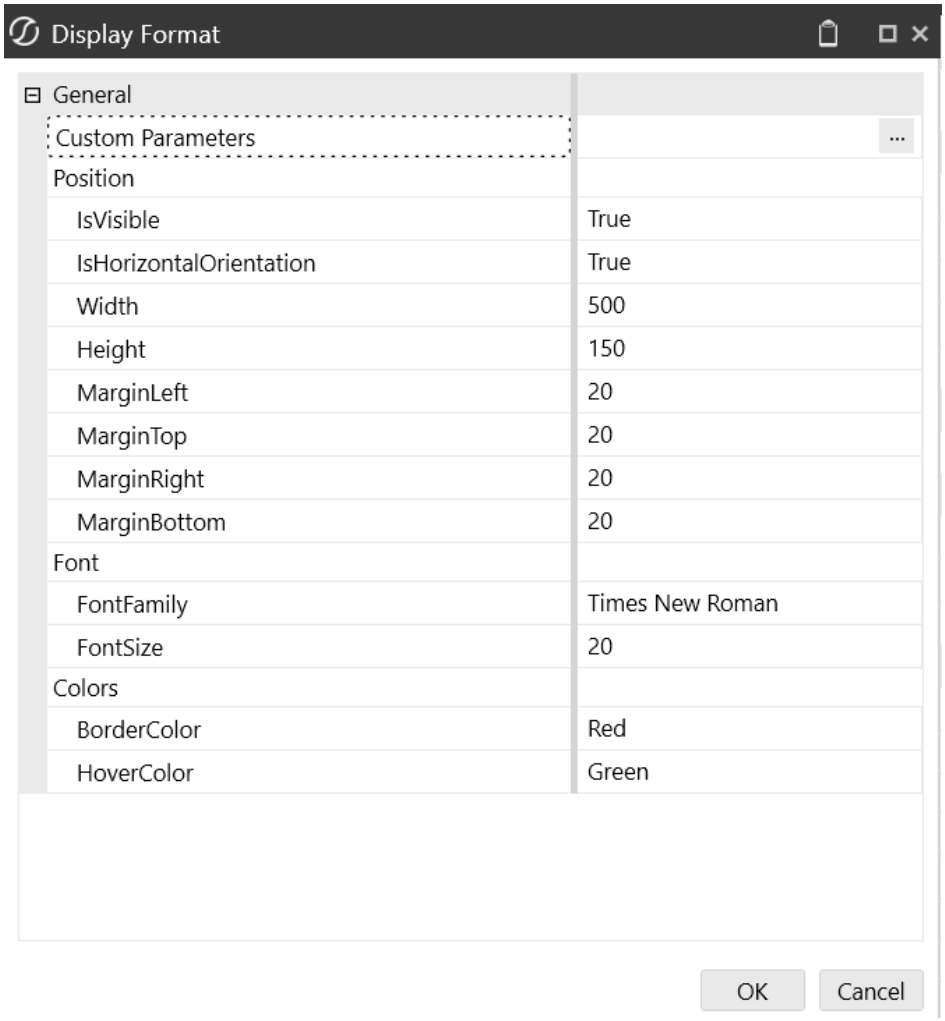
- **IsVisible**: Defaults to (Use Default). Select (Use Default), True, or False to determine if the component is visible on the dashboard.
- **IsHorizontalOrientation**: Defaults to (Use Default). Select (Use Default), True, or False.
- **Width**: Enter a width number.
- **Height**: Enter a height number.
- **MarginLeft**: Enter a margin left number.
- **MarginTop**: Enter a margin top number.
- **MarginRight**: Enter a margin right number.
- **MarginBottom**: Enter a margin bottom number.

Font

- **FontFamily**: Enter a font family name.
- **FontSize**: Enter a font size number.

Colors

- **BorderColor**: Defaults to (Use Default). Select a color, enter a custom color, or use the default value.
- **HoverColor**: Defaults to (Use Default). Select a color, enter a custom color, or use the default value.



Menu Items

Nested Menus

You can create nested menus with the Menu Component. A nested menu is a menu item with a submenu. This can be displayed at any nested level depth.

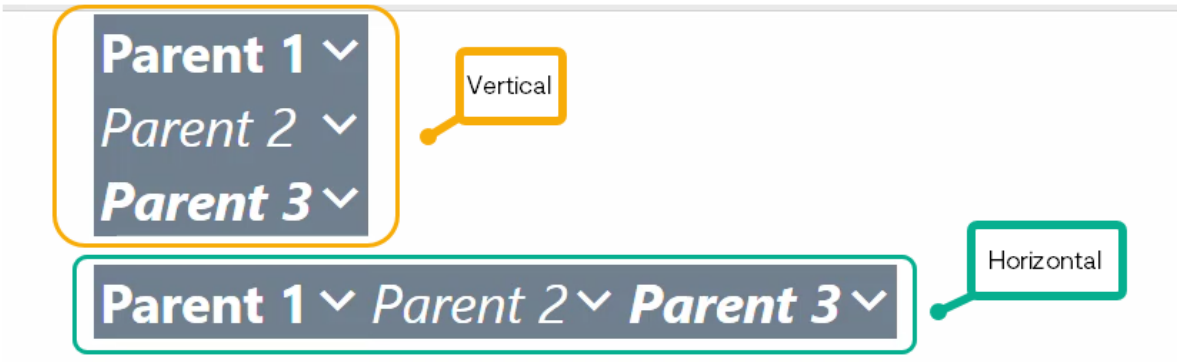
> **NOTE:** Parent menu items cannot have more than four nested child levels.

Enable or Disable Menu Items

You can set specific menu items in a Menu Component to Enabled or Disabled. When set to Disabled (bool isEnabled = false), you cannot click on or select that menu item. Both parent and child menu items can be set to Enabled or Disabled.

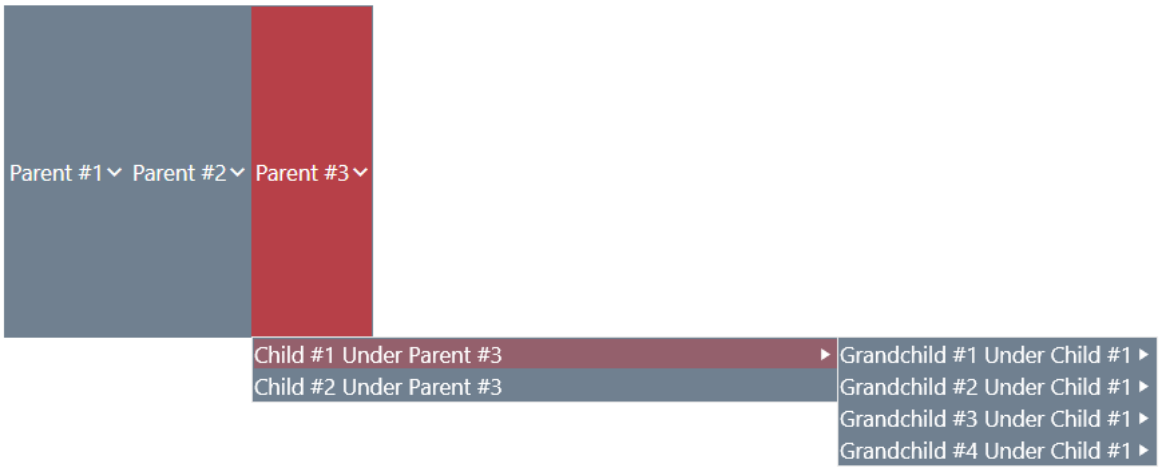
Vertical or Horizontal Menus

You can display menu components in a vertical or horizontal format. This property is configured in the Display Format Properties dialog.



Width and Height

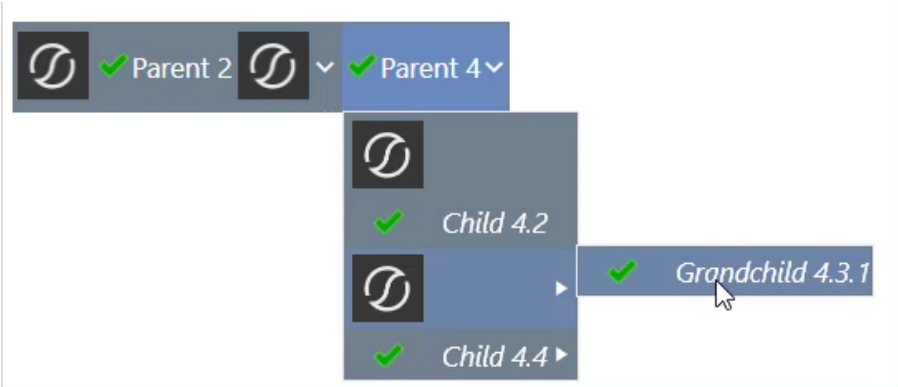
You can configure the overall Menu Component's width and height along with setting each individual menu item's width and height. The width or height of the overall Menu Component displayed on the dashboard are set in the Display Format Properties dialog. The width or height of individual Menu Items within the Menu Component are set through the Workspace Assembly or Business Rule code. Menu Item width and height can be specified at any nested level depth.



Images

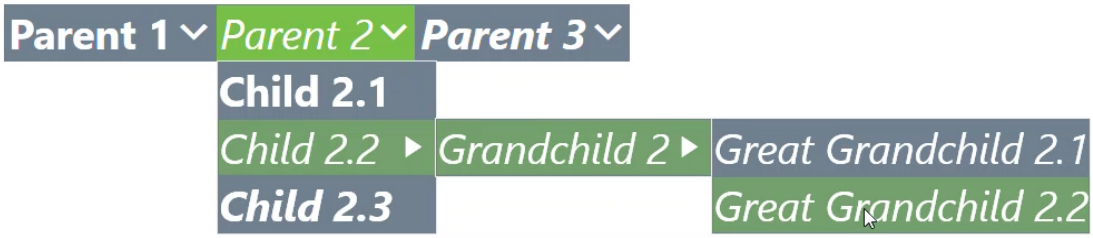
Images can be set to display within a Menu Component's parent and nested menu items. You can choose to display the image on its own within the menu item or along with header text or an arrow icon. If the menu item uses an image and the drop-down arrow, the image will always display to the left of the arrow.

> **NOTE:** The arrow icon will only display for parent menu items if "Show Down Arrow" is set to True for the Menu Component and they contain nested items. Nested items under the parent menu items, or children, are not affected by this setting and will always display an arrow if they contain their own nested items.



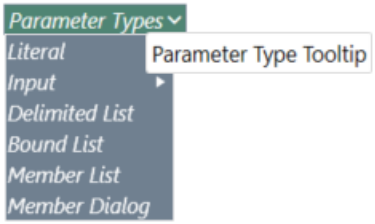
Bold and Italic Menu Options

You can set the menu item text to be bold and/or italic in the Workspace Assembly or Business Rule code.



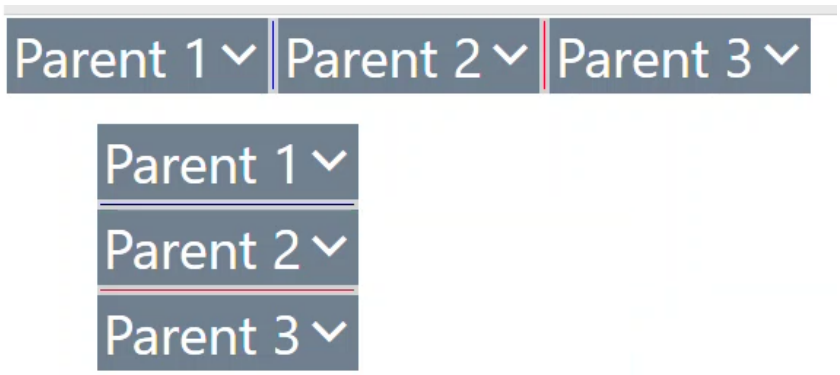
Tooltip

You can customize tooltips for the Menu Component's menu items. When the menu item has a set string for the tooltip, the text will display when you hover the cursor over that menu item.



Separators

You can set and create horizontal and vertical separators within the Menu Component. Separators are their own individual menu items set through Workspace Assembly or Business Rule code and can be listed anywhere in the Menu Component. When the isSeparator arguments is set to True on an XFMenuItem class, then it ignores most settings except for the color formatting options. If the Menu Component is horizontal, all Separator menu items created at the parent/root level will be vertical, and vice versa. Nested separators always display horizontally between child items.

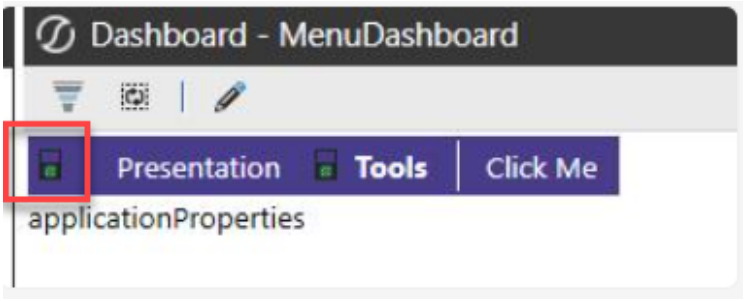


Text Color for Background and Text Color Styles

You can customize the text, background, and border colors. This can be displayed at any nested level depth.

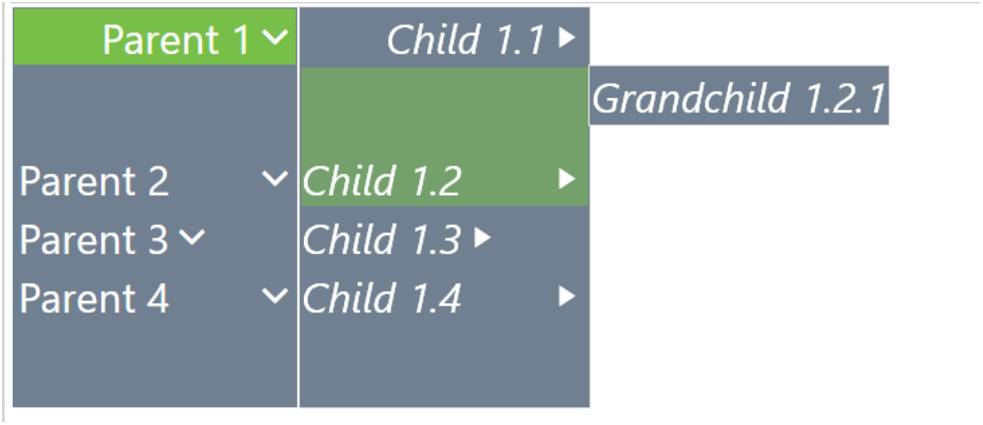
Toolbar Image Menu

You can also convert navigational menus into toolbars with image icons. You can omit header text and configure the width, height, and padding properties in the Workspace Assembly or Business Rule code to achieve this. You can also pull images from Dashboard Files, Client Images, and URLs.



Padding Options

You can add and set padding for each menu item. The padding settings are set in the Workspace Assembly or Business Rule code (PaddingLeft, PaddingTop, PaddingRight, and PaddingBottom) and can be set for each individual menu item.

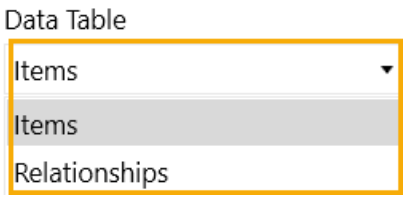


Data Adapter and Menu Component

Developers can create a Data Set using a Workspace Assembly or Business Rule to populate a Data Adapter that's used by Menu Component Items.

Data Table

The Data Table within a Data Adapter displays the Items and Relationships in your Menu Component.



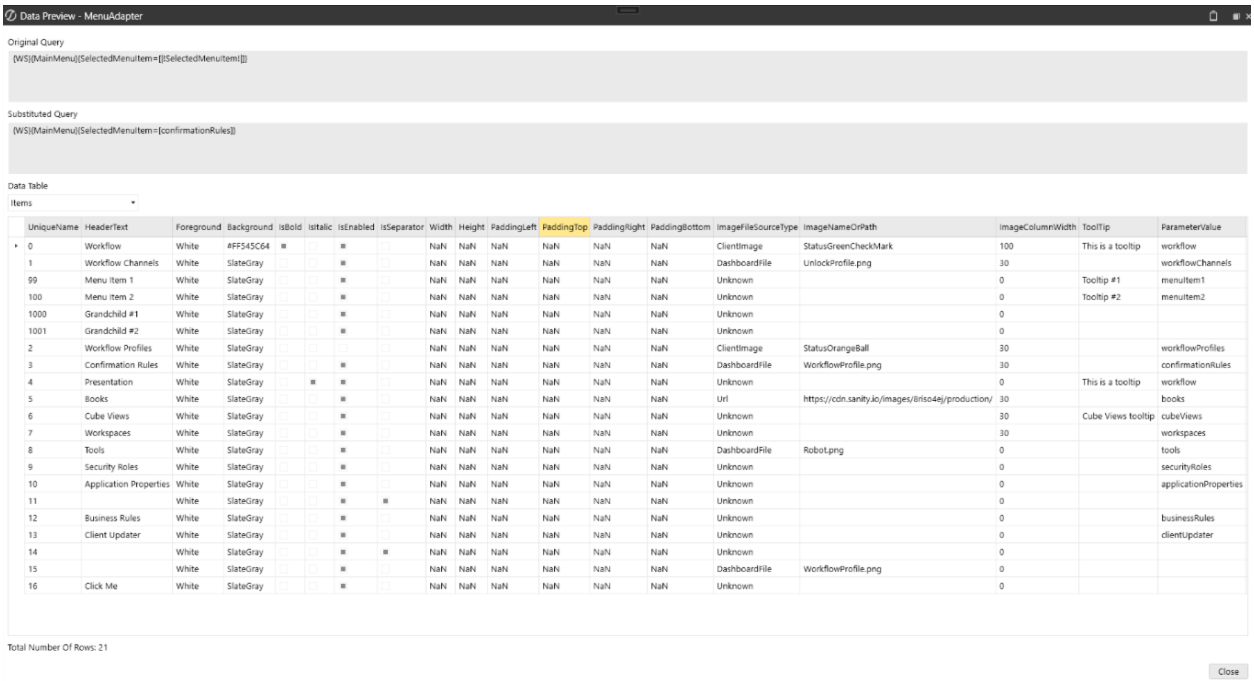
Items

The Menu Component Data Adapter contains the following fields and displays them in a table format:

Field	Description
UniqueName	Unique Id
HeaderText	Menu Item Name (text that displays)
Foreground	Foreground color of Menu Item (this displays the text color)
Background	Background color of Menu Item (this displays the background color behind the text)
IsBold	Determines if HeaderText is Bolded (True or False)

IsItalic	Determines if HeaderText is Italicized (True or False)
IsEnabled	Determines if the Menu Item is selectable (True or False) **NOTE:** IsEnabled is different from IsVisible. When IsEnabled is False, the Menu Item will still be visible but you cannot select the item. isVisible determines if the Menu Component is visible.
IsSeparator	Determines if the Menu Item is a Separator (True or False)
Width	Width of menu item label
Height	Height of menu item label
PaddingLeft	Amount of padding added between the HeaderText and left border of the menu item
PaddingTop	Amount of padding added between the HeaderText and top border of the menu item
PaddingRight	Amount of padding added between the HeaderText and right border of the menu item
PaddingBottom	Amount of padding added between the HeaderText and bottom border of the menu item
ImageFileSourceType	Source of the Image such as Dashboard, File Explorer, and more

ImageNameOrPath	Name of Image
ImageColumnWidth	Amount of width added to the column of the image
Tooltip	Displays text when hovering over the menu item
ParameterValue	Parameter value of the menu item



UniqueName	HeaderText	Foreground	Background	IsBold	IsItalic	IsEnabled	IsSeparator	Width	Height	PaddingLeft	PaddingTop	PaddingRight	PaddingBottom	ImageFileSourceType	ImageNameOrPath	ImageColumnWidth	ToolTip	ParameterValue
0	Workflow	White	#FF545C64	■		■		NaN	NaN	NaN	NaN	NaN	NaN	ClientImage	StatusGreenCheckMark	100	This is a tooltip	workflow
1	Workflow Channels	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	DashboardFile	UnlockProfile.png	30		workflowChannels
99	Menu Item 1	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0	Tooltip #1	menuItem1
100	Menu Item 2	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0	Tooltip #2	menuItem2
1000	Grandchild #1	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		
1001	Grandchild #2	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		
2	Workflow Profiles	White	SlateGray					NaN	NaN	NaN	NaN	NaN	NaN	ClientImage	StatusOrangeBall	30		workflowProfiles
3	Confirmation Rules	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	DashboardFile	WorkflowProfile.png	30		confirmationRules
4	Presentation	White	SlateGray		■	■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0	This is a tooltip	workflow
5	Books	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Url	https://cdn.sanity.io/images/8riso4ej/production/	30		books
6	Cube Views	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		30	Cube Views tooltip	cubeViews
7	Workspaces	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		30		workspaces
8	Tools	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	DashboardFile	Robot.png	0		tools
9	Security Roles	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		securityRoles
10	Application Properties	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		applicationProperties
11		White	SlateGray			■	■	NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		
12	Business Rules	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		businessRules
13	Client Updater	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		clientUpdater
14		White	SlateGray			■	■	NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		
15		White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	DashboardFile	WorkflowProfile.png	0		
16	Click Me	White	SlateGray			■		NaN	NaN	NaN	NaN	NaN	NaN	Unknown		0		

Relationships

- **ParentName**: This is the name of the parent menu item of a menu item.

  > **NOTE:** Top-Level Menu Items will be blank or display Empty for the ParentName field.

- **ChildName**: This is the name of the child menu item.

- **SortOrder**: This number specifies the numerical order of the menu item in the parent-child relationship.

> **NOTE:** The first menu item in the parent-child relationship will have a SortOrder value of 0.

Data Table

Relationships

ParentName	ChildName	SortOrder
1	1.2	1
1.2	1.2.1	0
1.2	1.2.2	1
1.2	1.2.3	2
1.2	1.2.4	3
	2	1
2	2.1	0
2	2.2	1
2	2.3	2
2	2.4	3
2	2.5	4
	3	2
3	3.1	0
3.1	3.1.1	0
3.1.1	3.1.1.1	0

Total Number Of Rows: 37

To view the data of an existing Data Adapter, follow these steps:

1. Expand the **Workspace** > **Maintenance Unit** > **Data Adapter** node, and then select an existing Data Adapter.
2. Click the **Test Data Adapter** toolbar button.
3. If a Parameter dialog box appears, input valid arguments for the Parameters, then click the **OK** button.

Set Up Menu Component

To use the full capacity of the Menu Component, you must have the following items created and setup before customization:

- Workspace and Maintenance Unit
- Dashboard Group and Dashboard
- Menu Component
- Data Adapter
- Data Set Service or Dashboard Data Set created through a Workspace Assembly or Business Rule

> **IMPORTANT:** You can create and set up the following items in any order. You do not need to follow this order.

> **NOTE:** You can also use a Parameter if it is part of your implementation.

Create a Menu Component

1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. Create a Workspace and Maintenance Unit, or select an existing one. If you are selecting an existing Workspaces, ensure it is a Default Workspace > Non-Default Maintenance Unit or any Non-Default Workspace > Maintenance Unit. See Create a Workspace.

3. Create a Dashboard Group and Dashboard, or select an existing one. See Dashboard.
4. Navigate to and select the Components node, and click the **Create Dashboard Component** toolbar button.
5. In the **Create Dashboard Component** dialog box, click **Menu**, then **OK**.
6. Name the Menu component.
7. Click the **Save** button to save the changes.

Create a Data Adapter and Parameter

1. In the Maintenance Unit, navigate to and select the **Data Adapters** node, and click the **Create Data Adapter** toolbar button.
2. Name the Data Adapter.
3. Click the **Save**button to save the changes.
4. (Optional) Then, create a Parameter. See Create a Parameter.

   > **NOTE:** Parameters are not a necessity, but can be used in certain Command Type queries.

5. (Optional) After creating a Parameter, utilize it as needed within the Data Adapter's Data Source properties.

Create a Workspace Assembly

1. In the Maintenance Unit, navigate to and select the **Assemblies** node, and click the **Create Assembly** toolbar button.
2. Create an Assembly. When naming the Assembly, ensure naming conventions match with what is present in the Workspace or Maintenance Unit's Workspace Assembly Service field.

> **Example:** If the Workspace contains an Assembly named "MenuAssembly", you will need to specify "MenuAssembly.WsasFactory" in the Workspace Assembly Service field.

3. Create an Assembly file using the Service Factory source code type.

> **NOTE:** You will choose the Service Factory and Data Set Service source code types to create two different Assembly files to use with the Menu Component.

   a. Create a new Assembly File for your Menu Component. In the Assemblies Files, right-click on **Files** and select **Add File**.

      In the Add File dialog box:

      a. Type **WsasFactory** as the File Name.

      b. Set your Source Code Type to **Service Factory**.

      c. Set your Compiler Action to **(Use Default)**.

      d. Select **OK**.

4. Create the second Assembly File. In the Assemblies Files, right-click on **Files** and select **Add File**.

   a. In the Add File dialog box:

      a. Type **WsasDataSet** as the File Name.

      b. Set your Source Code Type to **Data Set Service**.

      c. Set your Compiler Action to **(Use Default)**.

      d. Select **OK**.

5. Select the **WsasFactory.cs (Service Factory)** Assembly File and uncomment the lines of code that reference the previously created WsasDataSet - Data Set Service Assembly File.

   > **IMPORTANT:** The Service Factory - Assembly File must contain uncommented code to function successfully: **case WsAssemblyServiceType.DataSet:return new WsasDataSet();**

6. Select the **WsasDataSet.cs (Data Set)** Assembly File, and begin to customize and build out your Menu Component. See Customization and Menu Component and Items.
7. Click the **Save** toolbar button to save the changes.
8. Click the **Compile Assembly to check syntax** button to ensure the Workspace Assembly files compile successfully.

Customization

Developers can customize the Menu Component content through the Data Set Assembly File. Menu Components can contain nested menu items that will expand to display new menu items when hovering over or selecting them.

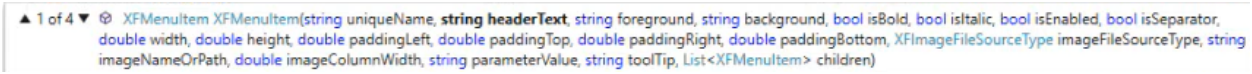
> **IMPORTANT:** A maximum of 250 menu items are supported.

> **CAUTION:** When attempting to open a dashboard with a Menu Component that contains more than 250 Menu Items, you will receive an error message: **Unable to create XFMenuItemCollection. A maximum of 250 menu items are supported**.

See Menu Component and Menu Items for a list of all options you can use and set up in the Workspace Assembly or Business Rule code.

The Data Set Assembly file returns a method call that contains the whole collection of Menu Items that builds your Menu Component. When building this out, you will notice that the code contains repetitive elements. When creating new menu items, it will auto-complete and display the default arguments including the Menu Item's unique name, header text, foreground color, background color, and so on.

```
XFMenuItem XFMenuItem(string uniqueName, string headerText, string foreground, string
background, bool isBold, bool isItalic, bool isEnabled, bool isSeparator,
double width, double height, double paddingLeft, double paddingTop, double paddingRight,
double paddingBottom, XFImageFileSourceType imageFileSourceType, string
imageNameOrPath, double imageColumnWidth, string parameterValue, string toolTip,
List<XFMenuItem> children)
```



Sample Customization

See Sample Customization for samples of fully customized Data Set Service Files of a custom Menu Component.

Connect Items to the Menu Component

After customizing and creating your Menu Component, you must update, link, and connect these items. The Workspace retrieves information through its Workspace Assembly Service property, which contains the path to the Assembly and Service Factory - Assembly File. For example, MenuAssembly.WsasFactory. The Service Factory File references the Data Set Assembly file which contains the data and content needed for the Data Adapter. The Data Adapter can then query this data and be connected to the Menu Component to generate the menu items' content on a Dashboard.

> **IMPORTANT:** You can connect any of the following items in any order. You do not need to follow this order.

Update the Data Adapter

1. In the Maintenance Unit, navigate to **Data Adapter**.
2. Update the **Data Source**:
    a. Update Command Type to **Method**.
    b. Update Method Type to **Business Rule**.
    c. Click on the Method Query ellipsis button, and update the Method Query to reflect the Workspace Menu Component item as it is being returned in the Assembly file or Business Rule. You can also do this through other Command Types, Method Types, or Queries.

        > **Example:** **{WS}{MainMenu}{SelectedMenuItem=[|!SelectedMenuItem!|]}**

3. Click the **Save** button to save the changes.

Update the Bound Parameter

> **NOTE:** (Optional) Follow these steps if you are using Bound Parameters as part of your implementation.

1. In the Maintenance Unit, navigate to **Parameters**.
2. Rename the Parameter to work with the path of the Data Adapter. Click the **Rename Selected Item** toolbar button.

    > **Example:** Rename Test Parameter to SelectedMenuItem.

3. Click the **Save** button to save the changes.

4. Add the specified Bound Parameter "SelectedMenuItem" to the Menu Component's Bound Parameter property.

Link the Data Adapter

1. In the Maintenance Unit, navigate to the **Menu Component**.
2. Click the **Data Adapters** tab.
3. Click the **Add Dashboard Component** toolbar button, and select the Data Adapter you created.
4. Click the **OK** button and then the **Save** button to save the changes.

Link the Menu Component

1. Navigate to a **Dashboard Group containing a Dashboard**, and select the dashboard to connect with the Menu Component.
2. Click the **Dashboard Components** tab.
3. Click the **Add Dashboard Component** toolbar button, and select the Menu Component you created.
4. Click the **OK** button and then the **Save** button to save the changes.

Update the Workspace Assembly Service

1. Navigate to **Workspaces**, and select the Workspace created.
2. Under the **Assemblies**, go to the **Workspace Assembly Service** property.
3. Enter the path to the Service Factory - Assembly File you created.

   > **Example:** MenuAssembly.WsasFactory

4. Click the **Save** button to save the changes.

Run the Dashboard

1. Navigate to **Dashboard Groups**, and select the dashboard linked with the Menu Component.
2. Click the **View Dashboard** toolbar button.
3. If the Parameter dialog box appears, enter a valid argument and then click the **OK** button.

> **NOTE:** The Menu Component displays on the dashboard.

Sample Customization

The following sample is a fully customized Data Set Service File in C# of a custom Menu Component.

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

namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
{
    public class WsasDataSet : IWsasDataSetV800
    {
        public object GetDataSet(SessionInfo si, BRGlobals brGlobals, DashboardWorkspace
workspace, DashboardDataSetArgs args)
        {
            try
            {
                if ((brGlobals != null) && (workspace != null) && (args != null))
                {
                    if (args.DataSetName.XFEqualsIgnoreCase("MainMenu"))
                    {
```

```
                        var menu = new XFMenuItemCollection();

                        // Top-level Menu Item #1
                        var topLevelMenuItemOne = new XFMenuItem("1", "Menu Item #1",
"White", "SlateGray", false, false, true, false, double.NaN, double.NaN, double.NaN,
double.NaN, double.NaN, double.NaN, XFImageFileSourceType.Unknown, null, 0, "menuItem1",
null, null);

                        topLevelMenuItemOne.Children = new List<XFMenuItem>();
                        topLevelMenuItemOne.Children.Add(new XFMenuItem("2", "Child #1",
"White", "SlateGray", false, false, true, false, double.NaN, double.NaN, double.NaN,
double.NaN, double.NaN, double.NaN, XFImageFileSourceType.Unknown, null, 0, "child1",
"Tooltip #1", null));
                        topLevelMenuItemOne.Children.Add(new XFMenuItem("3", "Child #2",
"White", "SlateGray", false, false, true, false, double.NaN, double.NaN, double.NaN,
double.NaN, double.NaN, double.NaN, XFImageFileSourceType.Unknown, null, 0, "child2",
"Tooltip #2",
                            new List<XFMenuItem>()
                            {
                                new XFMenuItem("4", "Grandchild #1", "White", "SlateGray",
false, false, true, false, double.NaN, double.NaN, double.NaN, double.NaN, double.NaN,
double.NaN, XFImageFileSourceType.Unknown, null, 0, null, null, null),
                                new XFMenuItem("5", "Grandchild #2", "White", "SlateGray",
false, false, true, false, double.NaN, double.NaN, double.NaN, double.NaN, double.NaN,
double.NaN, XFImageFileSourceType.Unknown, null, 0, null, null, null)
                            }
                        ));

                        // Top-level Menu Item #2
                        var topLevelMenuItemTwo = new XFMenuItem("6", "Menu Item #1",
"White", "SlateGray", false, false, true, false, double.NaN, double.NaN, double.NaN,
double.NaN, double.NaN, double.NaN, XFImageFileSourceType.Unknown, null, 0, "menuItem2",
null, null);

                        menu.MenuItems.Add(topLevelMenuItemOne);
                        menu.MenuItems.Add(topLevelMenuItemTwo);

                        return menu.CreateDataSet(si);
                    }
                }

                return null;
            }
            catch (Exception ex)
            {
                throw new XFException(si, ex);
            }
        }
    }
}
```

This is how this sample Menu Component displays in a dashboard:

The following sample is a fully customized Data Set Service File in C# of a custom Menu Component.

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

namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
{
    public class MenuSvc : IWsasDataSetV800
    {
        public object GetDataSet(SessionInfo si, BRGlobals brGlobals, DashboardWorkspace
workspace, DashboardDataSetArgs args)
        {
         try {
                if ((brGlobals != null) && (workspace != null) && (args != null)) {

                    // sample menu
                    if (args.DataSetName.XFEqualsIgnoreCase("MainMenu"))
                    {
                        // create the menu object
                        XFMenuItemCollection menu = new XFMenuItemCollection();

                        // create top-level items
                        XFMenuItem parentMenuItemOne = new XFMenuItem("1", "Parent 1",
                            "White", "SlateGray", false, false, true, false, null);
                        // create items for the second level
                        XFMenuItem childMenuItemOne = new XFMenuItem("1.1", "Child 1",
                            "Black", "White", true, true, true, false, null);
                        XFMenuItem childMenuItemTwo = new XFMenuItem("1.2", "Child 2",
```

```
                        "Black", "White", true, true, true, false, null);
                    // create items for the third level
                    XFMenuItem grandChildMenuItemOne = new XFMenuItem("1.1.1",
"Grandchild 1",
                        "White", "SlateGray", true, true, true, false, null);

                    // create the hierarchy by adding children to direct parents as a
List
                    childMenuItemOne.Children = new List<XFMenuItem>(){
                        grandChildMenuItemOne
                    };
                    parentMenuItemOne.Children = new List<XFMenuItem>(){
                        childMenuItemOne
                    };

                    // you can also manipulate the list once created
                    parentMenuItemOne.Children.Add(childMenuItemTwo);

                    // add top-level items to the menu object
                    menu.MenuItems.Add(parentMenuItemOne);

                    // generate the dataset and return it
                    return menu.CreateDataSet(si);
                }
            }

            return null;

        } catch (Exception ex) {
            throw new XFException(si, ex);
        }
    }
  }
}
```

This is how this sample Menu Component displays in a dashboard:

Embedded Components

Embedded Dashboard
When a Dashboard is created in the Maintenance Unit, an Embedded Dashboard is also created. These Embedded Dashboards are used for sharing across Dashboard Groups in the same Maintenance Unit.

Embedded Custom Control Dashboard

A parent dashboard can contain multiple instances of the same custom control. Independent input parameters are mapped to each instance while each embedded dashboard component listens for events from the custom control.

Instance Name
An optional name that uniquely differentiates this instance of an embedded custom control dashboard from other instances. This instance name is used when saving state information related to interactions with the dashboard when using multiple instances of the same custom control dashboard.

Input Parameter Values
A comma separated list of name-value pairs for the embedded custom control dashboard initial parameter values. For example: "Paraml=Value1, Param2=(Value2], Param3=|!AnotherParamName!|."

Event Listeners
Actions to take when a component inside the referenced embedded custom control dashboard uses its navigation action settings to forward a selection changed event to the parent dashboard.

Data Adapters

Data Adapters specify the kind of data used within a dashboard. Once the Data Adapter is configured and pointing to the appropriate data, attach it to a Dashboard component in order to display it on a dashboard.

General Properties

Name
The name of the Data Adapter.

Description
A quick description of the Data Adapter.

Maintenance Unit
The Maintenance Unit to which the Data Adapter belongs.

There are three command types:

Cube View

Choose a Cube View as the source of a Data Adapter. Additional options can be selected here to include supplemental information for the resulting tables. However, adding on to what is defaulted may have a slight impact on performance:

Cube View

This command type allows for a pre-configured Cube View to be the Data Source for a dashboard.

Click and begin typing the name of the Cube View in the blank field. As the first few letters are typed, the names are filtered making it easier to find and select the one desired. If the name is unknown, expand a Cube View Group and scroll through the list to select the correct one. Once the Cube View is selected, click CTRL and Double Click. This will enter the correct name into the appropriate field.

Data Table Per Cube View Row

At the creation of the Data Adapter, the default is set to True. When set to True, a Data Table is created for each row in the dashboard. This allows for conditional formatting per Cube View row. Set this to False in order to merge the rows into one table. This will omit any undefined Cube View row.

Include Title

At the creation of the Data Adapter, the default is set to False. When set to True, the title will be displayed from the Report section of the Cube View as the title for the dashboard. Settings are True or False.

Include Header Left Label 1-4

At the creation of the Data Adapter, the default is set to False. When set to True, the left header labels will be displayed from the report section of the Cube View for the dashboard.

Include Header Center Label 1-4

At the creation of the Data Adapter, the default is set to False. When set to True, the center header labels will be displayed from the report section of the Cube View for the dashboard.

Include Header Right Label 1-4

At creation of the Data Adapter, the default is set to False. When set to True, the right header labels will be displayed from the report section of the Cube View for the dashboard.

Include… POV

If set to True, the POV information for the Cube, Entity and all other Dimensions are included. Use these if the report or dashboard needs this information.

Include Member Details

If set to True, additional Member property details are included in the results.

Include Row Navigation Link

If set to True, this data Adapter will include a row navigation link from a Cube View.

Include HasData Status

Includes additional true/false data on whether the row of results contains data for filtering purposes. Settings are True or False.

Include … View Member Text

This determines whether different Data Attachment text is going to be part of the results. This is necessary for showing text in a Data Explorer object or for using a Waterfall Chart and wanting to optionally show comments. Settings are True or False.

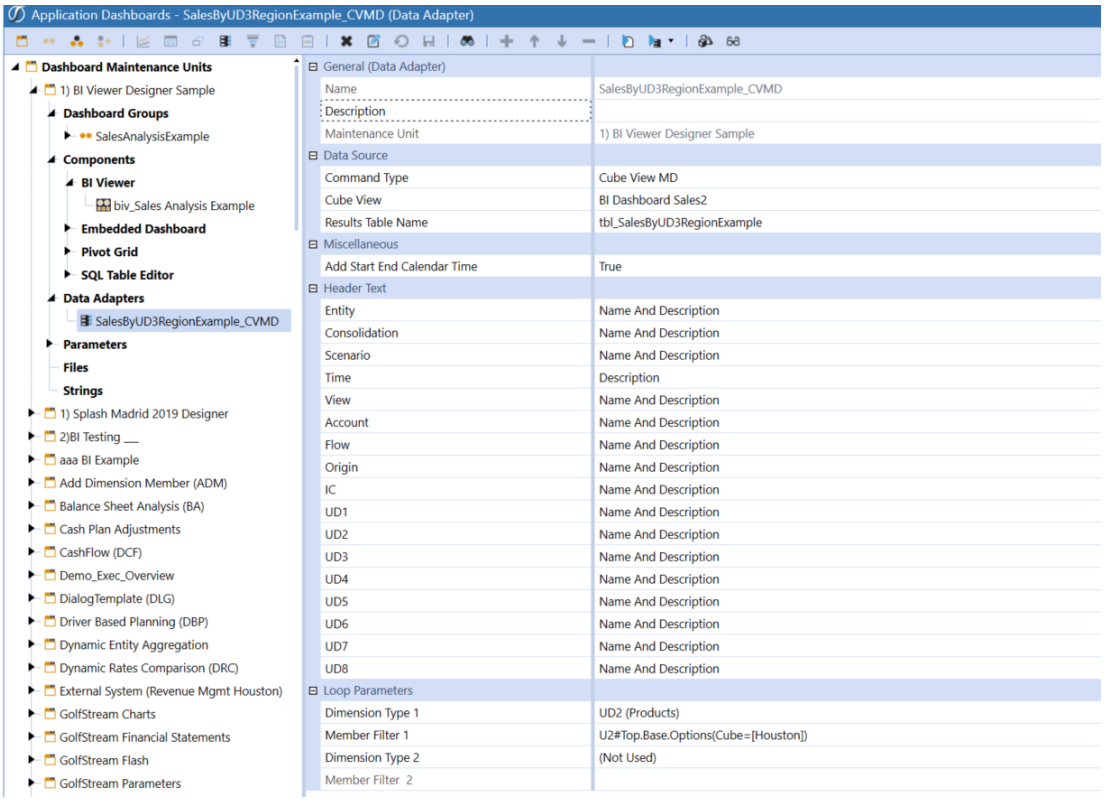
Results Table Name

This specifies the name of the resulting table generated when the Data Adapter is run, otherwise it will default with a name of Table.

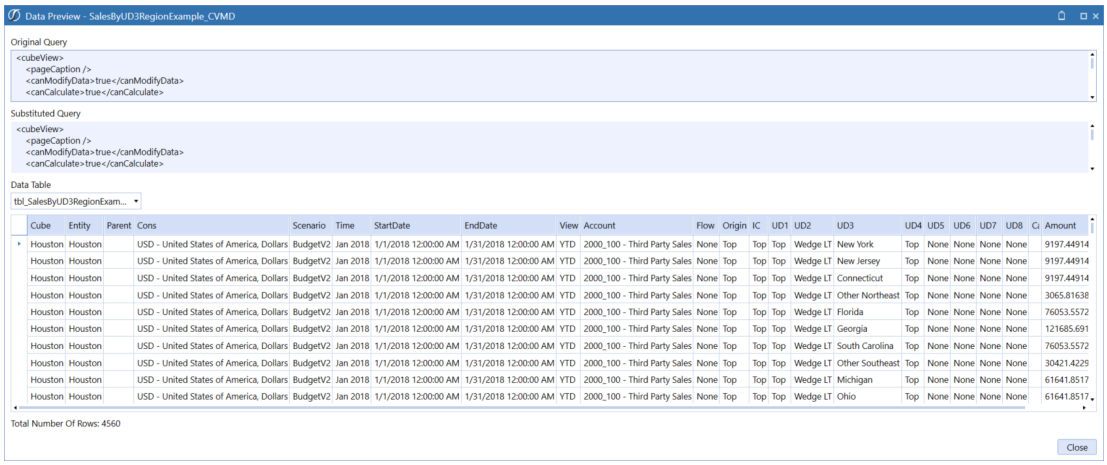
Cube View MD

Choose a Cube View MD (Multi-Dimensional) as the source of a Data Adapter. This Command Type will return the selected Cube View as a Multi-Dimensional Fact Table versus the reporting table that is returned by the Cube View Command Type. The results of the Cube View MD are Dimensions (Entity, Consolidation, Scenario, time, View, Account, Flow, Origin, IC, UD1-UD8) as columns. This simplifies the report building process in the BI Designer, Pivot Grid, and dashboard development. There are additional Loop Parameter options that can be selected here to include incremental information from the modified Cube View definition in the resulting tables. Adding on to what is defaulted, however, may have a slight impact on performance.

Cube View MD Data Adapter example



Results Table from the Cube View MD Data Adapter example above.



Cube View

This command type allows for a pre-configured Cube View to be the Data Source for a dashboard.

Click and begin typing the name of the Cube View in the blank field. As the first few letters are typed, the names are filtered making it easier to find and select the one desired. If the name is unknown, expand a Cube View Group and scroll through the list to select the correct one. Once the Cube View is selected, click CTRL and Double Click. This will enter the correct name into the appropriate field.

Results Table Name

This specifies the name of the resulting table generated when the Data Adapter is run, otherwise it will default with a name of Table. It is optional but highly recommended that a name is defined here to distinguish between different Data Adapters if there are many assigned when using the BI Designer.

For example; tbl_OperatingExpenses can be used to identify the Results Table Name and the Name of the Data Adapter (OperatingExpenses_CVMD in this example) associated with this table.

Add Start End Calendar Time

When set to True, the Data Table incorporates the Start and End Date used in the POV / Time Profile for the Cube View and creates two additional columns in a Date/Time Field Type Format; StartDate, EndDate for each row in the dashboard. This allows for the ability to utilize the Date Grouping functions in the BI Designer. Set this to False to not add or display these Date Time fields.

Entity

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the Entity. When set to Name, the Entity Name will be displayed in the results table from the Cube View. When set to Description, the Entity Description will be displayed in the results table from the Cube View.

Consolidation

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the Consolidation. When set to Name, the Consolidation Name will be displayed in the results table from the Cube View. When set to Description, the Consolidation Description will be displayed in the results table from the Cube View.

Scenario

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the Scenario. When set to Name, the Scenario Name will be displayed in the results table from the Cube View. When set to Description, the Scenario Description will be displayed in the results table from the Cube View.

Time

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the Time. When set to Name, the Time Name will be displayed in the results table from the Cube View. When set to Description, the Time Description will be displayed in the results table from the Cube View.

View

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the View. When set to Name, the View Name will be displayed in the results table from the Cube View. When set to Description, the View Description will be displayed in the results table from the Cube View.

Account

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the Account. When set to Name, the Account Name will be displayed in the results table from the Cube View. When set to Description, the Account Description will be displayed in the results table from the Cube View.

Flow

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the Flow. When set to Name, the Flow Name will be displayed in the results table from the Cube View. When set to Description, the Flow Description will be displayed in the results table from the Cube View.

Origin

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the Origin. When set to Name, the Origin Name will be displayed in the results table from the Cube View. When set to Description, the Origin Description will be displayed in the results table from the Cube View.

IC

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the IC. When set to Name, the IC Name will be displayed in the results table from the Cube View. When set to Description, the IC Description will be displayed in the results table from the Cube View.

UD1-UD8

At the creation of the Data Adapter, the default is set to Name And Description. This will display both the Name and Description of the UD1-UD8. When set to Name, the UD1-UD8 Name will be displayed in the results table from the Cube View. When set to Description, the UD1-UD8 Description will be displayed in the results table from the Cube View.

Loop Parameters

This section allows changes to be made to the output of a Cube View definition being used in a table for reporting. The Loop Parameter filters the results and considers the additional parameters to be passed to the Cube View definition and add those results to the table accordingly.

The parameter(s) overrides the POV. So, if the Entity POV is set to CT, and the loop filter parameters are set to NY, MA, and NJ, then data for those will be returned and NOT CT. A Loop must be used in order to change parameters.

For example, a Loop Parameter may be used to loop through a list of Entities in the Cube View definition and return multiple Entities for that specific Cube View. The Dimension Type and Member Filters should be added here to pass along the appropriate Loop (e.g. Dimension Type=Entity, Member Filter= E#US.Base) which applies to each Entity included in the Loop.

> **NOTE:** It is recommended to not loop on any Dimensions that already exist in the Cube View's rows or columns.

At the creation of the Data Adapter, the default for each Dimension Type (1 & 2) are set to (Not Used) and Member Filter (1&2) are greyed out. This will display the results without any consideration of additional parameters to pass to the query. When the Dimension Types are set along with the Member Filters, the results will consider the additional parameters to be passed to the Cube View definition and add those results to the table accordingly.

Dimension Type 1

The Dimension Type containing the list of Members. e.g., Entity or Account

At the creation of the Data Adapter, the default for each Dimension Type (1 & 2) are set to (Not Used) and Member Filter (1&2) are greyed out. This will display the results without any consideration of additional parameters to pass to the query. When the Dimension Types are set along with the Member Filters, the results will consider the additional parameters to be passed to the Cube View definition and add those results to the table accordingly.

Member Filter 1
Enter a Member Filter here to determine what is seen in the Parameter.

At the creation of the Data Adapter, the default for each Dimension Type 1 is set to (Not Used) and Member Filter 1 is greyed out. This will display the results without any consideration of additional parameters to pass to the query. When the Dimension Types are set along with the Member Filters, the results will consider the additional parameters to be passed to the Cube View definition and add those results to the table accordingly.

The name of the Dimension containing the list of Members. Start typing in the blank field OR click the ellipsis button in order to launch the Member Script Builder and enter a Member Script to change the Cube View definition. The example below is changing the POV for the Products.
Example: UD2; U2#Top.Base

Dimension Type 2
At the creation of the Data Adapter, the default for Dimension Type 2 is set to (Not Used) and Member Filter 2 is greyed out. This will display the results without any consideration of additional parameters to pass to the query. When the Dimension Types are set along with the Member Filters, the results will consider the additional parameters to be passed to the Cube View definition and add those results to the table accordingly.

Member Filter 2
Enter a Member Filter here to determine what is seen in the Parameter.

At the creation of the Data Adapter, the default for Dimension Type 2 is set to (Not Used) and Member Filter 2 is greyed out. This will display the results without any consideration of additional parameters to pass to the query. When the Dimension Types are set along with the Member Filters, the results will consider the additional parameters to be passed to the Cube View definition and add those results to the table accordingly.

The name of the Dimension containing the list of Members. Start typing in the blank field OR click the ellipsis button in order to launch the Member Script Builder and enter a Member Script to change the Cube View definition. The example below is changing the POV for the Products.
Example: UD2; U2#Top.Base

Dimension Leveling

The Dimension Level property setting in the Cube View MD Data Adapter displays dimensional data as a hierarchical tree in the BI Viewer. Dimension leveling allows you to display data in a hierarchical structure into which you can drill down to view child data. After the data is leveled, you can use the BI Viewer to view data as a tree, a pivot table, a chart, or a grid.

Prerequisites

To use dimension leveling to display hierarchical data, you must first create a cube view MD data adapter with data.

Create the Data Adapter

Cube View MD Data adapters allow you to pull data from a Cube View. To create a data adapter:

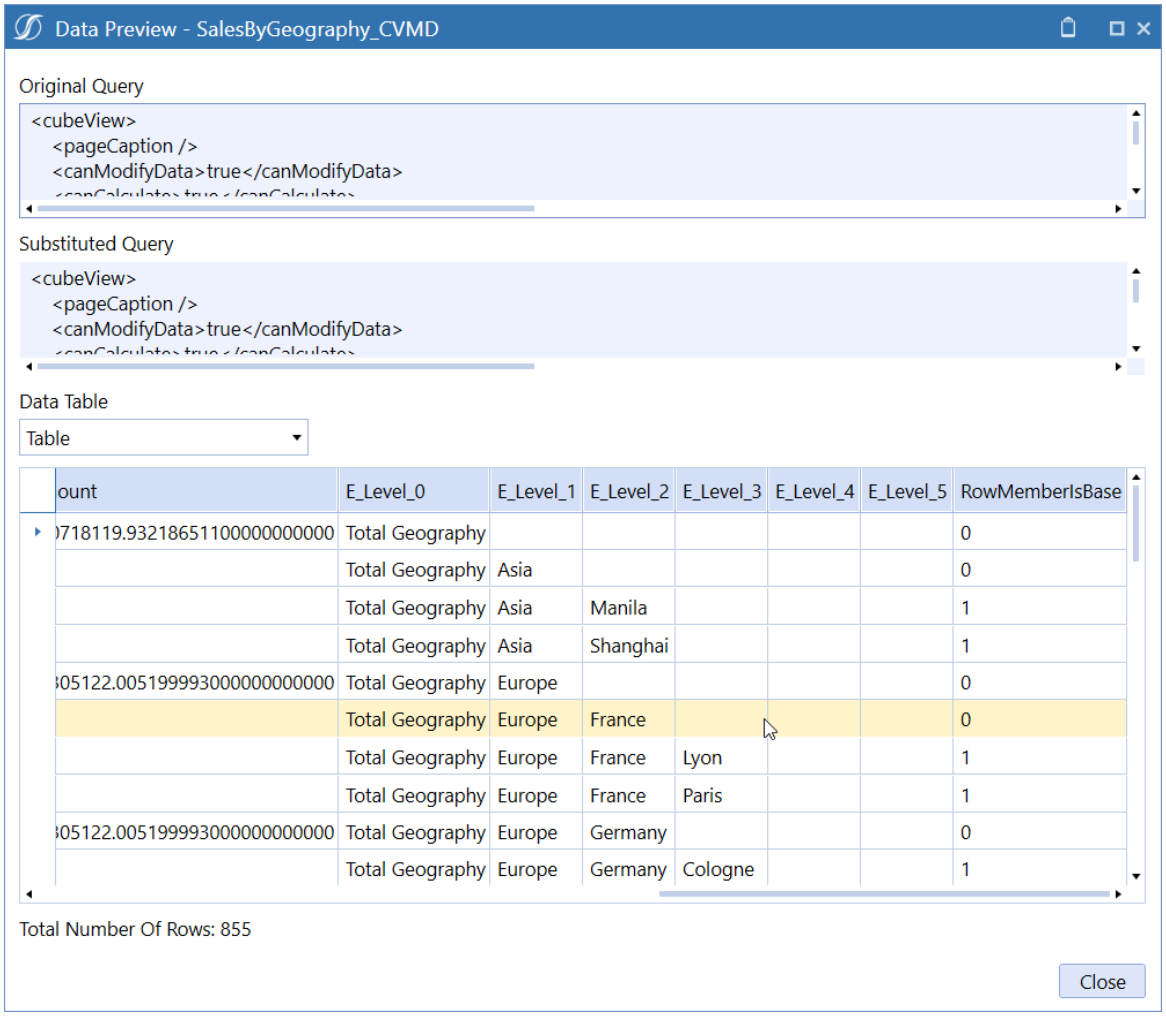
1. Inside the OneStream application, click the **Application Dashboards** tab at the bottom of the screen.
2. Expand the appropriate Dashboard Maintenance Unit.

3. Click the **Data Adapter** label.
4. Click the **Create Data Adapter** button on the toolbar.
5. In the Name field, type the name of the data adapter.
6. In the Command Type field, click the drop-down arrow and select **Cube View MD**.
7. Click the **Edit**button ( )at the far right of the Cube View field.
8. In the **Object Lookup** dialog box, select the appropriate Cube View and then click **OK**.
9. In the Dimension to Level field, click the drop-down arrow and select the appropriate leveling option. With this step, you are leveling on this dimension, which in this case is Entity.
10. Select one of the following:
    - **Outermost Row** - hierarchy uses the first row and first level (of row) definition.
    - **Outermost Column** - hierarchy uses the first column and first level (of column) definition.
    - **Both** - hierarchy uses the first row and first level definition and uses the first column and first level (of column) definition.

> **NOTE:** *When Data Adapter is run, the Data Table will generate the additional columns for the levels including a column(s) to determine the status of the level:*

RowMemberIsBase = 1: Row Member contains the Base level of data
RowMemberIsBase = 0: Row Member is not the Base level of data
ColumnMemberIsBase= 1: Column Member contains the Base level of data
ColumnMemberIsBase= 0: Column Member is not the Base level of data

11. Click **Save**.
12. Test the data adapter by clicking **Test Data Adapter**. The **Data Preview** dialog box displays the table data.



> **NOTE:** When Data Adapter runs, the Data Table generates the additional columns for the levels including columns to determine the status of the level.

Each dimension leveled column is prefixed with an E to indicate it is an Entity, followed by a level number. A RowMemberIsBase value of 1 indicates that there is no remaining child data.

13. Click **Close**.

View Results in the BI Viewer

You can view data in various ways using the BI Viewer. For example, you can configure the BI Viewer to display data in pivot tables and charts at the same time. To configure the BI Viewer:

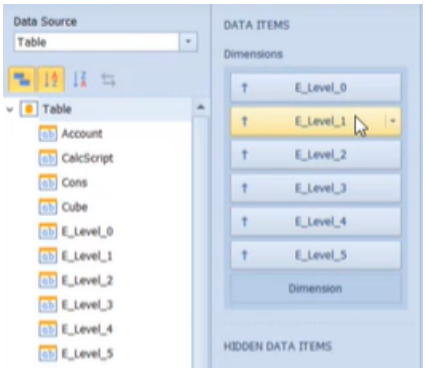
1. Click the Application Dashboard tab at the bottom of the window.
2. Create a BI Viewer component:
    a. Click the **Components** label in the tree.
    b. Click the **Create Dashboard Component** button on the toolbar.
    c. In the **Create Dashboard Component** dialog box, click **BI Viewer** and then click **OK**
    d. On the Component Properties tab, in the Name field, type the name of the new component.
    e. Click the **Data Adapters** tab at the top of the window.
    f. Click the **Add Dashboard Component** button on the toolbar.
    g. In the **Add Data Adapter** dialog box, select the appropriate data adapter and then click **OK**.
    h. Click **Save**.

3. Design the BI Viewer dashboard:

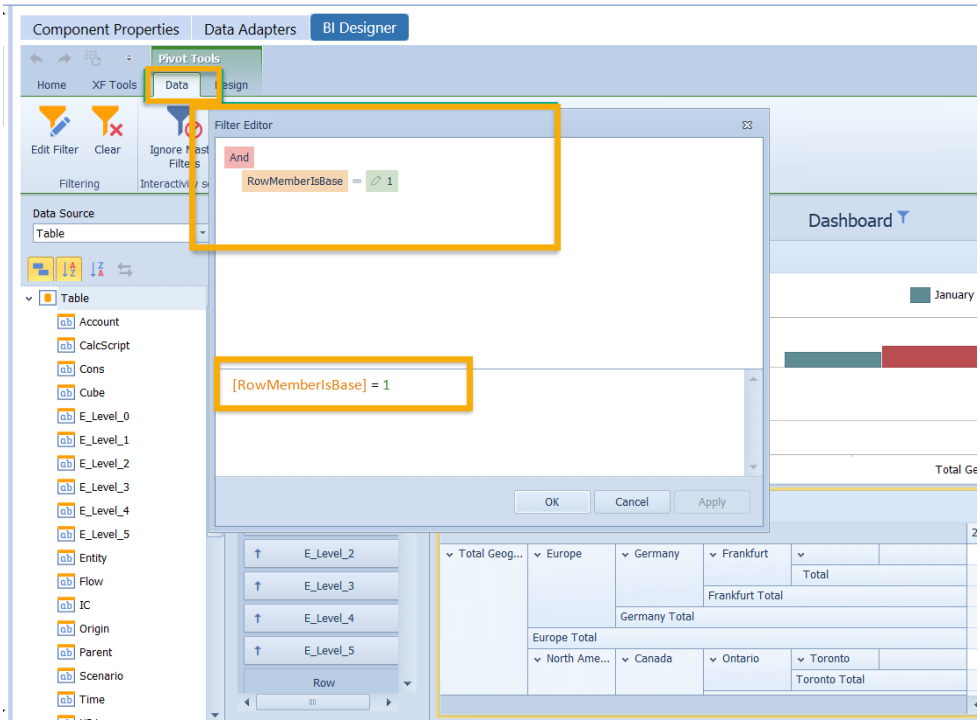
    a. Click the **BI Designer** tab at the top of the window.

    b. In the Data Source field, click the drop-down arrow and select the data source from which to pull data. This is usually a table.

    c. Drag each dimension level that you want to view from the Table view into the Data Items column under Dimensions.

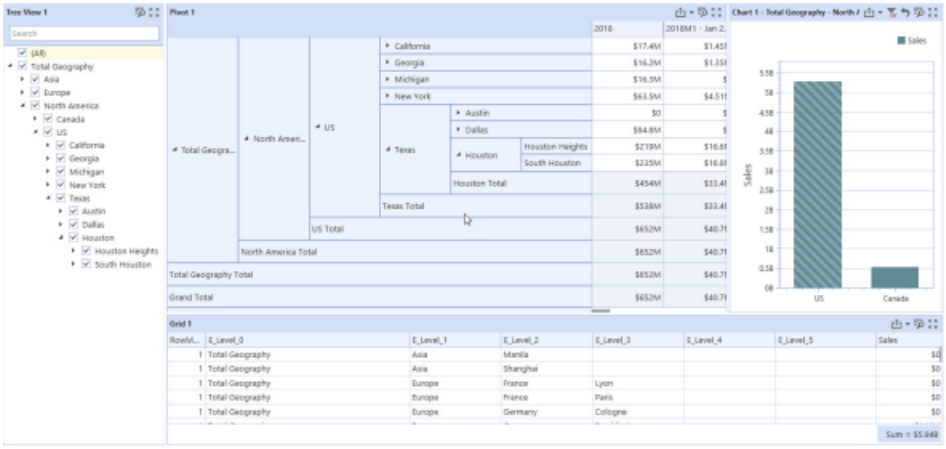


    d. If you want to filter items based on whether they are base items, drag the **RowNumberIsBase** item into the Data Items column but under Hidden Data Items. This item will not display in the resulting dashboard but will be available for you to filter on if necessary.



e. From the BI Designer ribbon, select the type of dashboard item you want to view in the dashboard. For example, select **Filter Element** > **Tree View**. In the resulting tree view in the dashboard notice that you can expand parent members down to their base child members.

If you add a Grid, all levels of the parent entity display in their own columns. If you want to see a pivot table, select **Pivot** from the ribbon and notice that each level is expandable similar to tree view.



Conclusion

With Dimension leveling, you can view data in an easy-to-understand, hierarchical format. With the BI Viewer, you can design various dashboard items in which to view the dimension leveled data.

Method Query

> **TIP:** To view an example Method Query, leave the Method Query field blank, click Save, and then click **Test Data Adapter** .

Method Type

BusinessRule

Use the Business Rule option when creating a custom rule to incorporate within a Method Query. The Business Rule is used as the first set of {} within the Method Query.

```
Example Method Query: {XFR_DataUnitCompare}{DataUnitComparisonDataSet}
{Cube1=|!Members_Cubes!|, Entity1=|!Members_Entities_AllDims_Base!|,
Parent1=[], Cons1=|!Members_Cons_Statutory!|,
Scenario1=|!Members_Scenarios_AllDims_Base!|, Time1=|!Members_Time_WFYear_
Base!|, Cube2=|!Members_Cubes!|, Entity2=|!Members_Entities_AllDims_Base!|,
Parent2=[], Cons2=|!Members_Cons_Statutory!|, Scenario2=|!Members_Scenarios_
AllDims_Base_Var!|, Time2=|!Members_Time_WFYear_Base!|, View=
|!Members_View_Numeric!|, SuppressMatches=|!DataUnit_SuppressMatches!|}
```

CertificationforWorkflowUnit

This lists all Certification Questions for the particular Workflow Unit.

Example Method Query:

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Include Descendants}{} or
 {Dallas}{Actual}{2011M2}{true}{}.
```

ConfirmationforWorkflowUnit

This lists the Confirmation Rules results for a particular Workflow Unit.

Example Method Query:

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Include Descendants}{Filter}
or {Montreal}{Actual}{2011M6}{true}{} Name}{Include Descendants}{Filter} or
{Montreal}{Actual}{2011M6}{true}{}
```

DataUnit

This returns all rows of data related to the specified Data Unit (i.e. Cube, Entity, Parent, Consolidation Member, Scenario, Time and View).

Example Method Query:

```
{Cube}{Entity}{Parent}{Cons}{Scenario}{Time}{View} {True}
{Empty String or Filter Expression}
```

DataUnitComparison

This returns all rows from two different Data Units specified for comparison purposes.

Example Method Query:

```
{Cube1}{Entity1}{Parent1}{Cons1}{Scenario1}{Time1}
{Cube2}{Entity2}{Parent2}{Cons2}{Scenario2}{Time2}{View}{True}{True}{
Empty String or Filter Expression}
```

Excel

This returns data sourced from an Excel file. Parameters for method type 'ExcelFile' should include the following, each enclosed in curly braces:

{FileSourceType}{UrlOrFullFileName}{Sheet}{NamedRange}{UseFirstRowForColumnNames}{IncludeNumericColumns}{Filter},

Example:

```
{Application}{Documents/Public/MyExcelfile.xfDoc.xlsx}{Sheet1 or Empty}
{MyDataRange or Empty}{False}{False}{}.
```

FormsStatusForWorkflowUnit

This lists detailed information about the Forms for a particular Workflow Unit. Example Method Query:

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Form Status}{Filter} or
{Houston}{Actual}{2011M1}{All}{}
```

Groups

This returns the Group ID, Name, Description and whether or not this is an Exclusion Group. For example:

```
{GroupName = 'FinanceGroup'}
```

GroupsforUsers

Select User properties and all of the Groups to which the user. This returns the same group properties as the Group Method Query.
For example:

```
{UserName = 'Administrator'}{}
```

ICMatchingforWorkflowUnit

This returns a detailed Intercompany Matching Report table for the given Workflow Unit and several other Parameters. The Parameters override what is set up in the Workflow Profile. For example:

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Plug Account Override}{
Suppress Matches Override}{Tolerance Override}{Filter} or
{Flint}{Actual}{2011M1}{Empty String or A#MyPlugAccount}{Empty String or
true/false}{Empty String or 0.0}{Empty String or Filter Expression}.
```

ICMatchingPlugAccountsforWorkflowUnit

This returns the list of Intercompany Plug Accounts for a given Workflow Profile and Scenario Type configured for the Workflow Profile.

Method Query Example

```
{MyWorkflowProfileName}{Actual}{2011M1}
{PlugAccount for Workflow Parameter Set (Exclude A#)}{Empty String
or C#MyCurrencyOverride}{Empty String or V#MyViewOverride}{Empty String or
A#MyPlugAccountOverride}{Empty String or true/false}{Empty String or 0.0}
{Empty String or E#MyEntityOverride}{Empty String or E#MyPartnerOverride}
{Empty String or MyDetailDimsOverride (F#All:O#Top:U1#All:U2#All:U3#All:
U4#All:U5#All:U6#All:U7#All:U8#All)}{Empty String or Filter Expression}
```

JournalforWorkflowUnit

This lists the Journals entered for a given Workflow Unit

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Journal Status}{Filter} or
{Frankfurt}{Actual}{2011M3}{All}{}
```

Members

This returns Dimension ID, Member information such as ID, Name and Description, and a few other properties for the chosen Dimension and Member Filter.

```
{Account}{MyAccountDim}{A#Root.TreeDescendants}
{Empty String or Filter Expression}
```

UserCubeSliceRights

This lists each user's Data Access settings on a given Cube.

For example:

```
{UserName}{CubeName}{Filter} or {AllUsers}{AllCubes}{}.
```

UserEntityRights

This returns the Cubes and Entities the user has access to according to the security settings under Entities. For example:

```
{UserName}{CubeName}{Filter} or {US Clubs Controller}{AllCubes}{}.
```

Users

This returns all properties associated for the chosen User Name.

For example:

```
{UserName = 'Administrator'}
```

UserScenarioRights

This returns all accessible Scenarios and many related Scenario properties for the chosen User Name filter and Cube.

Method query example: {AllUsers}{AllCubes}{Empty String or Filter Expression}

UserinGroups

This returns a list of Users and selected User properties for the chosen User Group. For example:

```
{GroupName = 'FinanceGroup'}{}
```

UserWorkflowProfileRights

This lists the rights assigned to users for Workflow Profiles.

Method query example:

```
{User Name}{Workflow Cube Name}{Workflow Profile Type}{Filter} or
{Administrator}{GolfStream}{AllProfiles}{}.
```

WorkflowandEntityStatus

This returns properties for Workflow status, status code/description, last executed step, date/time information, completed steps, and data status for the chosen Workflow Unit including both the Workflow Profile level and individual Entity level.Method query example:

```
{MyWorkflowProfileName}{Actual}{2011M1}
{AllProfiles}{Descendants}{Empty String or Filter Expression}
```

WorkflowCalculationEntities

This lists the Entities that appear under Calculation Definitions for this Workflow Profile.

WorkflowConfirmationEntites

This lists the Entities that appear under Calculation Definitions with a Confirmed check box for this Workflow Profile.

WorkflowProfileandDescendantEntities

This creates a list of Entities and all descendants located under Entity Assignment for this Workflow Profile.

WorkflowProfileEntities

This creates a list of Entities located under Entity Assignment for this Workflow Profile.

WorkflowProfileRelatives

This lists related Workflow Profiles based on certain criteria.

For example:

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Workflow Profile Type}
{Relative Type}{Include Requesting Profile}{Filter} or {GolfStream}{Actual}
{2011M1}{AllProfiles}{Descendants}{true}{}.
```

WorkflowProfiles

```
This lists the Workflow Profiles. Example:
{WorkflowProfileType}{Filter} or {AllProfiles}{Type = 'InputAdjChild'}.
```

WorkflowStatus

This lists the status, lock status and last step completed of a given Workflow Unit. For example:

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Workflow Profile Type}
{Relative Type}{Filter} or {Houston}{Actual}{2011M1}{AllProfiles}
{Descendants}{}
```

WorkflowStatusTwelvePeriod

This returns status value and text summary for 12 months for a given Workflow Profile, Scenario and Year. For Workflow Profile Type, options are AllProfiles, CubeRoot, Default, Review, BaseInput, InputImportChild, InputFormsChild, InputAdjChild and ParentInput. Example:

```
 {MyWorkflowProfileName}{Actual}{2011}{AllProfiles}
{Descendants}{Empty String or Filter Expression}
```

Method Query

The query ran for this Data Adapter.

Results Table Name

This specifies the name of the resulting table generated when the Data Adapter is run, otherwise it will default with a name of Table.

WorkflowLockHistory

Displays all Lock history details in a report for a given Workflow Profile with the ability to filter by Scenario, Time, Workflow Profile, Origin, Channel, Time, User and Lock Status. For example:

```
{Workflow Profile Name}{Scenario Name}{Time Name}{Workflow Profile Type}
{Relative Type}{Filter}
```

Method Query Parameter Options

Workflow Profile Type

AllProfiles, AllProfilesExceptInputChild, AllProfilesWithTemplates, AllTemplates, BaseAndParentInputProfile, BaseInput, ClassStandardProfiles, ClassTemplateProfiles, CubeRoot, Default, InputAdjChild, InputChildren, InputFormsChild, InputImportChild, NonInputParentProfiles, ParentInput and Review.

Workflow Profile Relative Type
Ancestors, Descendants, FirstLevelChildren, Siblings and Unrelated.

Workflow Status or Forms Status
Completed, HasError, InProcess, NoStatus, NotExecuted and Unknown.

Journal Status
Approved, Posted, Rejected, Submitted

SQL

A SQL query against either the Application or Framework database can be written as a Data Source. Reference substitution variables such as |WFProfile| from within the SQL statement.

Database Location

Application
The current OneStream Application database where Stage and financial Cube data resides.

Framework
The connected OneStream Framework database where security and log data resides.

External
Any other database outside of OneStream.

External Database Connection
If External is the chosen Database Location, select the External Database Connection name here. This list is defined in the OneStream Server Configuration Utility.

SQL Query
The SQL statement ran for this Data Adapter.

Results Table List

This specifies the name of the resulting table generated when the Data Adapter is run, otherwise it will default with a name of Table.

BI Blend Adapter

The purpose of the BI Blend Adapter is to provide dashboard designers with a pre-defined interface to querying BI Blend tables. The data sourced by the BI Blend Adapter can be used to design standard OneStream dashboard.

The BI Blend Workflow process is designed to generate large external database tables formatted in a column store index optimized for analytic reporting. The overall number of records that may be generated by the BI Blend process may be too large for the Dashboard components to process. Therefore, the designer should manage the returned dataset by defining an appropriate Where Clause to retrieve a “slice” of the BI Blend table which is suitable for the BI Blend Adapter.

⊟ General (Data Adapter)	
Name	BIBlend_DataAdapter
Description	BI Blend Adapter
Maintenance Unit	BIBlend
⊟ Data Source	
Command Type	BI-Blend
Results Table Name	BIBLEND1
Table Info	BIB_|AppName|_|WFText1|_|WFScenario|_|WFTime|
Group By	Entity,HoustonProducts,HoustonCustomers
Data Field Aggregation Types	AggType1 = [2018M1, Sum],AggType2 = [2018M1, Count]
Where Clause	HoustonCustomers= 'shanks'

Results Table Name

Define a table name to store the content of the BI Blend adapter query.

Table Info

This defines the name of the BI Blend table name to be queried. This label supports the use of Parameters and Substitution Variables.

Group By

This defines the source database columns, and their order, which are to be returned to the adapter. These labels must match the names specified on the database. Any field to be pivoted must be in the Group by list.

Data Field Aggregation Types

The Data Field Aggregation Type is used to identify which database column should be used as the Measure field. The adapter can support multiple database columns as Measures. Each defined measure can be defined to derive the results as a different Aggregation Type. Parameters and/or Substitution Variables are supported.

- Sum
- Min
- Max
- Avg
- Count

The syntax required is to define each Aggregation Type result as a unique key. AggregationType = [databaseColumnName, Type]

- AggType1 = [2018M1, Count], AggType2 = [2018M1, Sum]
- AggType1 = [|WFYear|M1, Count]

Where Clause

The is used to “pre-filter” the table. The BI Blend Adapter will query the source table and return all the results to the client for processing, such as for use in a Pivot Grid. The size of the query must be managed to ensure the overall performance of the Dashboard report is optimized. The Where Clause uses standard SQL syntax to define the filters which are applied to the query.

- DatabaseColumn = ‘Text’
- DatabaseColumn Like ‘2019%’ - As all columns beginning with 2019
- DatabaseColumn1 = ‘CostCenter’ And DatabaseColumn2 = ‘Midwest’

Direct Load Reporting Data Adapter Method Types

Dashboard reporting against Direct Load Workflow activity is simplified with specialty Data Adapter Methods.

- **Direct Load Info**: Retrieves results from the StageDirectLoadInformation table. Method query example:

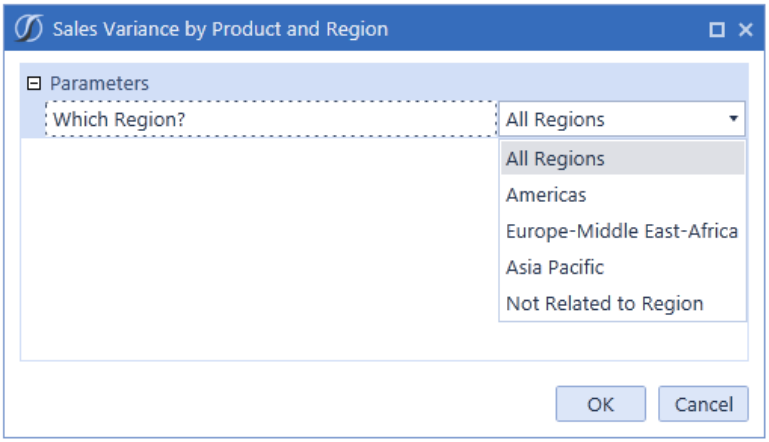
  ```
  {Workflow Profile Name}{Scenario Name}{Time Name}{Filter}
  ```

- **Stage Summary Target Data**: Query designed to automatically determine the method used to manage Summary Target Data, as Row or Blob, and appropriately return the results. Method query example:

  ```
  {Workflow Profile Name}{Scenario Name}{Time Name}{Filter}
  ```

Parameters

Parameters prompt you to enter or modify values so you can filter the data to display on a dashboard. For example:



See:

- About Parameters
- Create a Parameter

Files

Administrators use the File option to upload images and documents to create personalized, company-specific Dashboards. Items such as PDF report books and Word or Excel Extensible Documents can also be stored and used in Dashboards. You can:

- Specify a file name and a brief description.
- Identify the maintenance unit to which the file belongs.
- Upload or browse to the content file to use.

Strings

Strings are setup as an object type under a Dashboard Maintenance Unit. Strings can be used with XFString functionality and displayed in a cube view as a Caption or Heading, for example. See "Report Alias" for more information.

Linked Dashboards

Similar to the linked cube views, linked dashboards offers more flexibility and ease of use when performing data analysis. It provides the option to launch a dashboard from a cube view when the latter is viewed in either the data explorer grid, the Windows Application Spreadsheet tool, or the Excel Add-In.

When you right-click a cube view data cell, the context menu displays a list of linked dashboards along with linked cube views. Selecting a link launches the linked dashboard in a dialog providing more detail and visibility.

Linked dashboards enhance data analysis, providing an all-encompassing, guided-reporting experience as the data presented in the linked reports is directly related to the main cube view through Bound parameters. It also eliminates the need to incorporate too much detail in a single cube view or dashboard because you can drill down into the details as needed.

Reporting and Linking Objectives

Reporting objectives help you understand how to link the reports and determine what will be built. You should determine the following:

- Overall purpose of the linked dashboards in relation to the source cube view.
- The data you want to display via the source cube view as well as the detail data you want to display in the linked dashboards. This may require additional detail cube views.
- The POV information you want to pass from the source cube view to the dashboard. This determines the Bound Parameters needed on the source cube view.
- How the POV information will be used in the dashboard. This determines where the Bound Parameters need to be defined.

Data entry example: Using a linked dashboard and cube view for Budget data entry. The objective is to launch an Operating Expense dashboard form from a Budget Review cube view (the source cube view). Linked forms must be specific to the selected entity, time, and account from the source cube view.

Source Cube View Setup

1. Create a source cube view to link to the dashboard and/or cube view.

   Source cube view example: The Budget Review cube view displays parent Operating Expense Accounts by scenario. Two Member List parameters are used to select an entity and time period at run time.

Budget Review Entity: Houston Heights Time: Jan 2022					
	Baseline Budget V1	Revisions	BudgetV2	Target	Comments
50300 - Total Employee Compensation	1,928,768	214,409	2,143,177	2,281,469	
51099 - Total Utilities	68,094		68,094	68,592	
51199 - Total Professional Services	43,516		43,516	43,835	
52099 - Marketing & Advertising	177,289		177,289	178,586	
52199 - Travel & Entertainment	57,147		57,147	57,565	
52299 - Total Facility Expense	76,494	4,390	80,884	77,053	
52399 - Total HR Expenses	6,666		6,666	6,715	
52499 - Total Equip Expense	46,568		46,568	46,909	
53099 - Total Telecom	13,371		13,371	13,469	
53199 - Total R&D Expenses	62,603		62,603	63,061	
54099 - Depreciation & Amortization Expense	5,269		5,269	9,434	
54199 - Total Other Operating Expenses	87,427		87,427	88,067	
54400 - Total Operating Exp Before Allocation	2,573,214	218,799	2,792,013	2,934,754	
54299 - Total Allocations In	503,328	-503,328			
54300 - Allocations Out	-503,328	503,328			
54350 - Total Allocations	0				
54500 - Total Operating Expenses	2,573,214	218,799	2,792,013	2,934,754	
62000 - Total Operating Income	4,145,096	-218,799	3,926,297	4,315,560	

2. Define a Bound Parameter for each POV member you need to pass to the linked reports (this varies and may not always be applicable).

Source cube view Bound Parameters example: The linked form in this example must be specific to the selected entity, time and account from the source cube view. Each of those dimensions require a Bound Parameter.

Bound Parameter Names	
Entity Bound Parameter Name	drillentity
Consolidation Bound Parameter Name	
Scenario Bound Parameter Name	
Time Bound Parameter Name	drilltime
View Bound Parameter Name	
Account Bound Parameter Name	drillaccount

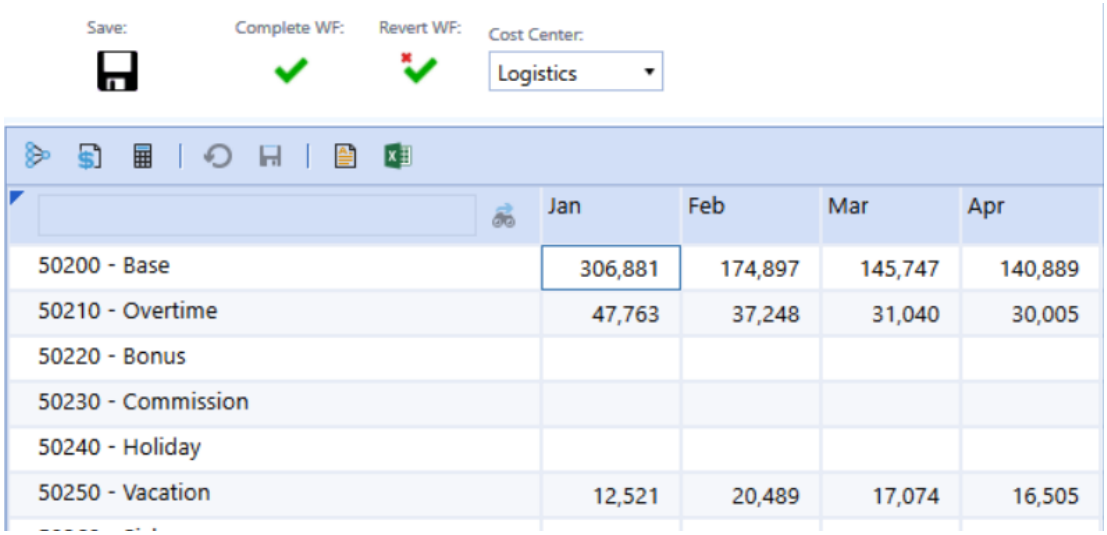
Linked Dashboard Setup

Dashboard setup varies based on the report purpose and how the dashboard and cube views are linked.

The Operating Expense dashboard contains:

- Button Components: Run specific Save and Workflow Task Actions
- Combo Box Component: Select a Cost Center

- Cube View Component: Display an Operating Expense data entry Cube View



Cube View Bound Parameters in the Linked Dashboard

When using Bound Parameters on the source cube view, it is important to understand how the source cube view's Bound Parameters relate to the dashboard's components, and how they impact the dashboard results.

The intent of Bound Parameters is to pass the parameter value in the background rather than through a prompt, thereby creating a seamless navigation experience. Bound Parameters impact the dashboard when there is a member dependency between one of its components and the source cube view.

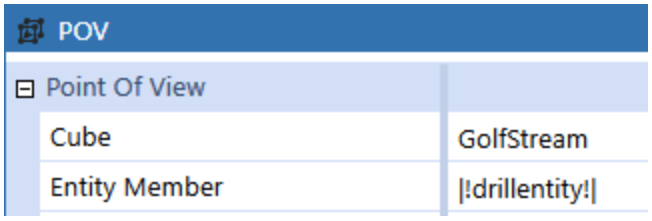
Passing Bound Parameters from a cube view to a linked dashboard requires:

1. Properly applying them to the component data source.
2. Creating a supplied parameter dashboard component for each cube view Bound Parameter and assigning the components to the dashboard.

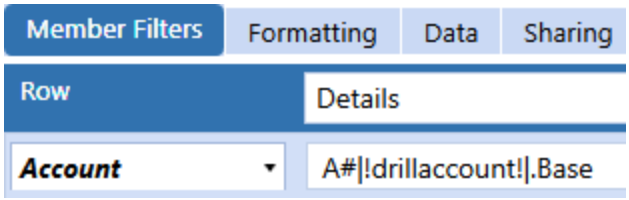
Requirement 1

The data source for the dashboard cube view component is an Operating Expense cube view. The results of this cube view are dependent on the source cube view account, entity and time members.

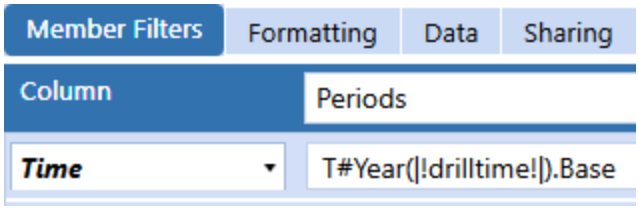
To pass these values, three Bound parameters (drilltime, drillaccount, drillentity) from the source cube view must be assigned to the OpEx (Operating Expense) cube view. This is the cube view assigned to the cube view dashboard component and it must use the same entity member.



Rows must display the Base Accounts of the source cube view selected parent account:



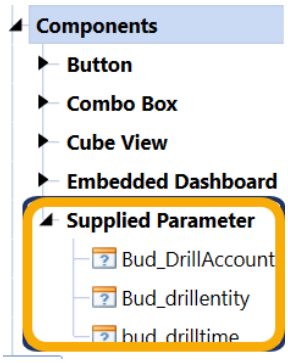
Columns must display time periods based on the selected year in the source cube view:



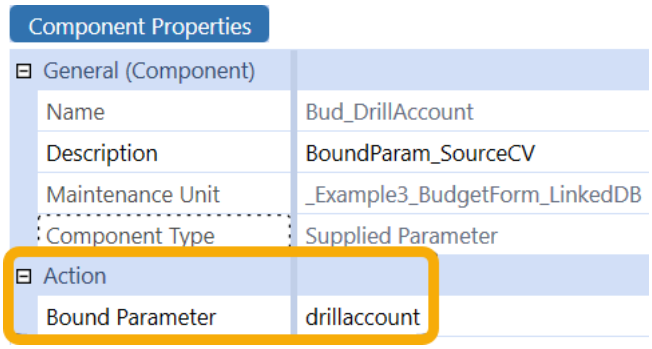
> **NOTE:** The selected Cost Center in the Dashboard combo box is also used by the OpEx cube view. This does not require a cost center (UD1) value/Bound Parameter from the source and is specific to the dashboard functionality.

Requirement 2

The source cube view uses three Bound parameters: drillaccount, drillentity, and drilltime. Each requires three supplied parameter components.

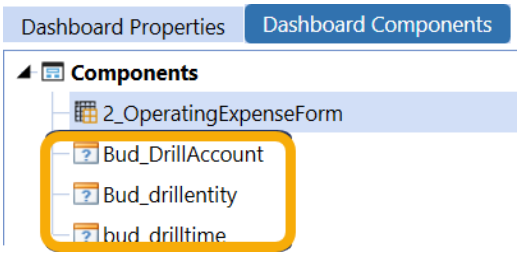


1. Type the name of the respective cube view Bound parameter in the Bound Parameter.



2. Construct and assign dashboard components as normal making sure to include these

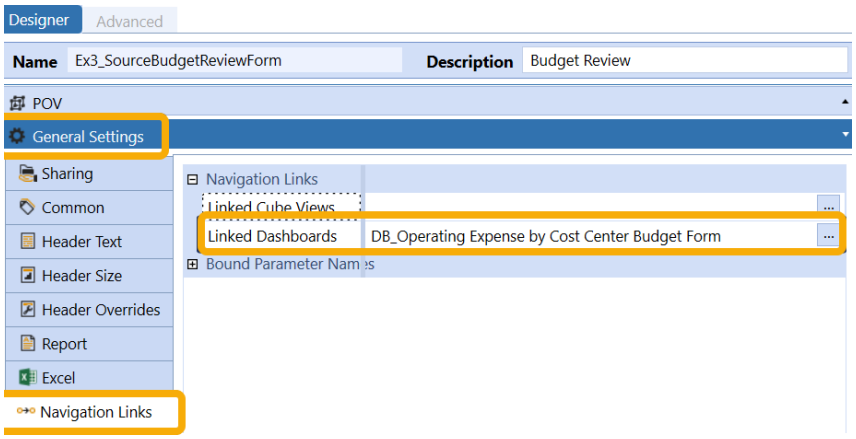
supplied parameters.



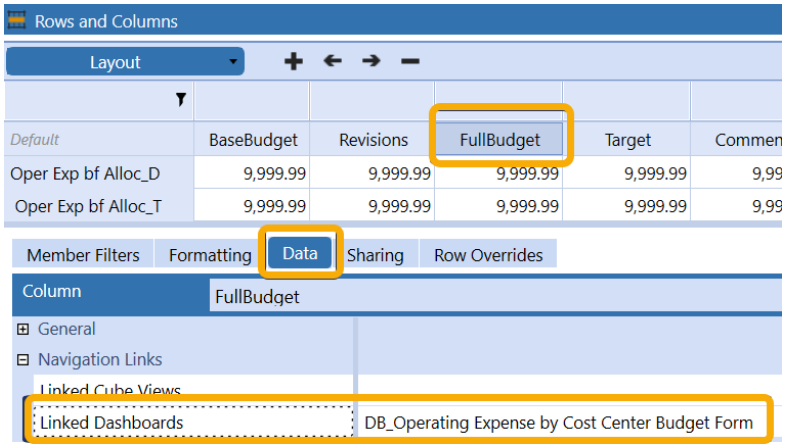
Link the Dashboard to the Cube View

Navigate back to the source cube view and assign the dashboard accordingly. This has the same behavior as linked cube view.

The dashboard is available when you right-click on any cube view data cell.

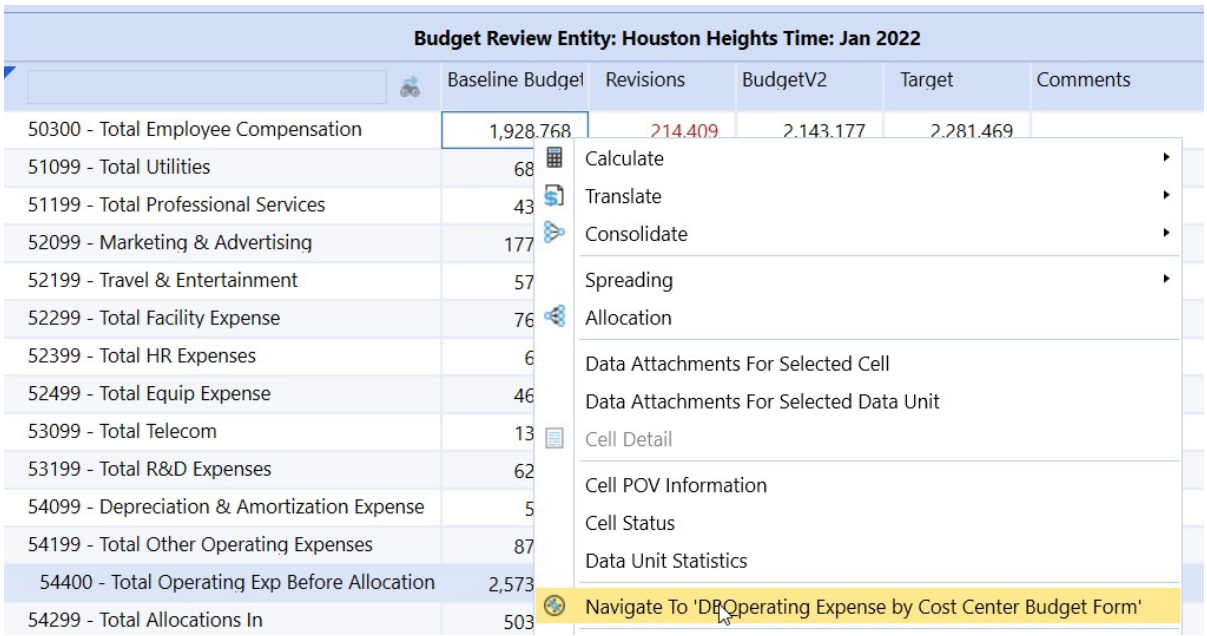


The dashboard is only available when you right-click on data cells in this specific column or row.



Launch the Dashboard

1. Run the source cube view.
2. Right-click on a data cell (this varies based on where it's assigned).
3. Navigate to the dashboard.



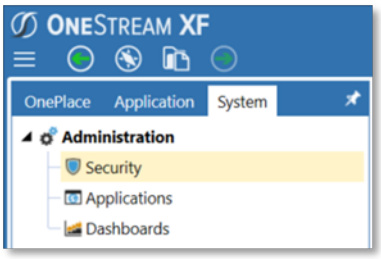
Use Cases

This functionality gives you the ability to drill down to:

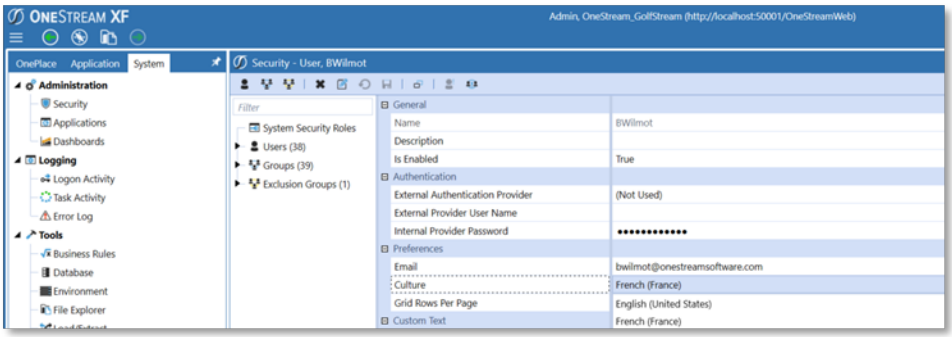
- Interactive dashboards for detailed analysis (for example, the BI Viewer and Advanced Charts)
- Workspaces
- Resources

User Culture

A User's culture is set in the Culture property field in each User's configuration. This is located within the Security section in the System tab under Administration.



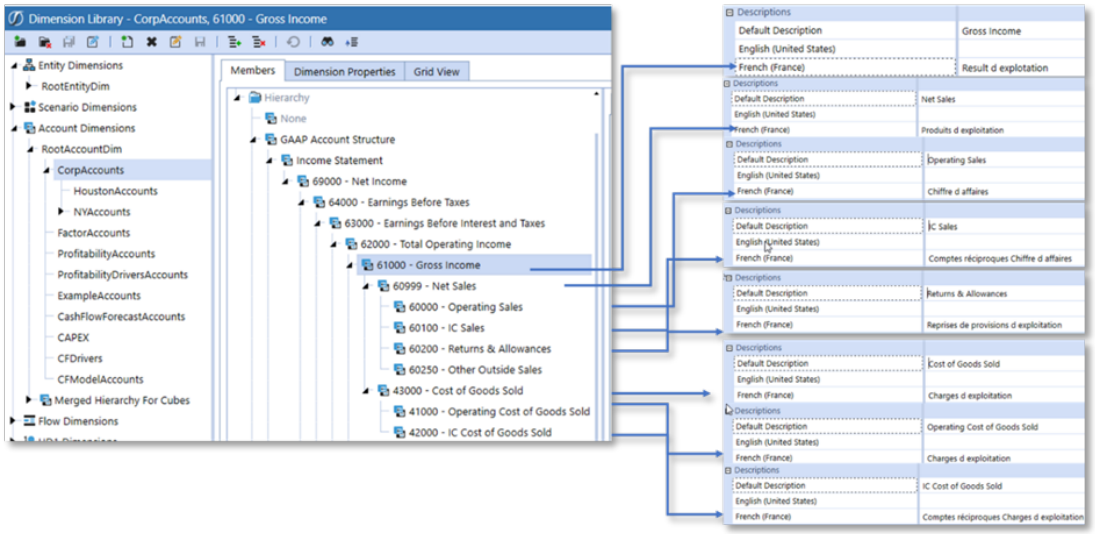
When the User is selected, a property grid will display. The Preferences tab within the property grid will contain an option for Culture which can be selected from the dropdown field. These options are controlled in the OneStream Server Configuration Utility and additional languages can be added there as needed.



Alias Dimension Member Descriptions

Dimension members can have more than one Description to reflect other languages for that culture. The properties for the members will be reflected in cube views and quick views as related to the user's default or prompted culture settings.

In this example, select the Dimension tab (e.g. Account Dimension shown below). Next select the desired Dimension members (e.g. 61000 – Gross Income) to be updated. In the Descriptions field under the Member Properties tab, enter the translated description (e.g. Result d explotation) next to the language (e.g. French) being updated.



Alias Member Descriptions in Cube Views

Dimension members can have more than one description to reflect other languages for that culture. The properties for the members will be reflected in cube views and quick views as related to the user's default or prompted culture settings. This can be achieved by making certain changes to the cube view's general settings.

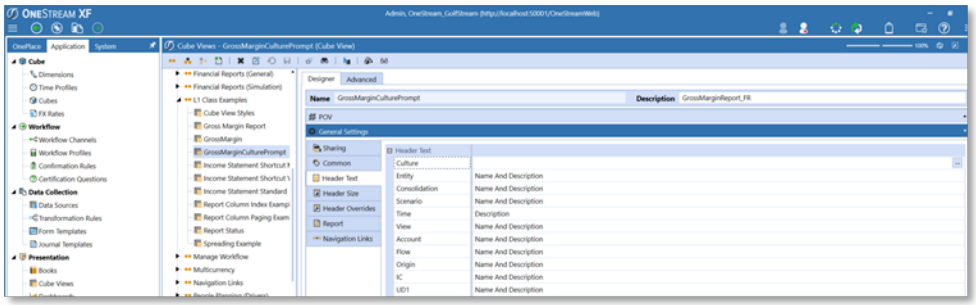
On the **Application** tab, select **Cube Views**, then select an existing cube view from Cube View Groups (or create a new one). On the **Designer** tab click **General Settings** and then select the Header Text box.

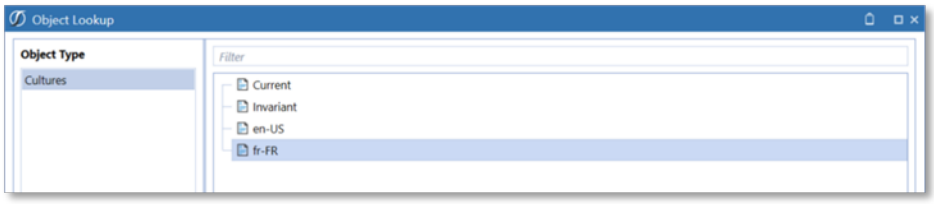
The Culture setting is associated with how the member description will be displayed in the cube view. This setting can be set multiple ways to display in different languages associated with the application. Selecting the ellipsis (…) at the end of the Culture field will provide options to define how the Members are displayed in the cube view:

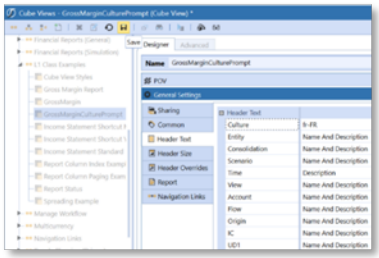
Current (default) – When blank, reacts the same as Current, which means the current User's Culture setting is used to apply a Report Alias during cube view rendering.

Invariant will display the members in their Default Description as defined in the item's member properties.

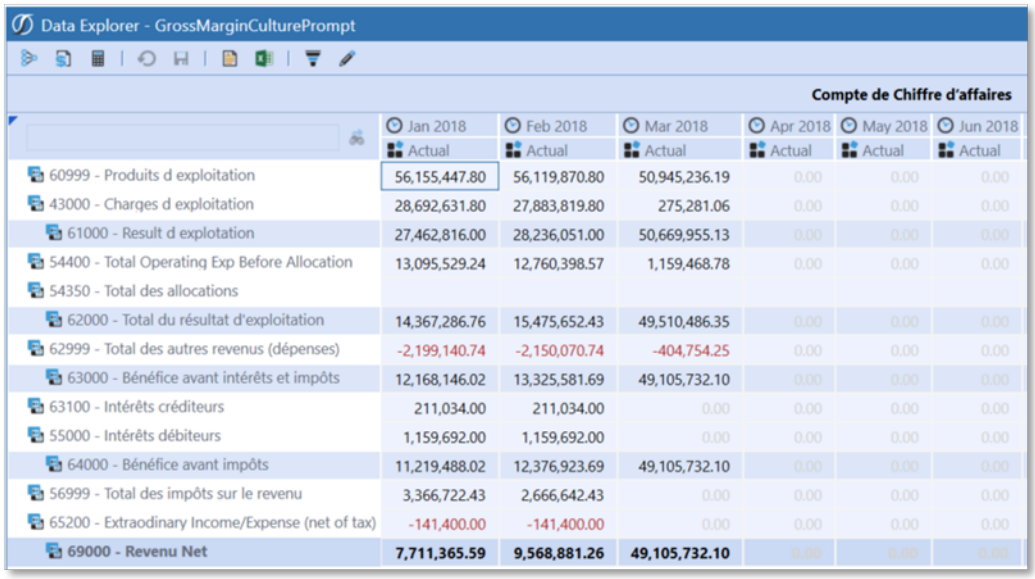
Selecting other cultures (e.g. fr-FR, as shown below) will display members in their respective languages for any User running the cube view. In the example below, we selected a Culture of "fr-Fr" and saved to display the members in French.







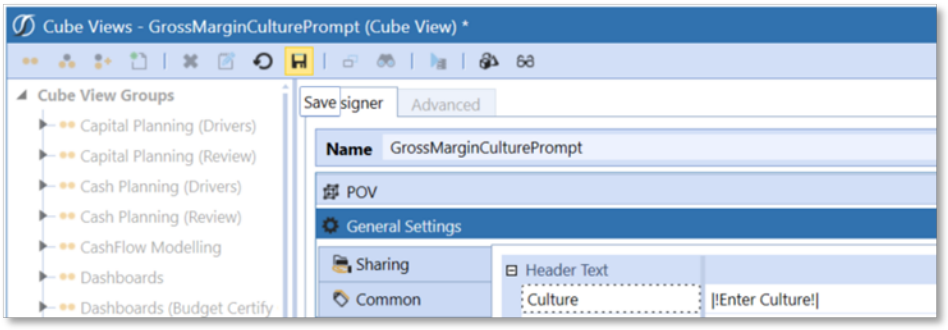
The results display the Account members in French for the given cube view.



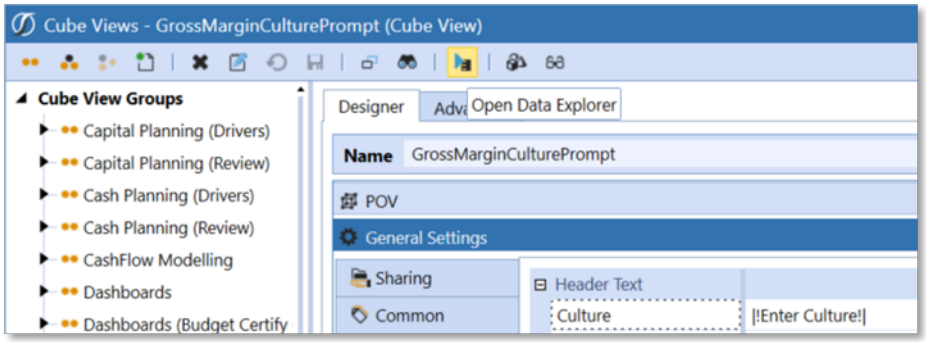
Modify the Culture Setting to Prompt the User for an Alternate Language

The Culture setting can also be set as a prompt to provide the option to change the language each time the cube view renders. The user can type in a Parameter to be used to prompt the user for a culture that will override the user's culture setting established in their security setup.

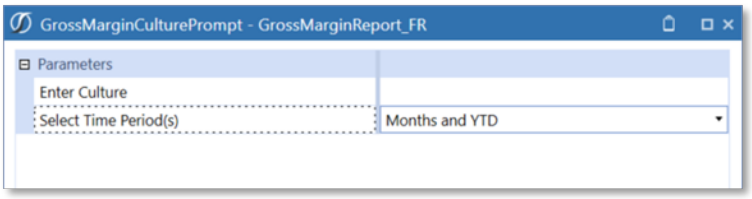
Below, the Parameter|!Enter Culture!| was entered in the Culture field.



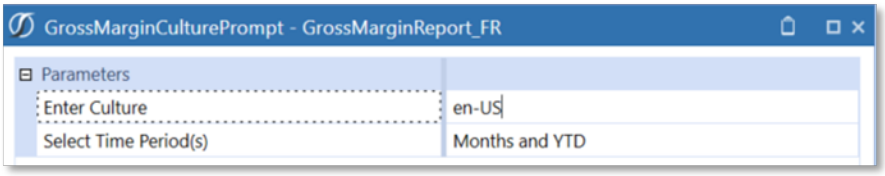
Run the cube view by clicking Open Data Explorer.



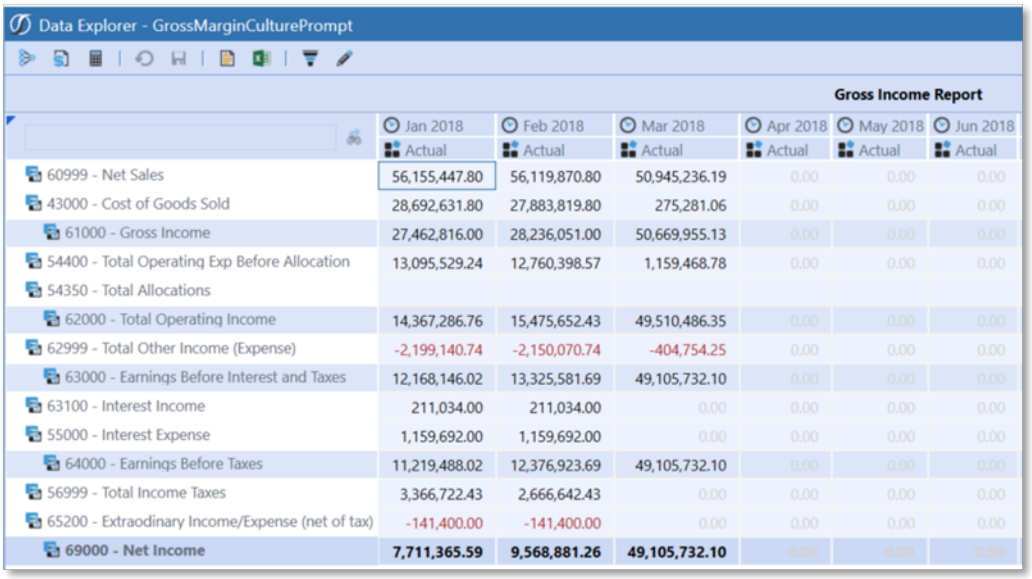
The user is prompted to enter the Culture, which they can type in. Alternatively, there could be a parameter set up under Application Dashboards which is a delimited list of applicable Cultures for this application.



The user can modify the Culture to reflect English by typing in “en-US”.



This results in a cube view displayed in English accounts.



	Jan 2018 Actual	Feb 2018 Actual	Mar 2018 Actual	Apr 2018 Actual	May 2018 Actual	Jun 2018 Actual
60999 - Net Sales	56,155,447.80	56,119,870.80	50,945,236.19	0.00	0.00	0.00
43000 - Cost of Goods Sold	28,692,631.80	27,883,819.80	275,281.06	0.00	0.00	0.00
61000 - Gross Income	27,462,816.00	28,236,051.00	50,669,955.13	0.00	0.00	0.00
54400 - Total Operating Exp Before Allocation	13,095,529.24	12,760,398.57	1,159,468.78	0.00	0.00	0.00
54350 - Total Allocations						
62000 - Total Operating Income	14,367,286.76	15,475,652.43	49,510,486.35	0.00	0.00	0.00
62999 - Total Other Income (Expense)	-2,199,140.74	-2,150,070.74	-404,754.25	0.00	0.00	0.00
63000 - Earnings Before Interest and Taxes	12,168,146.02	13,325,581.69	49,105,732.10	0.00	0.00	0.00
63100 - Interest Income	211,034.00	211,034.00	0.00	0.00	0.00	0.00
55000 - Interest Expense	1,159,692.00	1,159,692.00	0.00	0.00	0.00	0.00
64000 - Earnings Before Taxes	11,219,488.02	12,376,923.69	49,105,732.10	0.00	0.00	0.00
56999 - Total Income Taxes	3,366,722.43	2,666,642.43	0.00	0.00	0.00	0.00
65200 - Extraodinary Income/Expense (net of tax)	-141,400.00	-141,400.00	0.00	0.00	0.00	0.00
69000 - Net Income	**7,711,365.59**	**9,568,881.26**	**49,105,732.10**	0.00	0.00	0.00

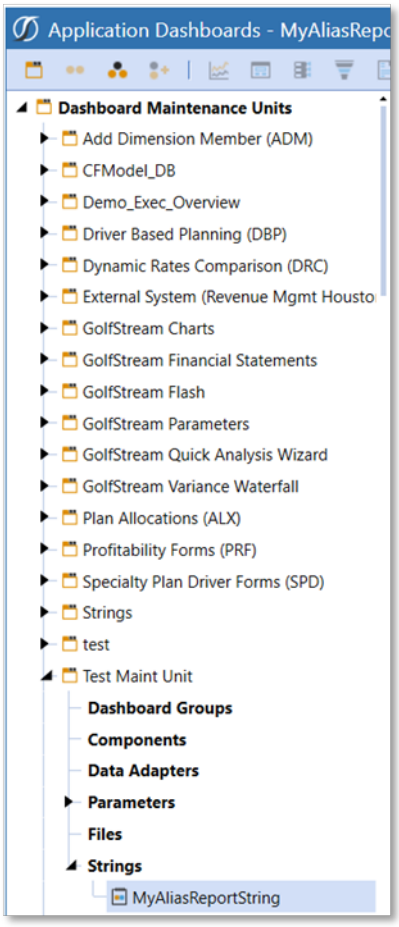
Alias for Headings: Strings within Dashboard Maintenance Unit

Strings are setup as an object type under a Dashboard Maintenance Unit. These strings can be used with XFString functionality and displayed in a cube view as a Caption or Heading in a cube view.

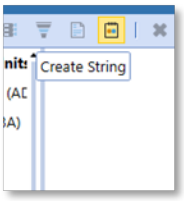
Reference Alias via XFString

You can use Alias strings with the XFString function to display page captions in another designated culture.

Example: On the Application tab, select Dashboards under the Presentation. In the Dashboard Maintenance Units section, we selected “Test Maint Unit” group.



Under the Strings section we used the “Create String” button:



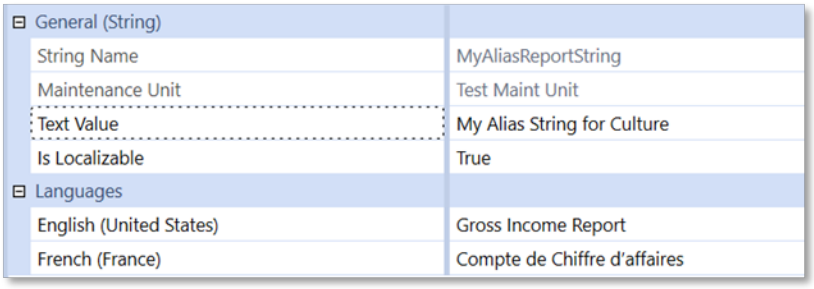
This adds a new String called MyAliasReportingString. This will be used in our Cube View example to display the Page Caption of the Cube View report in multiple languages based on the User's default culture setting and via a parameter to prompt the user for a specific culture to render that respective language.

Here are the languages in the string:

English (United States): Gross Income Report

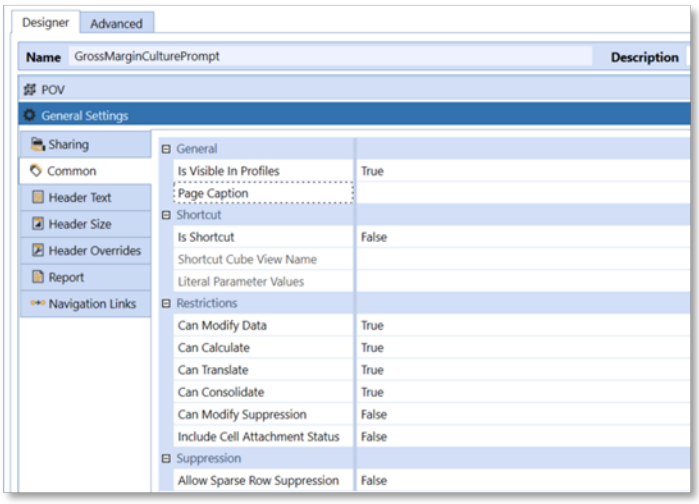
French (France): Compte de Chiffre d'affaires

From the General (String), set the Is Localizable value to True

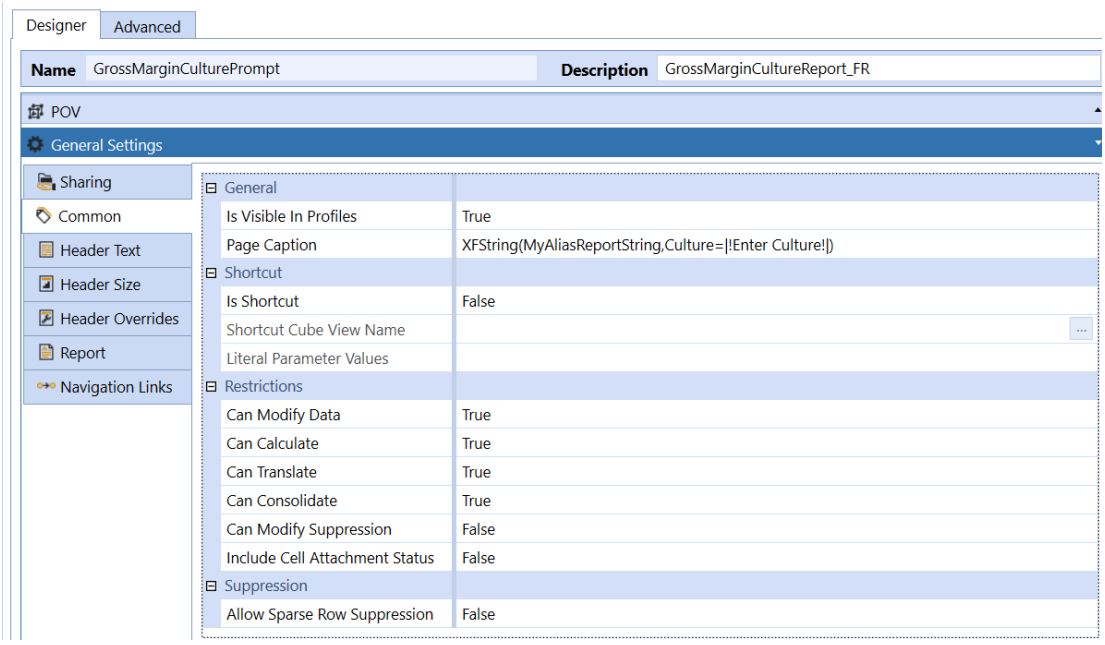


General (String)	
String Name	MyAliasReportString
Maintenance Unit	Test Maint Unit
Text Value	My Alias String for Culture
Is Localizable	True
Languages	
English (United States)	Gross Income Report
French (France)	Compte de Chiffre d'affaires

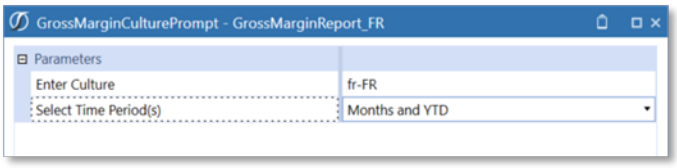
On the Designer tab under Cube View > Common > General, go to the Page Caption section. This section contains an updated with an XFString function to call a string from the Dashboard Maintenance Units.



Type in “XFString(MyAliasReportString, Culture=|!Enter Culture!|)” to the Page Caption field and select the Save button. This formula will call the String, “MyAliasReportString.”

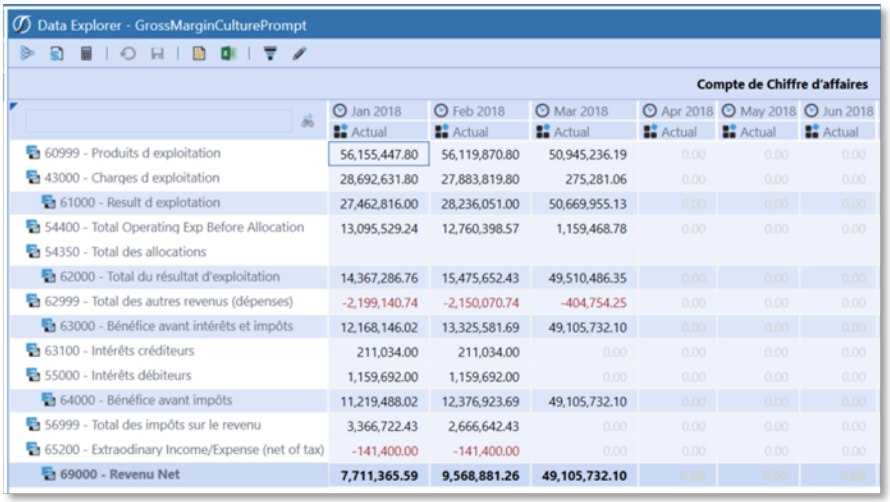


Using a parameter within the Page Caption in this instance will prompt the user to type in the culture of the language they wish the report caption to render.



The results shown below reflect the page caption of “Gross Income Report” in French:

“Compte de Chiffre d’affaires”



	Jan 2018 Actual	Feb 2018 Actual	Mar 2018 Actual	Apr 2018 Actual	May 2018 Actual	Jun 2018 Actual
60999 - Produits d exploitation	56,155,447.80	56,119,870.80	50,945,236.19	0.00	0.00	0.00
43000 - Charges d exploitation	28,692,631.80	27,883,819.80	275,281.06	0.00	0.00	0.00
61000 - Result d explotation	27,462,816.00	28,236,051.00	50,669,955.13	0.00	0.00	0.00
54400 - Total Operating Exp Before Allocation	13,095,529.24	12,760,398.57	1,159,468.78	0.00	0.00	0.00
54350 - Total des allocations						
62000 - Total du résultat d'exploitation	14,367,286.76	15,475,652.43	49,510,486.35	0.00	0.00	0.00
62999 - Total des autres revenus (dépenses)	-2,199,140.74	-2,150,070.74	-404,754.25	0.00	0.00	0.00
63000 - Bénéfice avant intérêts et impôts	12,168,146.02	13,325,581.69	49,105,732.10	0.00	0.00	0.00
63100 - Intérêts créditeurs	211,034.00	211,034.00	0.00	0.00	0.00	0.00
55000 - Intérêts débiteurs	1,159,692.00	1,159,692.00	0.00	0.00	0.00	0.00
64000 - Bénéfice avant impôts	11,219,488.02	12,376,923.69	49,105,732.10	0.00	0.00	0.00
56999 - Total des impôts sur le revenu	3,366,722.43	2,666,642.43	0.00	0.00	0.00	0.00
65200 - Extraodinary Income/Expense (net of tax)	-141,400.00	-141,400.00	0.00	0.00	0.00	0.00
69000 - Revenu Net	**7,711,365.59**	**9,568,881.26**	**49,105,732.10**	0.00	0.00	0.00

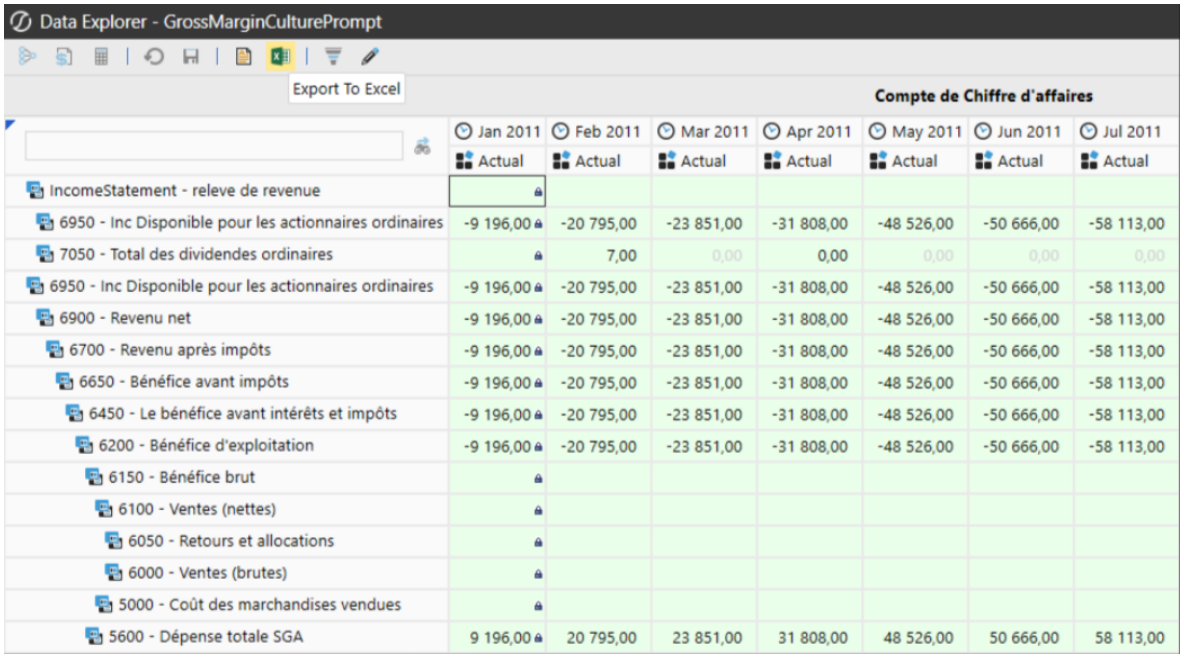
Report Alias (Displaying Alternate Descriptions for Members)

Report Alias functionality can make an end user’s reporting analysis or forms entry easier by rendering headings and headers in other languages at runtime so that the user can see these objects in their respective language based on the user’s Culture Property setting. Displaying alias descriptions for Members shown in the Headers of Cube Views as well as alias descriptions for Cube View and Report Headings is supported.

Report Alias with Spreadsheet and Excel

If alias properties are populated, the result is translated or alias member names are rendered in the Spreadsheet feature and Excel Add-in from Quick Views, exported Cube Views and using Cube View Connections.

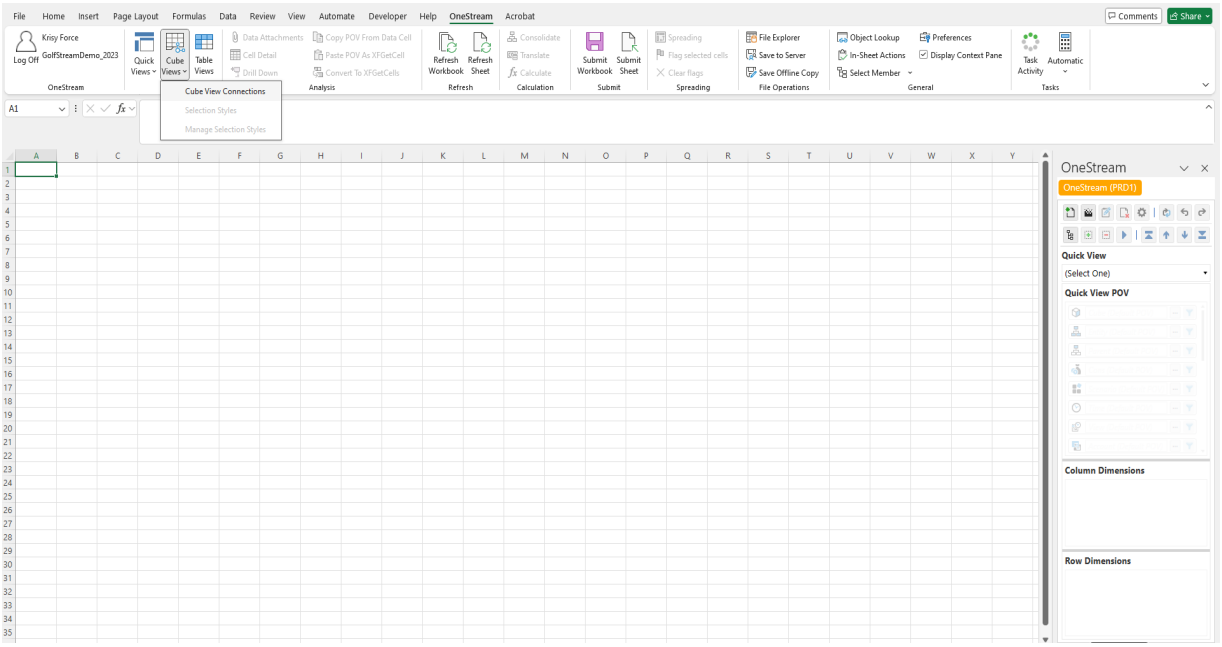
For example, in the Cube View via Data Explorer, click the **Export To Excel** button.



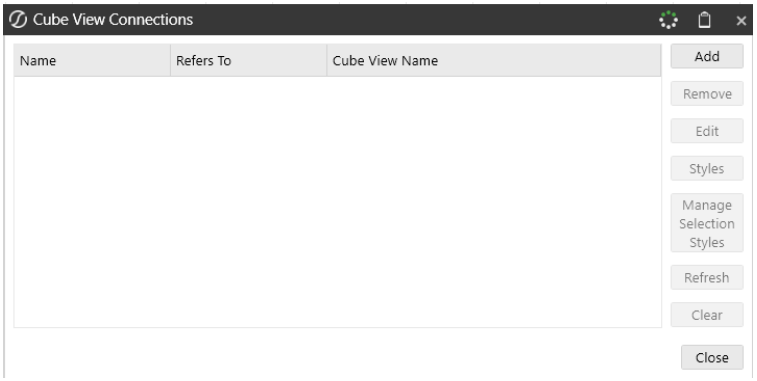
This will render the Cube View in Excel in the respective language.

	A	B	C	D	E	F	G	H
1							GrossMarginCultureReport_FR	
2								
3		Jan 2018	Feb 2018	Mar 2018	Apr 2018	May 2018	Jun 2018	Jul 2018
4		Actual	Actual	Actual	Actual	Actual	Actual	Actual
5	60999 - Produits d exploitation	56,155,447.80	56,119,870.80	50,945,236.19	0.00	0.00	0.00	
6	43000 - Charges d exploitation	28,692,631.80	27,883,819.80	275,281.06	0.00	0.00	0.00	
7	61000 - Result d exploitation	27,462,816.00	28,236,051.00	50,669,955.13	0.00	0.00	0.00	
8	54400 - Total Operating Exp Before Allocation	13,095,529.24	12,760,398.57	1,159,468.78	0.00	0.00	0.00	
9	54350 - Total des allocations							
10	62000 - Total du résultat d'exploitation	14,367,286.76	15,475,652.43	49,510,486.35	0.00	0.00	0.00	
11	62999 - Total des autres revenus (dépenses)	-2,199,140.74	-2,150,070.74	-404,754.25	0.00	0.00	0.00	
12	63000 - Bénéfice avant intérêts et impôts	12,168,146.02	13,325,581.69	49,105,732.10	0.00	0.00	0.00	
13	63100 - Intérêts créditeurs	211,034.00	211,034.00	0.00	0.00	0.00	0.00	
14	55000 - Intérêts débiteurs	1,159,692.00	1,159,692.00	0.00	0.00	0.00	0.00	
15	64000 - Bénéfice avant impôts	11,219,488.02	12,376,923.69	49,105,732.10	0.00	0.00	0.00	
16	56999 - Total des impôts sur le revenu	3,366,722.43	2,666,642.43	0.00	0.00	0.00	0.00	
17	65200 - Extraodinary Income/Expense (net of tax)	-141,400.00	-141,400.00	0.00	0.00	0.00	0.00	
18	69000 - Revenu Net	7,711,365.59	9,568,881.26	49,105,732.10	0.00	0.00	0.00	

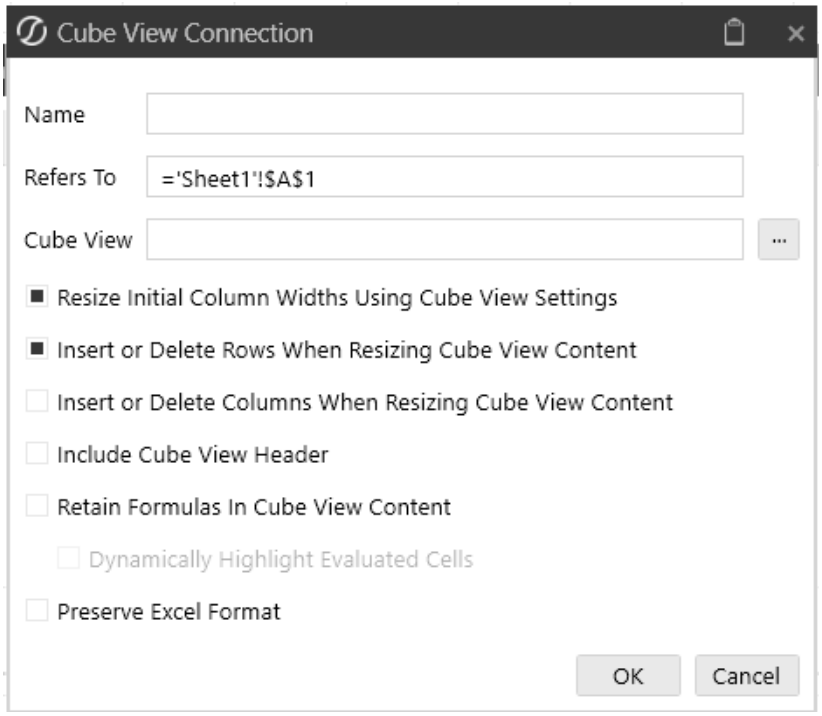
Using Cube View Connections in Spreadsheet and Excel Add-in will also render the Cube View in the language that is established. Open a new or existing Spreadsheet and select **Cube Views** > **Cube View Connections** in the OneStream menu.



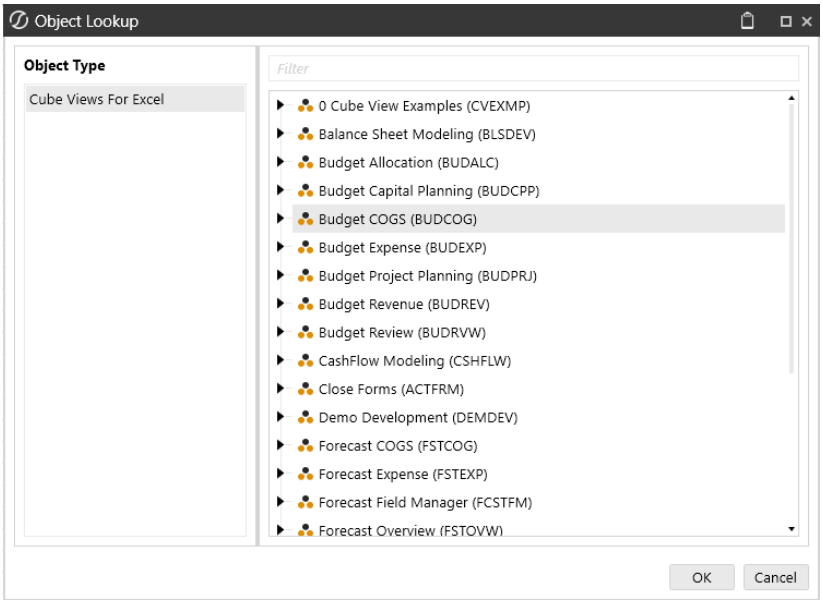
Click **Add** in the **Cube View Connections** dialog box.



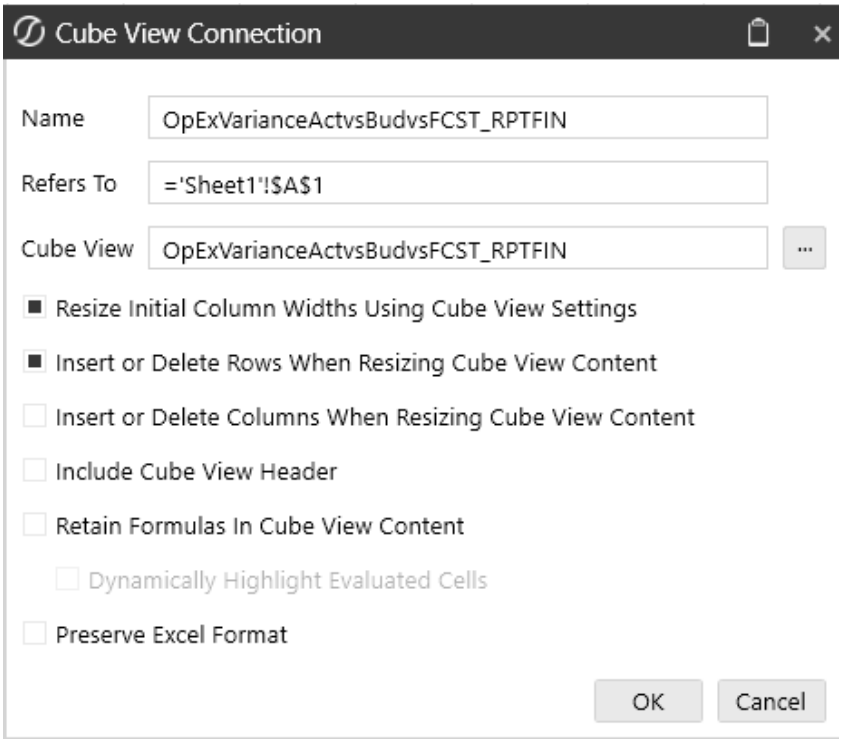
Click the ellipsis in the next Cube View Connection dialog box.



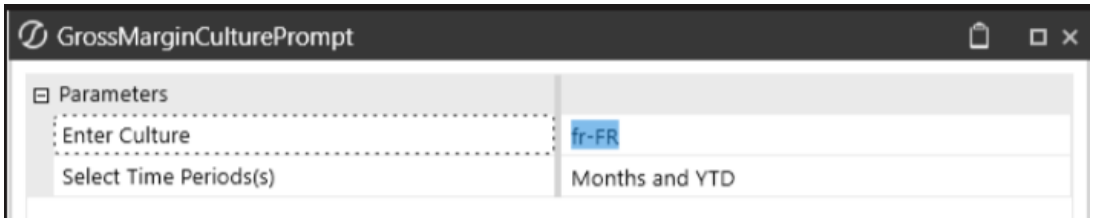
Search for your Cube View (Object Type) and then click the **OK** button.



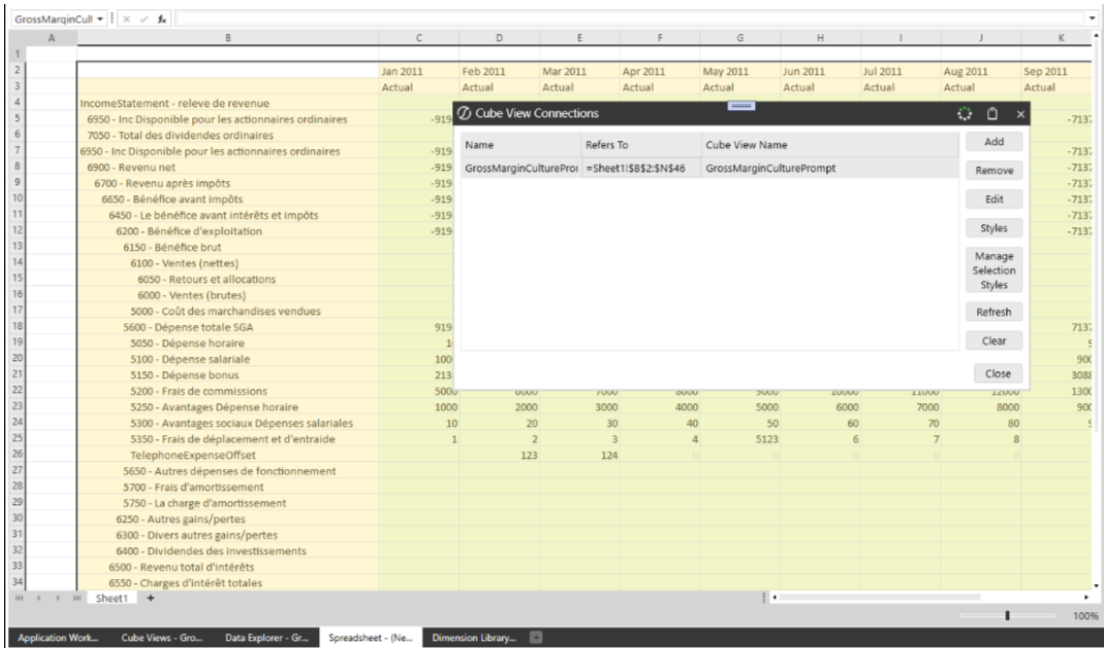
In the Cube View Connection dialog box, check your settings and click **OK**.



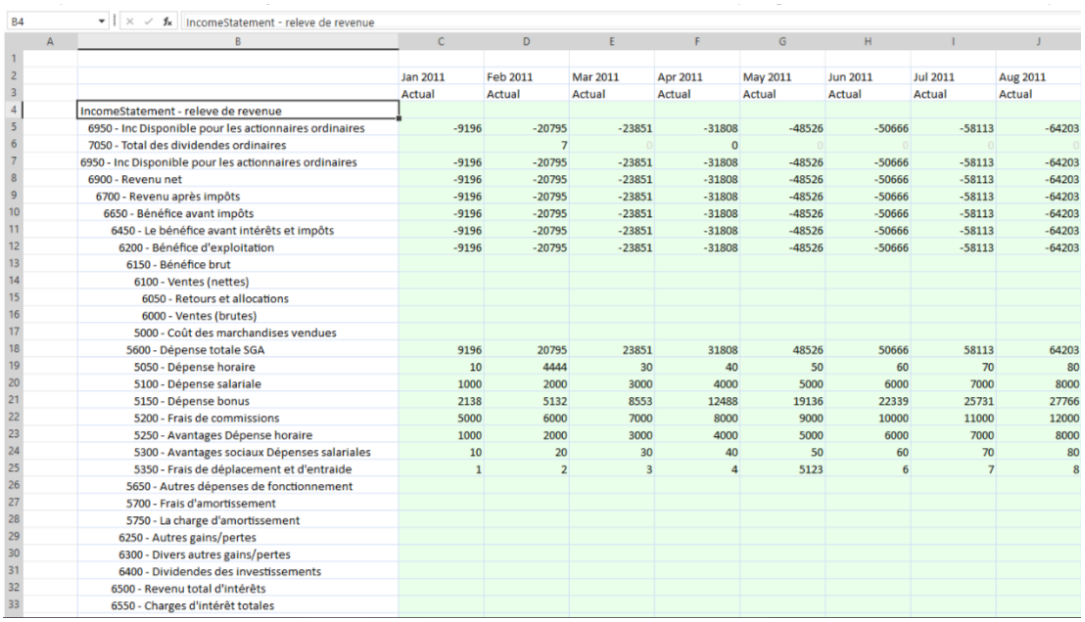
The Cube View selected contains a prompt for Culture and Time Periods. Enter those accordingly and click **OK**.



This Cube View is now added. Click **Close** to view the results.



The results of the Cube View connection with the Language updates (Report Alias) in the Cube View.



B4 | IncomeStatement - releve de revenue

	A	B	C	D	E	F	G	H	I	J
1										
2			Jan 2011	Feb 2011	Mar 2011	Apr 2011	May 2011	Jun 2011	Jul 2011	Aug 2011
3			Actual	Actual	Actual	Actual	Actual	Actual	Actual	Actual
4		IncomeStatement - releve de revenue								
5		6950 - Inc Disponible pour les actionnaires ordinaires	-9196	-20795	-23851	-31808	-48526	-50666	-58113	-64203
6		7050 - Total des dividendes ordinaires		7	0	0	0	0	0	0
7		6950 - Inc Disponible pour les actionnaires ordinaires	-9196	-20795	-23851	-31808	-48526	-50666	-58113	-64203
8		6900 - Revenu net	-9196	-20795	-23851	-31808	-48526	-50666	-58113	-64203
9		6700 - Revenu après impôts	-9196	-20795	-23851	-31808	-48526	-50666	-58113	-64203
10		6650 - Bénéfice avant impôts	-9196	-20795	-23851	-31808	-48526	-50666	-58113	-64203
11		6450 - Le bénéfice avant intérêts et impôts	-9196	-20795	-23851	-31808	-48526	-50666	-58113	-64203
12		6200 - Bénéfice d'exploitation	-9196	-20795	-23851	-31808	-48526	-50666	-58113	-64203
13		6150 - Bénéfice brut								
14		6100 - Ventes (nettes)								
15		6050 - Retours et allocations								
16		6000 - Ventes (brutes)								
17		5000 - Coût des marchandises vendues								
18		5600 - Dépense totale SGA	9196	20795	23851	31808	48526	50666	58113	64203
19		5050 - Dépense horaire	10	4444	30	40	50	60	70	80
20		5100 - Dépense salariale	1000	2000	3000	4000	5000	6000	7000	8000
21		5150 - Dépense bonus	2138	5132	8553	12488	19136	22339	25731	27766
22		5200 - Frais de commissions	5000	6000	7000	8000	9000	10000	11000	12000
23		5250 - Avantages Dépense horaire	1000	2000	3000	4000	5000	6000	7000	8000
24		5300 - Avantages sociaux Dépenses salariales	10	20	30	40	50	60	70	80
25		5350 - Frais de déplacement et d'entraide	1	2	3	4	5123	6	7	8
26		5650 - Autres dépenses de fonctionnement								
27		5700 - Frais d'amortissement								
28		5750 - La charge d'amortissement								
29		6250 - Autres gains/pertes								
30		6300 - Divers autres gains/pertes								
31		6400 - Dividendes des investissements								
32		6500 - Revenu total d'intérêts								
33		6550 - Charges d'intérêt totales								

Workspaces

A workspace is a framework for building software using software, creating a robust environment for developing products on the platform. It simplifies the development process and extends development capabilities for solution creators.

Workspaces store maintenance units and facilitate community development by providing an isolated environment for developers to segregate and organize solution objects. Maintenance units are stored, created, and maintained in workspaces, which vary by dashboard project need and application.

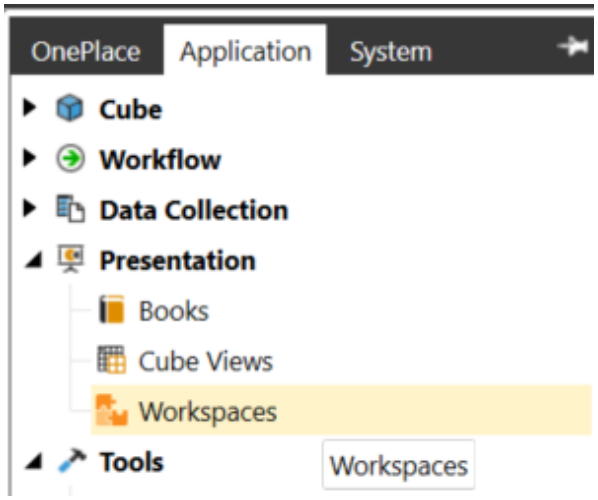
Benefits

Workspaces provide an isolated environment in which solution developers and creators can develop multiple solutions to solve complex business processes. Workspaces provide the following benefits:

1. Isolation between workspaces, which allows developers to work on the same solution or dashboard in a sandbox-like environment.
2. Greater flexibility among developers and other team members when testing, making changes, and planning.
3. Maintenance Units, along with their objects, can have the same names in separate workspaces and do not need to be renamed. This reduces the likelihood of naming conflicts especially when importing and exporting objects from other applications or sources.
4. You can selectively share workspace objects such as embedded dashboards, parameters, file resources, and string resources with other workspaces. This lets you reuse objects rather than copying them.
5. Workspace objects can have the same names in different workspaces.
6. Sets the foundation for future functionality and ongoing development.
7. Product packaging mechanism for creating, deploying, and migrating solutions.

The Workspaces Page

The Workspaces page, found on the Application tab and the System tab, is used to create application and system workspaces. The pages look similar between the two tabs.



Security Roles

The following security roles and locations are necessary when creating workspaces.

Role	Location	Property
Application Security	**Application** > **Tools** > **Security Roles**	ManageApplicationWorkspaces
Application User Interface	**Application** > **Tools** > **Security Roles**	WorkspaceAdminPage
System Security	**System** > **Administration** > **Security** > **System Security Roles**	ManageSystemWorkspaces
System User Interface	**System** > **Administration** > **Security** > **System Security Roles**	SystemWorkspaceAdminPage

Cube Views created prior to version 8.0 are accessible through the Cube View page and the Default Maintenance Unit in the Default Workspace.

The Access Group and Maintenance Group for these cube view groups can be edited if you are assigned to the ManageCubeViews Application security role. If cube views are created within the Workspaces page outside of the Default Maintenance Unit, you cannot edit the security of these cube view groups. Security for these cube views outside the Default Maintenance Unit is controlled by the maintenance unit in which the cube view resides.

With the Application security role, ManageApplicationWorkspaces provides full rights to edit anywhere outside the Default workspace and Default Maintenance Unit. To edit any assemblies, you must be a part of the AdministerApplicationWorkspaceAssemblies security role. This applies to all dashboard groups within the generic default workspace and all non-default workspaces. Maintenance Unit security determines access.

The Access Group and Maintenance Group properties for Data Management Groups are accessible through the Default Workspace and Default Maintenance Unit. They can be edited if you are assigned the ManageData security role along with the WorkspacesAdminPage role to access the Application Workspaces Page.

> **NOTE:** ManageData security role will override access at the Data Management Group security properties of Access Group and Maintenance Group.

Data Management Groups outside of the Default Workspace>Default Maintenance Unit will have the security properties of "Access Group" and "Maintenance Group" completely hidden from the user. Each Workspace and Maintenance Unit has their own Access Groups and Maintenance Group security properties. You will need access to the WorkspacesAdminPage user role to even access the Application Workspaces Page.

The Access Group and Maintenance Group properties for the Legacy Data Management page can be edited if you are assigned to the ManageData security role. You will need access to the DataManagementAdminPage to see legacy Data Management Groups located under **Application** > **Tools** > **Data Management**.

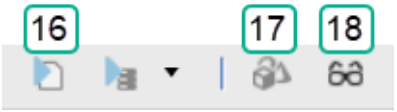
Toolbar Buttons

Several buttons on the Workspaces toolbar work with dashboards, cube views, and data management groups, depending on whether you are working with cube view groups and profiles, dashboard groups and profiles, or data management groups. Tooltips provide helpful hints and are accessed by hovering the cursor over the button.

1. **Create Group**: Create a new cube view group (if cursor is on the Cube View Groups icon), dashboard group (if on the Dashboard Groups text of a maintenance unit), or data management group (if cursor is on the Data Management icon).
2. **Create Profile**: Create a new cube view profile (if you have selected a cube view profile) or dashboard profile (if you have selected a dashboard profile).
3. **Manage Profile Members**: Displays the Profile Builder dialog box if you have selected either a cube view profile or a dashboard profile. If the cube view or dashboard is not in the Default workspace, a tooltip will display the workspace prefix when you hover the cursor over the group name.
4. **Create Cube View**: Create a new cube view when viewing cube view groups.
5. **Create Sequence**: Create a new sequence within data management groups.
6. **Create Step**: Create a new step within data management groups.

7. **Delete Selected Item**: Delete selected component item. You can select multiple cube views, cube view groups, and data management groups' steps and sequences. You can only delete empty Cube View Groups and multi-select more than one data management sequence and step to delete within the same data management group. Multiple cube views can only be selected and deleted if they are in the same Cube View Group. You cannot delete data management sequences or steps within different data management groups at the same time.
8. **Rename Selected Item**: Opens the Rename dialog box so that you can rename the selected item.
9. **Copy**: Makes a copy of the selected items. Select more than one item of the same type by pressing and holding the Ctrl button and clicking each item with the mouse. You can also select the items to copy then right-click to open a context menu. You can only copy items of the same type.
10. **Paste**: Pastes copied items into the selected destination. "_Copy" is appended to the name of the newly pasted item.
11. **Search**: Search for a component using the Select Workspace Object dialog box. Cube views and data management sequences and steps are added to the list of searchable objects.

12. **Plus Sign**: Adds a data management group sequence or step, cube view row or column, or dashboard component.
13. **Move Up**: Moves up a data management group sequence or step, cube view row or column, or dashboard component.
14. **Move Down**: Moves down a data management group sequence or step, cube view row or column, dashboard component.
15. **Minus Sign**: Removes a data management group sequence or step, cube view row or column, or dashboard component.



16. **Open Data Explorer or Run**: Allows you to view the data explorer grid of the selected cube view. Results will display in a separate tab. Also, allows you to run a data management sequence or step.
17. **Show Objects That Reference the Selected Item**: Opens the Object Usage Viewer and lists all objects that reference the selected item.
18. **Object Lookup**: Opens the Object Lookup dialog box letting you search for any object in the application, such as all workspaces that contain data management sequences and any other object.

Workspaces Filter

You can filter workspaces for more control and flexibility to search and browse within the Application Workspace page and System Workspaces page.

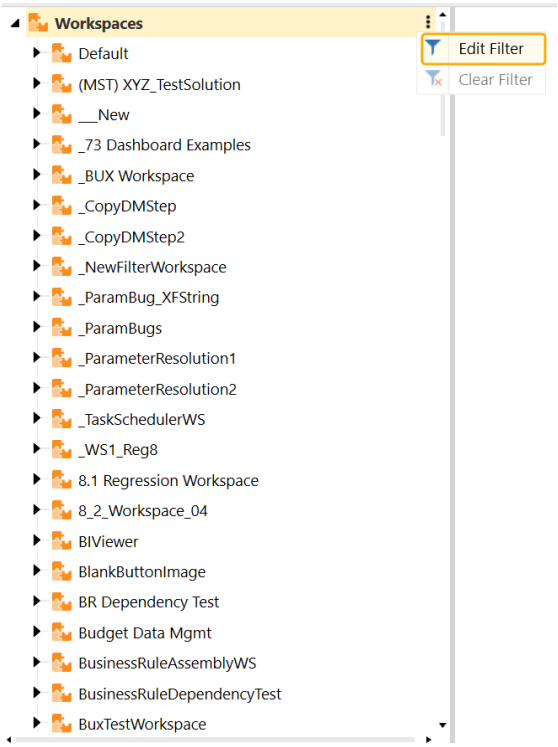
Edit Filter

Use Edit Filter to quickly and easily find what you need in the Workspaces tree hierarchy.

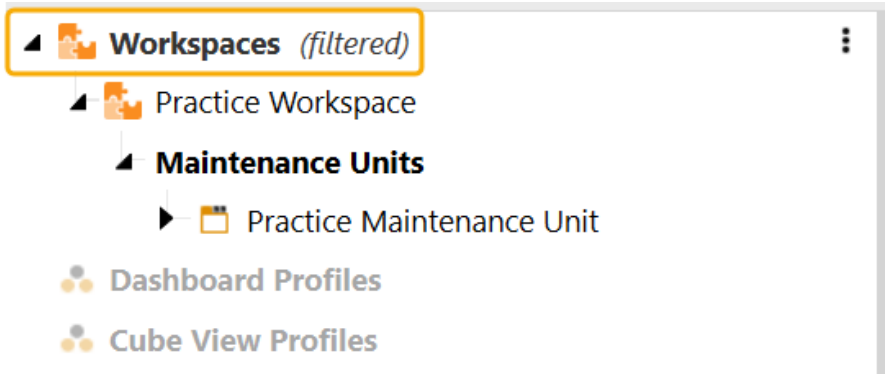
1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. In the**Application Workspaces**pane, to the right of**Workspaces,** click the **vertical kebab (⋮),** and select**Edit Filter.**
3. Type keywords in the **Search** bar to search the workspace name and maintenance unit. You must select the **Search Binocular**icon to initiate search. The tree structure will update based on your search. If a workspace has multiple maintenance units, it will display on your search. Select the **Maintenance Units** to load and display all children objects.

   > **NOTE:** The default state for the Workspace filter will have everything selected, such as All, All workspaces, and All associated maintenance units.

4. Select your workspaces and maintenance units.
5. Click **OK** button.



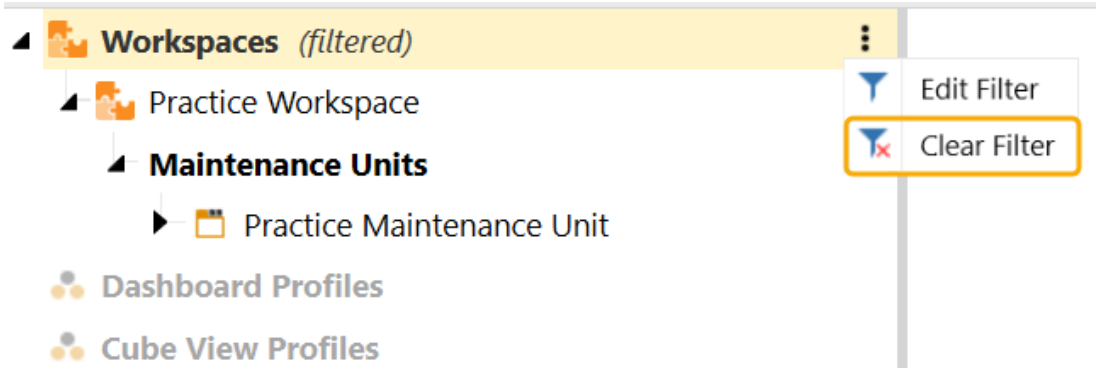
If you select any workspace and maintenance unit besides All, the Application Workspaces page will display **(filtered)** to let you know there is an active filter. Whenever a workspace filter is on, Dashboard Profiles and Cube View Profiles are grayed out. If you select All, the workspace will not display a filter.



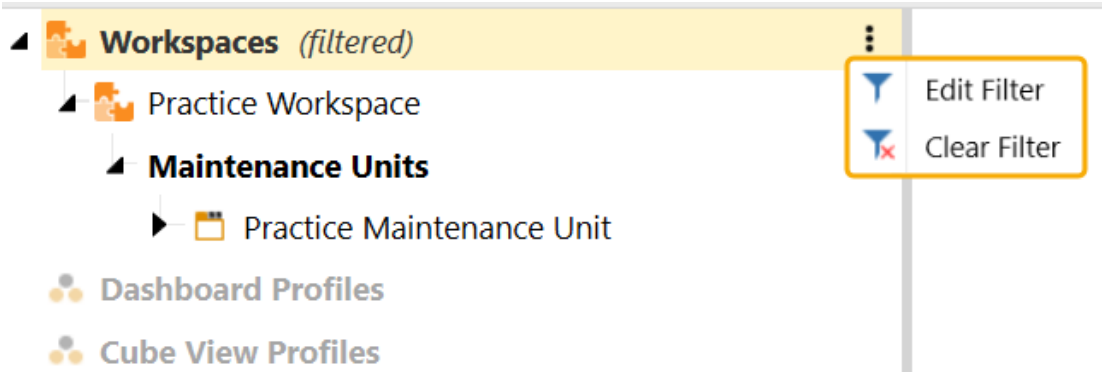
Clear Filter

Use Clear Filter to clear any previously selected and filtered workspace or maintenance unit.

1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. In the**Application Workspaces**pane, to the right of**Workspaces,** click the **vertical kebab (⋮),** and select **Clear Filter.**



NOTE: If you previously selected a particular workspace or maintenance unit, you have the option to select the **Edit Filter** to continue filtering or select **Clear Filter** to clear all filters.



Save Workspaces Filters

Once you have a filter saved on a particular workspace and the associated maintenance units, you will be able to log off and log back into OneStream and the filter will be saved per application. OneStream will save the state of the new workspaces filter unless you select **Clear Filter**.

NOTE: You can also edit, clear, and save workspaces filters on the **System** tab, under **Administration**> **Workspaces**.

Workspaces Properties

Workspaces are defined and function according to the following conventions.

Naming Conventions

All maintenance unit objects can have the same name as long as they are in separate workspaces.

Default Workspace
A system generated workspace where you can create solutions. All existing maintenance units prior to version 7.3 were moved to the Default workspace.

- The Default workspace is always shared and is accessible by any other workspace, even if not explicitly added to the Shared Workspaces list.
- Referencing parameters do not consider workspaces and backward compatibility is enabled.

> **NOTE:** The Default Maintenance Unit only exists on the Application Workspace page.

Properties

The following properties help to define a workspace.

General

Name
Name of a workspace.

Description
Brief description of a workspace.

Notes
Brief notes to add to a workspace.

Substitution Variable Items
Field to define substitution variables to be used within the workspace. Select the ellipses (...) to view, edit, or delete substitution variable items.

> **NOTE:** (Collection) will always be displayed for this property.

Security

Access Group

Manages the users that have access to the workspace.

Maintenance Group

Manages the users that have access to maintain and administer the workspace.

Sharing

Sharing allows developers to selectively share workspace items such as dashboards, parameters, components, and adapters across workspaces when necessary. You should implement all related items in the same workspace to minimize sharing of items from other workspaces. Sharing items across workspaces provides opportunities for re-use of items rather than copying them.

Is Shareable Workspace

If this property is set to True, other workspaces can reference objects in the current workspace.

> **NOTE:** To use Cube Views or Nested parameters within Spreadsheet or Excel AddIn, verify that the Workspaces containing those objects are set to Shareable.

The Default workspace is predefined by OneStream and the Is Shareable Workspace property is always set to True. All other workspaces can reference dashboards and parameters implemented in the Default workspace.

> **NOTE:** The Default workspace is shareable by default and cannot be changed.

The Default workspace also behaves as if all other workspaces are part of its Shared Workspace Names list. Therefore, any dashboard or parameter defined in another shareable workspace can be referenced by items in the Default workspace. This approach simplifies backward compatibility since new workspaces can access all items that previously existed in the Default workspace. However, this requires an implementor to avoid using similar names for items in the Default and custom workspaces.

Shared Workspace Names
This property indicates the workspace names for use in the current workspace. This is a comma-separated (no spaces) list of names that can be used to find embedded dashboards, parameters, file resources, string resources, and so forth while processing a dashboard in this workspace.

Referenced workspaces must have the Is Shareable Workspace property set to **True**. The search order for a requested item name is the current workspace followed by the Default workspace, then followed by the list of Shared Workspace Names in the order they were entered.

Assemblies

Namespace Prefix
Specify different names used for referencing assemblies in a workspace.

Imports Namespace 1-8
Corresponds to the _ImportsNamespace1 placeholder value within the import statements in code.

Workspace Assembly Service
Type a name for the workspace assembly service factory used to process the dynamic creation of dashboards within this workspace. This property must be configured for the service type assembly files to be used as intended. The syntax for this property is AssemblyName.AssemblyFileName.

Text

Text 1-8
Create string values.

Search Order

Sharing items across workspaces requires a defined search order when the same item name is used more than once. The following is the order of operations:

1. Search the workspace you are currently working in.
2. Search the Default workspace.
3. Search the Shared Workspace Names list.

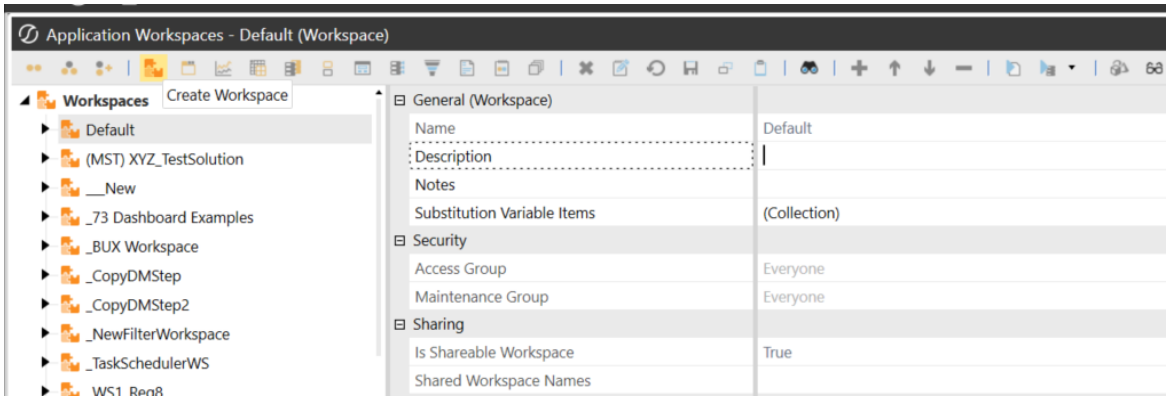
> **NOTE:** If all items are defined in the Default workspace, search order is irrelevant.

Workspaces Setup

Maintenance units created prior to version 7.4 can be found in the Default workspace.

To create a new workspace:

1. In OneStream, on the **Application** tab, click **Workspaces** under the **Presentation** section.
2. Select **Workspaces**.
3. Click **Create Workspace** and name the new workspace.



Workspace Assemblies

Assemblies are integrated into workspaces to give you the freedom to write logic exactly where you need it. Here, you can write inline business rules using the Assemblies node within a given maintenance unit.

This applies to dashboard-specific business rules only and consists of the following types:

- Cube View Extender
- XFBRString
- Dashboard DataSet
- Dashboard Extender
- Spreadsheet
- Finance Business Rules

Workspace assemblies cater to the OneStream developer community for those who build solutions and create dashboards. They are similar to the Visual Studio product. Developers can add folders and within those folders, they can create any number of C# or VB.Net files depending on the Compiler Language specified in the Assembly Properties tab. These files are then compiled into a single assembly.

The benefits of workspace assemblies extend to OneStream developers, customers, and partners. There are also a variety of service types that assist with the development of Solutions.

Because of the power and flexibility this feature extends to those who use assemblies, it is important to:

- Inform developers on how to take advantage of assemblies
- Recognize evolving recommendations on using assemblies
- Highlight key areas of assemblies to use within the broader community

> **IMPORTANT:** The compiler language field will default to C# when creating an Application or System Workspace Assembly.

OneStream Developers

Developers include:

- **MarketPlace Developers**: Engineers creating solutions that reside within the OneStream Solution Exchange for customers and partners supported by OneStream.
- **Solution Exchange Developers**: Anyone in the OneStream community who designs solutions featured in Partner Solutions or Community Solutions.
- **PreSales**: Sales engineers who create demonstrations and product proof of concept.
- **Consultants**: Product implementers building custom solutions for clients.
- **Advanced Application Solutions**: Technical team building advanced solutions for customers.

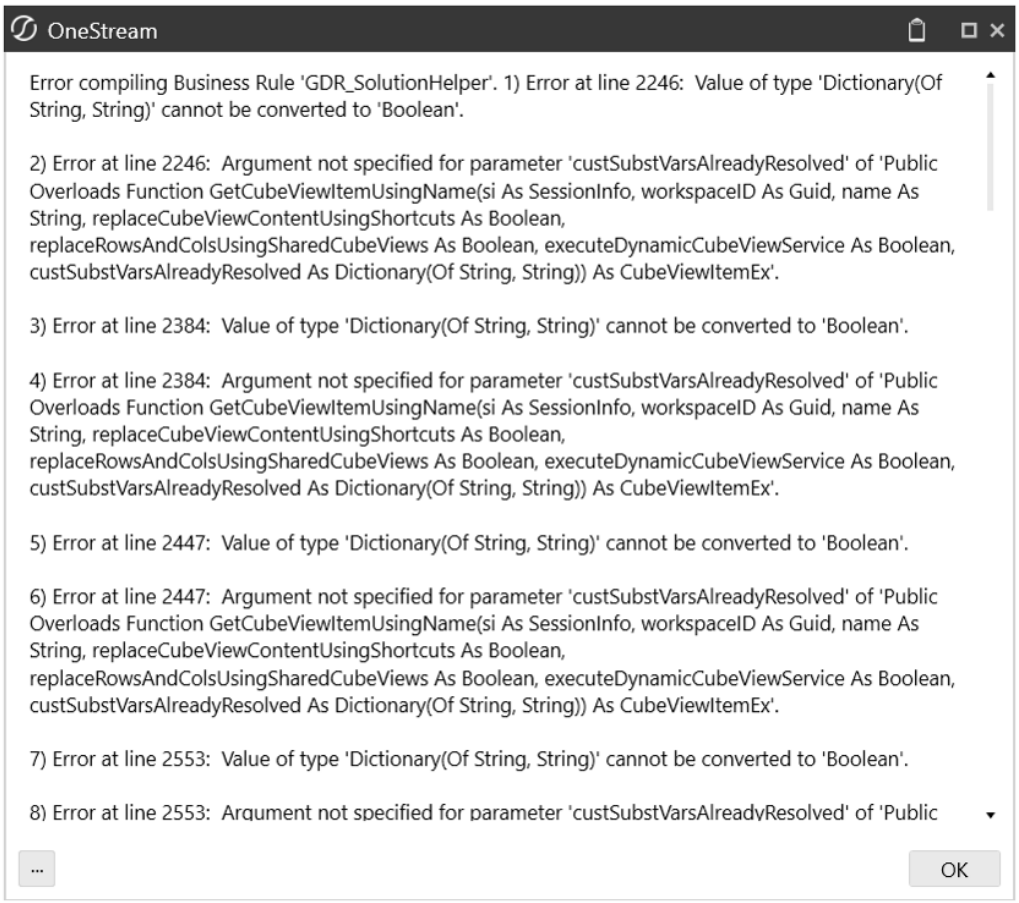
Assembly Encryption

Assembly files can be encrypted. You must be a member of the EncryptBusinessRules security role group. You must be a member of the EncryptBusinessRules security role. Within workspaces assemblies, you can create rules using either C# or VB.Net. Right-click on the file name and select Encrypt File. You will see the Encrypt File dialog box asking you to create a password using the same legacy rules. You can also decrypt files by right-clicking on the encrypted file, selecting Decrypt File, and then entering the previously created password.

Considerations

When calling business rules that reference the GetCubeViewItemUsingName and GetCubeViewItem methods, the methods needs to be updated to include this parameter: bool executeDynamicCubeViewService.

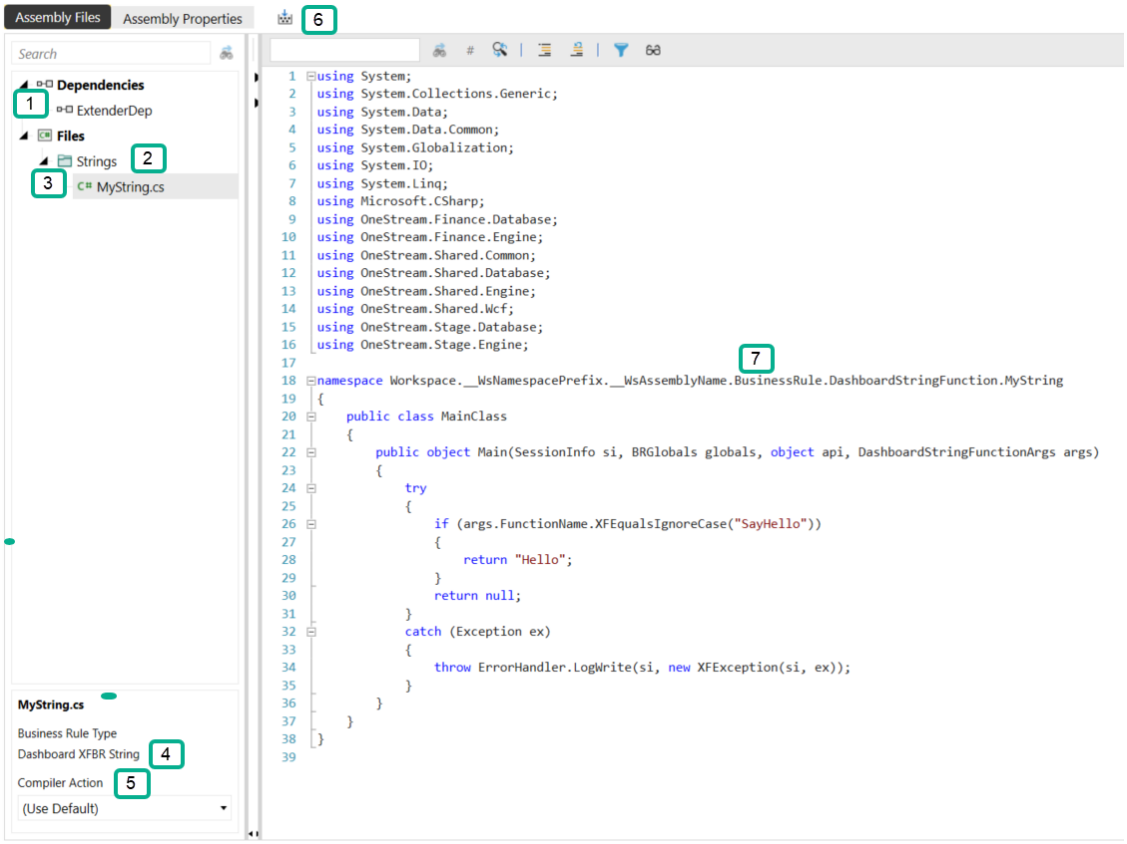
If this is not added to the methods, you will receive an error.



Get Started With Assemblies

Assemblies are structured similarly to Visual Studio where you can build assemblies using files and folders. This workspace page is also similar to the Business Rules page in OneStream, allowing you to quickly create assemblies as needed.

The following image of a workspaces page shows the various components you will need to build an assembly.



1. **Dependencies**: Reference other Workspaces assemblies, Business Rules, and Prepackaged assemblies from different workspaces. Right-click on **Dependencies** to create.

2. **Folders**: Organize assembly files. Right-click on **Files** and select **Add Folder**to create a folder.
3. **Files**: Written rules for the assemblies source code files, such as business rules, classes, interfaces, and so on. Right-click on **Files** to create a file.
4. **Business Rule Type**: Type of business rule or service type. Decided upon during creation and cannot be changed after the file is created. This item will say Not Used if you have selected a service type rather than a business rule type.
5. **Compiler Action**: Action to perform on the file. Can be Use Default, Disabled, or Compile (this is the same as Use Default). If you select Disabled, you cannot compile the assembly to verify syntax. This value can be changed after creating the assembly file.
6. **Compile Icon**: Compiles all files in the assembly when clicked.
7. **Namespace**: The Namespace of any given file will look different than what you would see on the Business Rules page. The filename, in this example, MyString, is the last item in the list.

Create an Assembly

Assemblies work similarly to Visual Studio Code, where there is a project with folders, each containing dependencies, such as assemblies and business rules. Assemblies let developers add references to known packages that are part of the standard server install, such as OpenXML. In the OneStream application, assemblies are located within Maintenance Units.

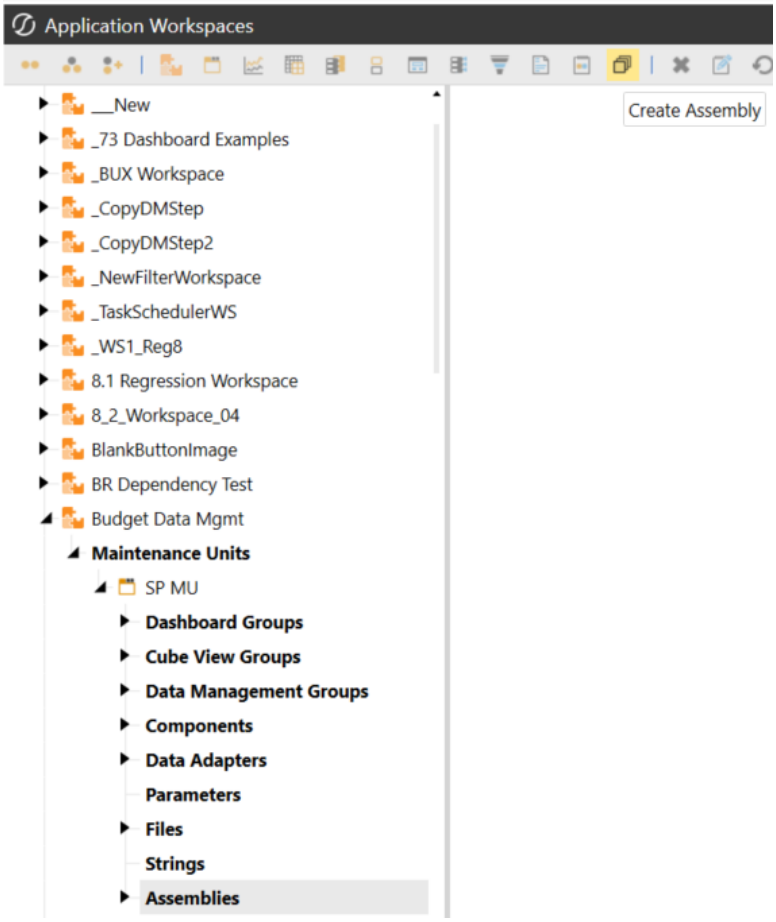
> **NOTE:** If you need to reference an assembly from a business rule, use the following syntax: WS\ followed by the long workspace assembly name. For example, WS\Workspace.wsName.wsAssemblyName.

Follow these steps to create an assembly. In this example, you will use VB.Net.

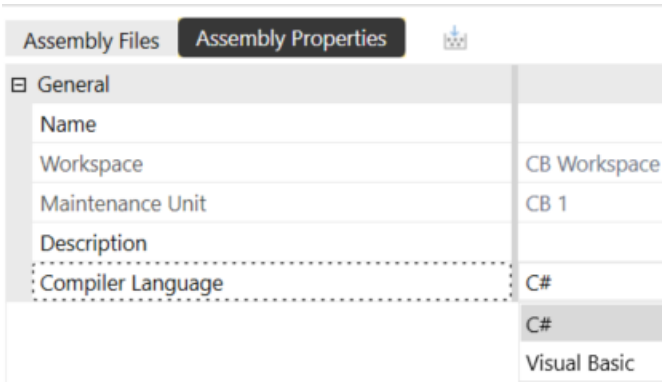
1. On the Application tab, click **Workspaces**.
2. Expand **Workspaces** and locate the workspace.
3. Expand the workspace, then expand **Maintenance Units**.
4. Expand the appropriate maintenance unit and click the **Assemblies** label.

   You will not see this in the default maintenance unit, which is located in the Default workspace. This maintenance unit only stores cube view groups.

5. Click **Create Assembly** on the toolbar. Assemblies can be C# or VB.Net.



6. On the Assembly Properties page, click the drop-down arrow in the **Compiler Language** field and select the type of assembly. It can be either C# or VB.Net.



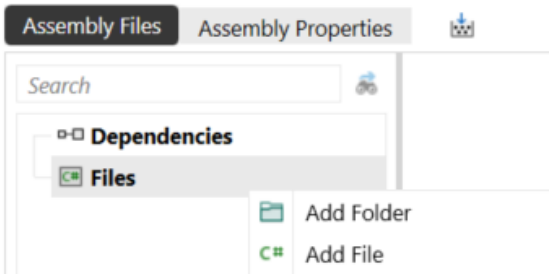
7. Give the assembly a name.
8. Optional. If you need to delete an assembly, while on the Assembly Properties page, click **Delete**.
9. Click **Save**.

You have now created an assembly.

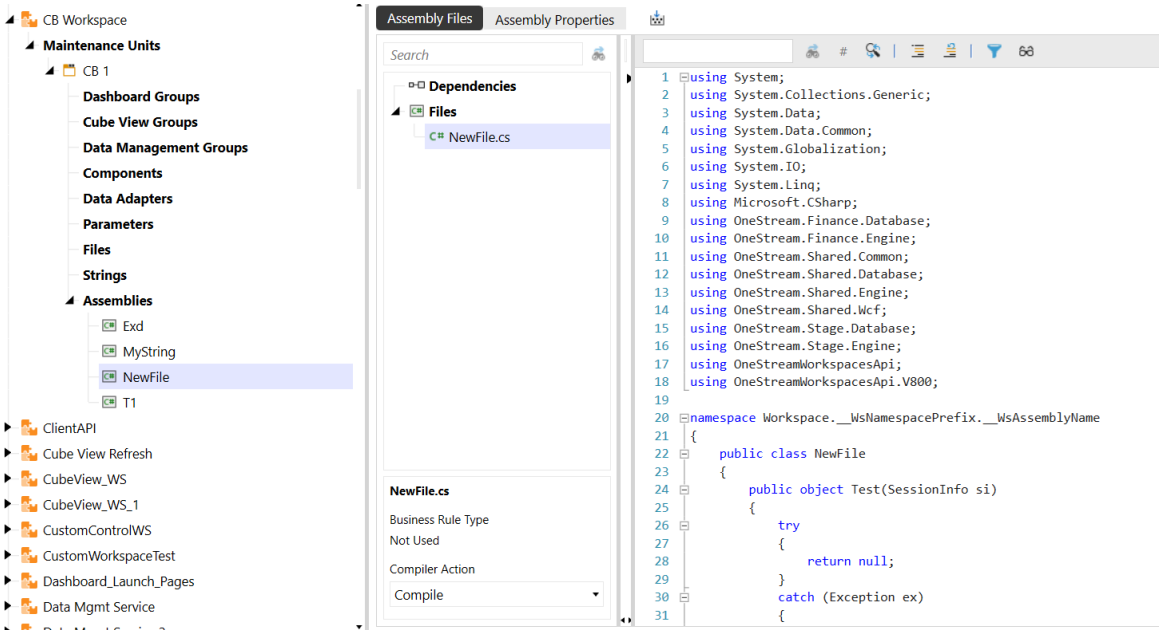
Create an Assembly File

The assembly file is where you write the logic. You can choose to write a Business Rule type or a rule of your own (Not Used). The Not Used option provides great flexibility and is a good choice for most developers. Additionally, some performance gains may be realized with this choice.

1. Choose an assembly and click the **Assembly Files** tab.
2. Right-click on **Files** and select **Add File**.



3. In the **Add File** dialog box:
    a. Type a file name.
    b. Select a **Source Code** type. You can choose either **Business Rule** or **Service Type**.
    c. Select a **Compiler Action**. You can choose to compile the file when the assembly is compiled. Use Default is the same as Compile.
4. Click **OK** then **Save**. Your workspace will look similar to the following.



> **NOTE:** You can change Compile Action inside this window but you cannot change the rule type. To do so, you need to delete the file.

Delete an Assembly File

1. Click on the file you would like to delete.
2. On the Assembly Properties tab, click **Delete**.

Encrypt an Assembly File

You may want others to have access to your Assembly Files. Encryption protects specific assembly files from users who may or may not have access to this workspace. Security can be controlled in the following ways:

- Workspace level (access and maintenance security)
- Maintenance unit level (access and maintenance security)
- Application security roles that prevent others from writing assemblies in the application workspaces page (AdministerApplicationWorkspaceAssemblies)
- System security roles that prevent others from writing assemblies in the system workspaces page (SystemApplicationWorkspaceAssemblies)

You must be inside the EncryptBusinessRules application security property to perform the following steps.

1. Right-click on an assembly file.
2. Select **Encrypt File**.
3. Provide a password and click **OK**. After you have provided a password, the file will be encrypted.
4. Optional. If you would like to decrypt the file, right-click and select **Decrypt File**.

Create an Assembly Folder

Assembly folders organize your assembly files. To create an assembly folder:

1. Select the assembly then click the **Assembly Files** tab.
2. Right-click the **Files** label and select **Add Folder**.
3. Give the folder a name and click **OK**.
4. Optional. To add files to the folder, right-click on the folder and select **Add File**.

> **NOTE:** You cannot drag and drop files into a folder.

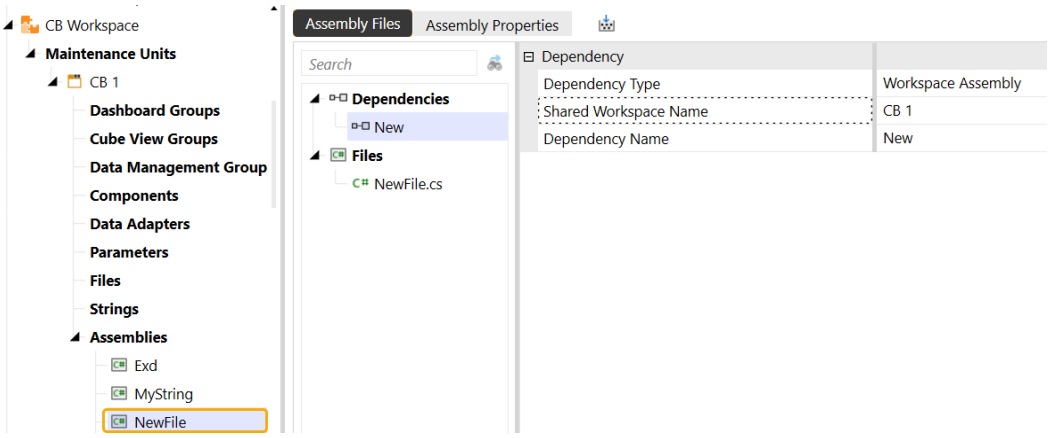
Create a Dependency

Dependencies are useful when you have methods within other assemblies or business rules.

1. Right-click the **Dependencies** label and select **Add Dependency**.
2. Choose a dependency type:

    - **Workspace Assembly**: Used when other assemblies might be included in this workspace. If you are creating a dependency with another workspace, you must provide the Shared Workspace Name and this workspace must have the Is Shareable Workspace property set to True.

      Here is an example of how you would complete this dependency setup (multiple workspaces can be entered here in a comma delimited list).



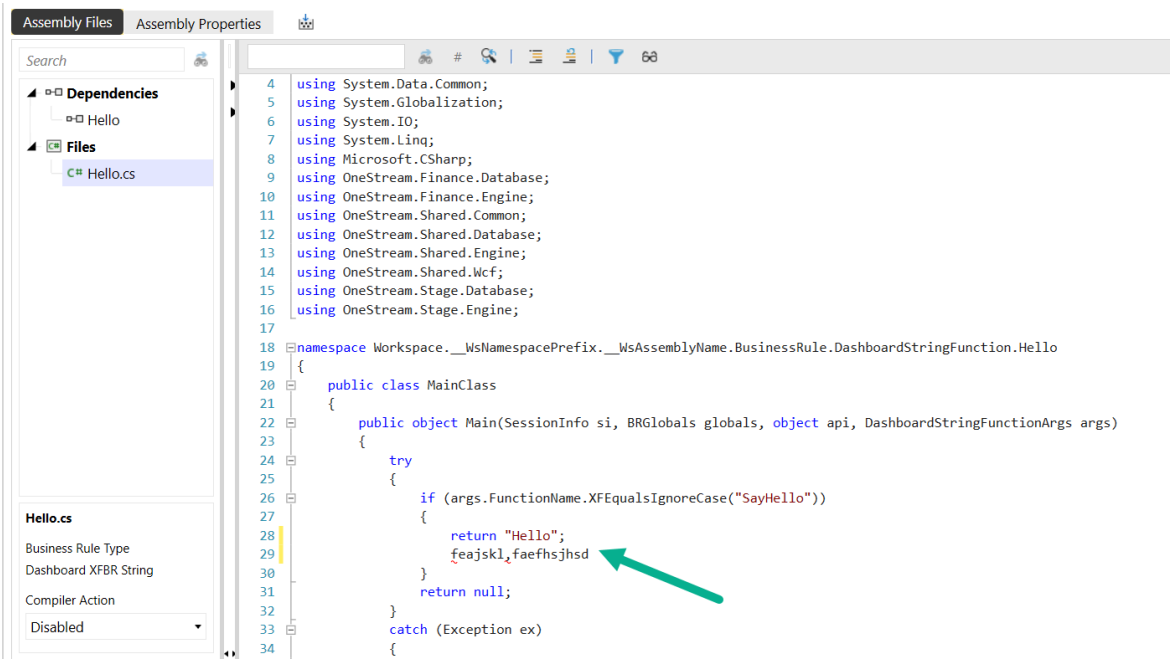
> **NOTE:** If you are creating a dependency within the same workspace, do not fill in the Shared Workspace Name property.

    - **Prepackaged Assembly**: Assembly automatically included with the OneStream installation.
    - **Business Rule**: Create dependencies with any business rules located on the Business Rules page. Type the name of the business rule in the Dependency Name property and click **Save**.

After creating dependencies, you might receive an error when compiling the assembly if there is anything incorrect within the assembly or dependency itself.

Important notes when compiling with assemblies:

- If the Compiler Action is set to **Disabled** for any dependent assembly file, you will not see an error even if you have issues with your syntax. For example, the following dependent assembly would not produce a syntax error even though one is present.



- Assembly Files properties require you to type in the exact name of the workspace, assembly, and business rule. If incorrect, you will receive a compile error.

Source Code Types

When you create an assembly file, you must choose a source code type. This populates the Business Rule Type property inside the rule. You can choose from two source code types: Business Rule types and Service types.

There are several duplications between Service types and Business Rule types, which provides a choice on how you write rules in assemblies. For example, the XFBR String Service and the Dashboard String Function business rule both function the same. In some cases, you might realize slight performance gains when using Service types as they offer more flexibility.

Business Rule Types

There are two ways to create the various business rule types (cube view extenders, XFBR strings, dashboard extenders, spreadsheets, or data sets):

1. Choosing a business rule type when the assembly is created.
2. Configuring a Service Factory.

In this section, you will learn more about the first option as this is more familiar to developers who have written business rules in OneStream. However, the recommended action for solution developers is to use the second option due to performance impacts.

Regardless of the option you choose, it is important to be aware of the slight differences when calling rules created as assemblies versus calling business rules.

Dashboard String Function Business Rules in Assembly Files

The following examples of an XFBRString were created as files within assemblies. These types of rules are often referenced throughout the application to more finely tune formatting and display options. These business rules are usually called using the following syntax:

XFBR(RuleName, FunctionName)

However, if this XFBR was created as an assembly, it would look like this:

XFBR(Workspace.WorkspaceName.AssemblyName.FileName, FunctionName)

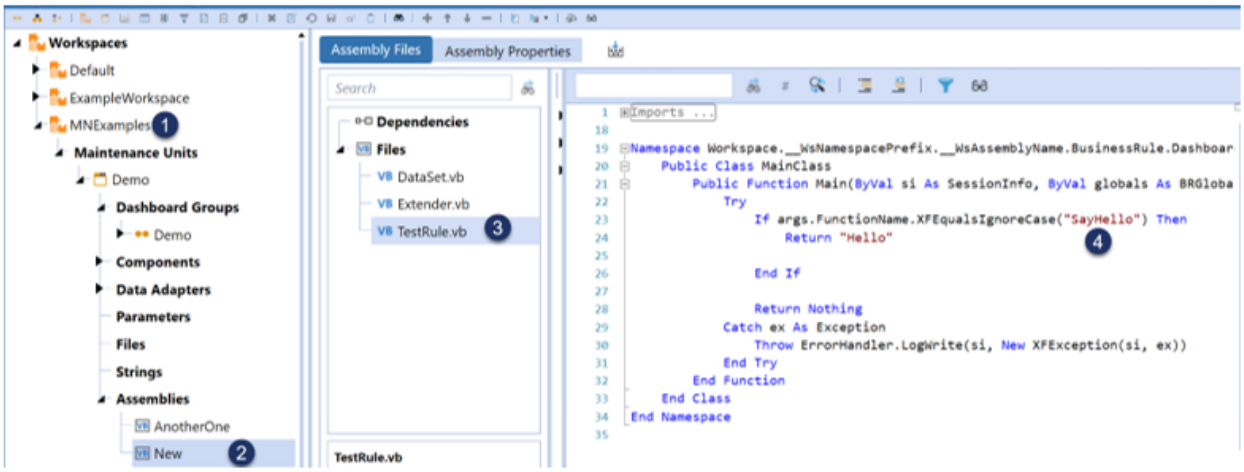
More information is required to call this rule to ensure correctness, such as:

XFBR(Workspace.MNExamples.New.TestRule, SayHello) or **XFBR (Workspace.Current.New.TestRule, SayHello)**

For workspace name, you can use the word "Current" to refer to the workspace you are currently in. This is useful when you are referencing rules within objects created in each workspace.

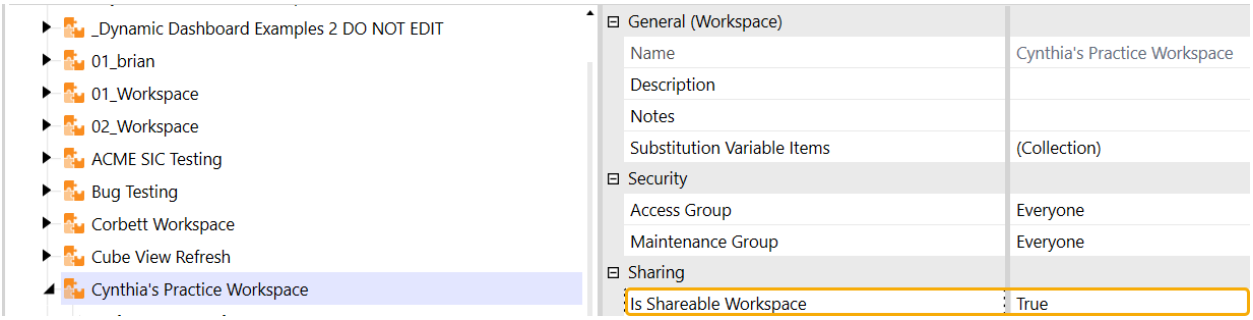
XFBR(Workspace.Current.AssemblyName.FileName, FunctionName)

To call an XFBR workspace assembly rule, use this syntax and replace "Current" with the actual workspace name.



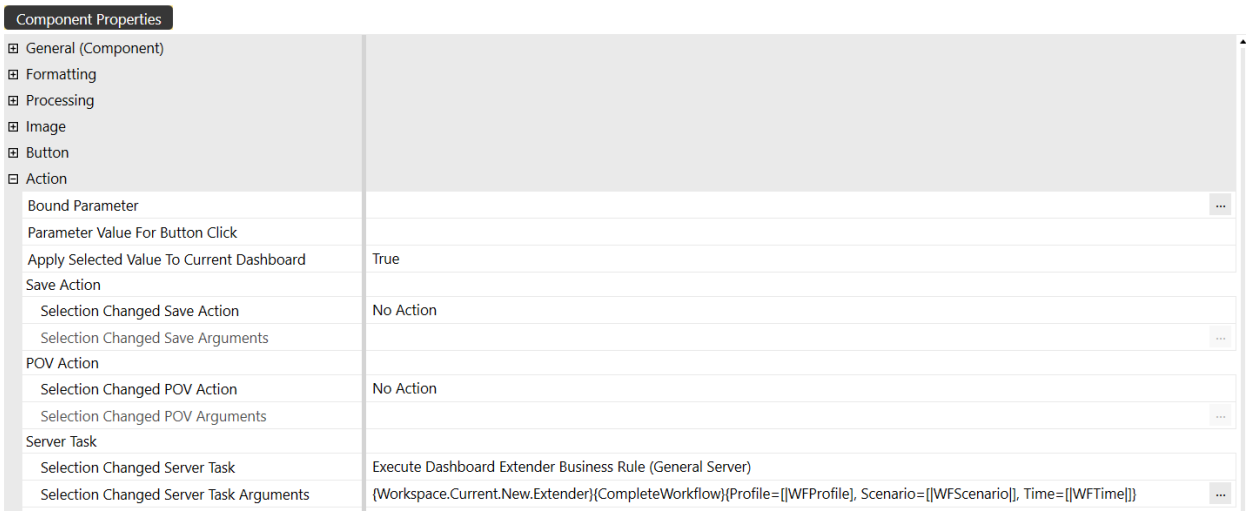
1. Name of the workspace
2. Name of the assembly
3. Name of the assembly file
4. Source code

> **NOTE:** If you are not using “Current” as the workspace name you must have the Is Shareable Workspace property set to True, even if you are referencing the rule within the same workspace. If you do not do this, your rule will not run.



Dashboard Extender Business Rules in Assembly Files

In the following example, when a button is clicked, a dashboard extender rule completes the selected workflow profile.



Typical syntax for this rule written in the business rules page would be:

{MyDashboardExtenderBRName}{MyFunction}{Param1=[MyValue1], Param2=[MyValue2]}

Because this is an assembly, however, the rule is modified to provide more specific syntax:

{Workspace.WorkspaceName.AssemblyName.FileName}{MyFunction}{Param1=[MyValue1], Param2=[MyValue2]}

Dashboard Data Set Business Rules in Assembly Files

In this example, a Method Query Data Adapter is referencing a business rule. However, because this was created as an assembly file, appropriate syntax must be used. This type of business rule is typically called with the following syntax:

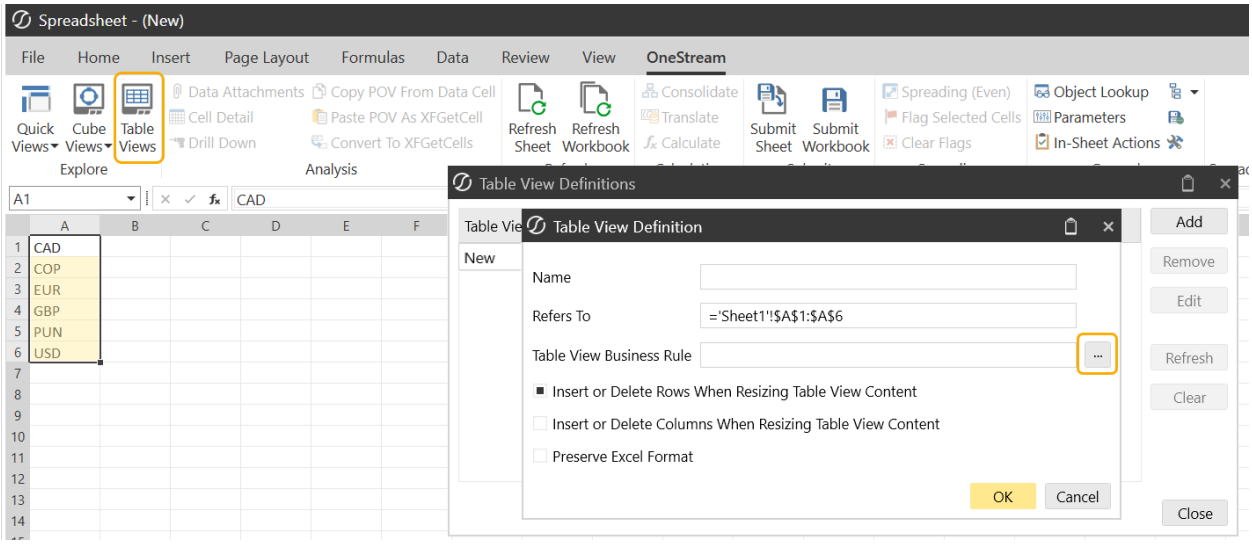
{MyDataSetBRName}{DataSetName}{Param1=[MyValue1], Param2=[MyValue2]}

Because this is an assembly, it has to be modified:

{Workspace.WorkspaceName.AssemblyName.FileName}{DataSetName}{Param1=[MyValue1], Param2=[MyValue2]}

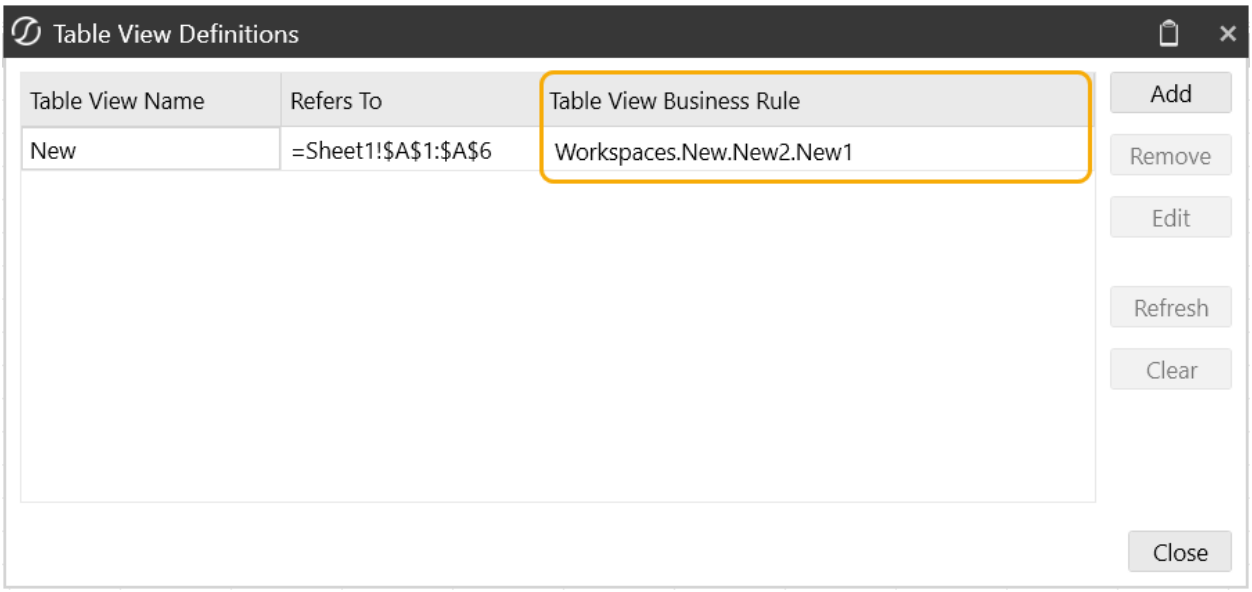
Spreadsheet Business Rules in Assembly Files

Business rule names are typically chosen by clicking the ellipsis icon in the spreadsheet:



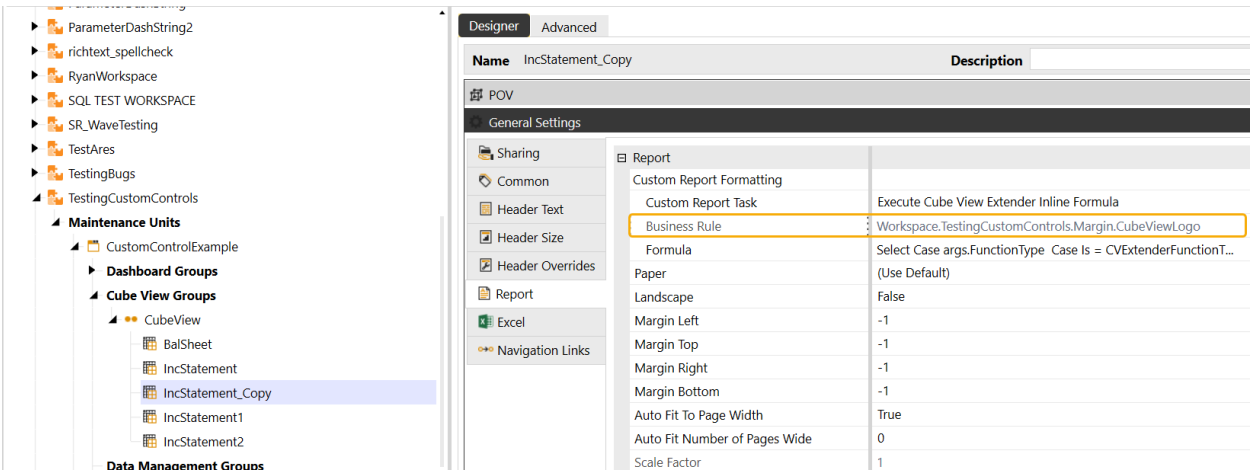
However, if you have created the spreadsheet rule as an assembly file, you will notice that the rule cannot be found by clicking the ellipsis icon. The spreadsheet assembly can still be referenced using the business rules syntax:

Workspace.WorkspaceName.AssemblyName.FileName



Cube View Extender Business Rules in Assembly Files

Cube view extender rules are commonly used to fine-tune formatting of cube views. You can write these rules directly on a cube view, from the Business Rules page, or through assemblies. These rules run only on the PDF version of the cube view.



To reference a cube view extender rule that was created as an assembly, you should set the cube view custom report task to "Execute Cube View Extender Business Rule." Refer to the rule using the following syntax:

Workspace.WorkspaceName.AssemblyName.FileName

Do not use the ellipsis icon to select a business rule in this case, as this only provides rules present in the Business Rules page.

Finance Business Rules

Finance Business Rules are integrated into Workspace Assemblies to make it easier to write and maintain business rules. You can write Finance Business Rules using one of the four finance source code types. Developers and Implementers can run the new Finance Business Rules in Workspace Assemblies using data management jobs, BRApi, or by using a dashboard button server task.

The Business Rules name assigned can be modified under the Cube Properties tab in the Business Rules section on the Cubes page. You can also manually type in the Business Rule instead of selecting from the Object Lookup dialog. To use Finance Business Rules in Workspace Assemblies with existing business rules, you can select and copy code from one of the service types and paste it in the same arguments.

> **NOTE:** The following code examples in this section are written in C#.

Finance Service Types

There are four source code service types you can select when creating Finance Business Rules:

- Finance Core Service
- Finance Custom Calculate Service
- Finance Get Data Cell Service
- Finance Member Lists Service

> **NOTE:** The cases for these Finance Service types have been included in the template code for new Service Factory files as shown below.

Finance Business Rules

```
//case WsAssemblyServiceType.FinanceCore:
// return new WsasFinanceCore();
//case WsAssemblyServiceType.FinanceCustomCalculate:
// return new WsasFinanceCustomCalculate();
//case WsAssemblyServiceType.FinanceGetDataCell:
// return new WsasFinanceGetDataCell();
//case WsAssemblyServiceType.FinanceMemberLists:
// return new WsasFinanceMemberLists();
```

These business rules are called using the following syntax:

Workspace.MyWorkspaceNameOrNSPrefix.WS

> **IMPORTANT:** These source code types can be written in C# and VB.Net.

> **NOTE:** The same objects are available for use with each finance service type, such as api and args.

Finance Core Service

Finance Core Service allows you to calculate functions. It contains six method types. These six methods are called automatically by the platform from business rules assigned to the Cube. This service is added to Cube Properties. You can separate functions replacing these **Select Case** blocks on:

- Calculate
- Translate
- ConsolidateShare
- ConsolidateEliminate
- GetCustomFxRate
- GetConditionalInputResult ("NoInput")

Most Implementers will use this service type and implement the following calculate function:

Finance Core Service

```
        public void Calculate(SessionInfo si, BRGlobals brGlobals, FinanceRulesApi api,
FinanceRulesArgs args)
 {
 try
 {
 }
 catch (Exception ex)
 {
 throw new XFException(si, ex);
```

Finance Custom Calculate Service

Finance Custom Calculate Service allows you to create custom calculations. It contains one method type. This service runs from a data management step. Its dedicated function is responsible for replacing **SelectCase** blocks.

Finance Custom Calculate

```
namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
{
    public class FinanceCustomCalculateServiceName : IWsasFinanceCustomCalculateV800
    {
 public void CustomCalculate(SessionInfo si, BRGlobals brGlobals, FinanceRulesApi api,
FinanceRulesArgs args)
 {
 try
```

```
 {
 }
 catch (Exception ex)
 {
 throw new XFException(si, ex);
 }
 }
    }
}
```

Finance Get Data Cell Service

Finance Get Data Cell Service allows you to use the GetDataCell expression in the performance section of a dynamic calculation. It contains one method type. This service is called from Cube Views. Its dedicated function is responsible for replacing **Select Case** blocks.

Finance Get Data Cell

```
namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
{
    public class FBR3 : IWsasFinanceGetDataCellV800
    {
 public object GetDataCell(SessionInfo si, BRGlobals brGlobals, FinanceRulesApi api,
FinanceRulesArgs args)
 {
 try
 {
 return null;
 }
 catch (Exception ex)
 {
 throw new XFException(si, ex);
 }
 }
    }
}
```

Finance Member Lists Service

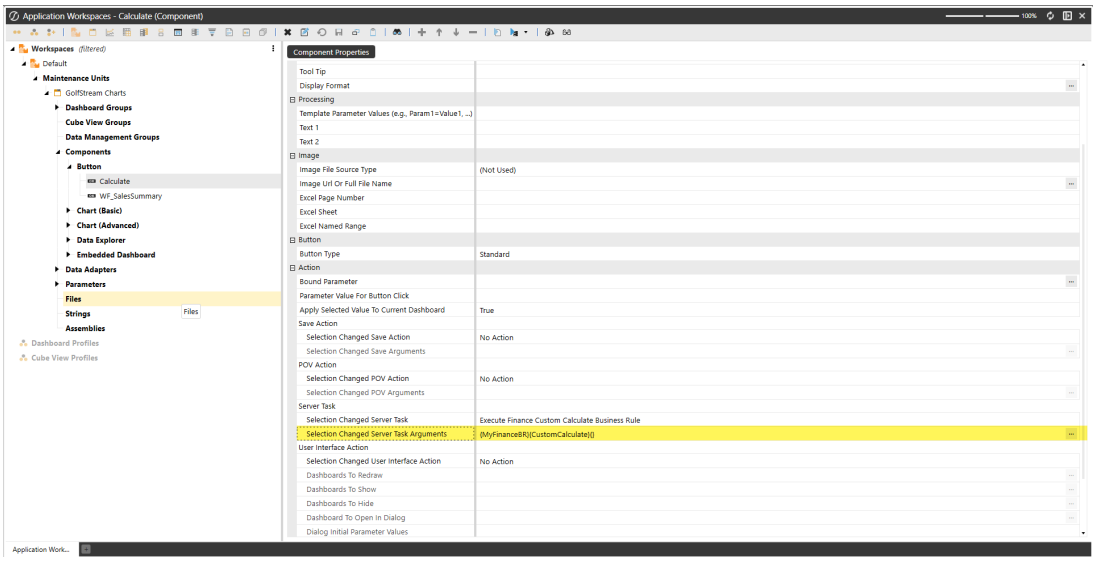
Finance Member Lists Service allows you to create a member list. It contains one method type. This service is called from Cube Views. Its dedicated function replaces **Select Case** blocks and removes the "Headers" blocks.

Finance Member List

```
namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
{
    public class FBR4 : IWsasFinanceMemberListsV800
    {
 public MemberList GetMemberList(SessionInfo si, BRGlobals brGlobals, FinanceRulesApi api,
FinanceRulesArgs args)
 {
 try
 {
 return null;
 }
 catch (Exception ex)
 {
 throw new XFException(si, ex);
 }
 }
    }
}
```

Dashboard Button Server Task

You can also run a Finance Business Rule with a Custom Calculate using a Dashboard Button Server Task.



Recompile

Workspace Assemblies will recompile when making updates and changes to them. The following examples indicate when a Workspace Assembly will recompile with Finance Business Rules changes.

> **Example:** If Workspace Assembly A has a dependency on Workspace Assembly B and changes are made to Workspace Assembly B, Workspace Assembly A will recompile.

> **Example:** If a Workspace assembly has a reference to a Business Rule and the Business Rule changes, then the Workspace assembly will recompile. This will then extend to the cube view grids, where if you change a Finance Business Rule, the cube view data will be marked to indicate that the underlying data and Finance Workspace assembly has changed, and therefore must be recalculated.

Object Lookup for Finance Business Rules

The Workspace Assembly Service syntax will display in the following areas:

- Go to **Application** > **Cubes** > **Cube Properties** > **Business Rules**. Click on ellipsis to open the Object lookup dialog box that contains the Workspace Assembly Services and Business Rules.

Application Business Rules will display in Object Lookup in the following areas:

- Go to **Application** > **Business Rules** and select the **Search** toolbar button. This will list the Business Rules.
- Go to **Application**> **Business Rule** > **Formula** > **code editor**, click on Object Lookup. This will display all objects including Workspace Assembly Services and Application Business Rules under the Business Rules Object Type.

System Business Rules will display in Object Lookup in the following areas:

- When logging in to **System** > **Business Rules** > **Search** toolbar button.
- Go to **System** >**Business Rule** > **Formula** > **code editor**, click on Object Lookup. This will display all object including Workspace Assembly Services and Application Business Rules under the Business Rules Object Type.

> **NOTE:** In most locations, the OneStream Code Editor will include the Object Lookup, which will display Workspace Assembly Services and Business Rules within the Business Rules Object Type.

Create Finance Business Rules

1. Go to **Application** > **Presentation** > **Workspaces**.
2. Create an Assembly or select an existing assembly.
3. Create an Assembly file using the Service Factory source code type.

   > **NOTE:** The Service Factory file will list the four Finance Business Rules service types.

4. Create a new Assembly File for your Finance Business Rule. When creating an Assembly File, you must select one of the four finance source code types.

- Finance Core Service
- Finance Custom Calculate Service
- Finance Get Data Cell Service
- Finance Member Lists Service

5. Configure the Workspace Service Factory code to enable your service and make updates.

   > **IMPORTANT:** The object name must match the file name.

6. Click **Save** to save all changes.
7. Go to **Application** > **Cubes** > **Cube Properties** > **Business Rules**. Factories can be referenced from Cube Properties.

   > **TIP:** The Business Rules fields are editable.

8. Add a Workspace Assembly Service to your Business Rule. Click the **Edit** button in the Business Rule field and then use the Object Lookup to copy the Workspace Assembly Services Syntax: **Workspace.MyWorkspaceNameOrNSPrefix.WS**
9. Replace the "MyWorkspaceNameOrNSPrefix" placeholder text with your Workspace Name or Namespace Prefix. Click **Save** to save the Business Rule to your Cube Properties.

XF Project

XF Project extract is for application project designers who are building solutions that span many artifacts, such as workspaces, data management groups, dashboard maintenance units, business rules, cubes, dimensions, cube views, and other artifacts. The application Extract and Load option collects defined objects, such as dashboards and business rules, as a single file export package that can be reloaded as a package.

XF Project is a convenient way to organize workspaces, data maintenance units, or similar solutions into a folder structure that can be integrated with a version control system, such as Git. Developers must create an XML file that is the definition for the contents of the project export.

Project File

Application designers must first manually define an XML file to support the export of objects as a project file. The file is then saved with the .xfProj file extension to a local project folder that also supports a version control system.

```
<xfProject topFolderPath="" defaultZipFileName="">
  <projectItems>
    <projectItem projectItemType="DashboardWorkspace" folderPath="" workspace="Corporate" name="Corporate" includeDescendants="false" />
    <projectItem projectItemType="DashboardMaintenanceUnit" folderPath="" workspace="Corporate" name="Corporate Reports" includeDescendants="true" />
    <projectItem projectItemType="DashboardMaintenanceUnit" folderPath="" workspace="Corporate" name="Corporate Templates" includeDescendants="true" />
    <projectItem projectItemType="DashboardProfile" folderPath="" workspace="Corporate" name="Corporate Solution" includeDescendants="true" />
  </projectItems>
</xfProject>
```

The root node of xfProject contains two attributes:

- **TopFolderPath**: Creates and defines the starting folder location where the specified files are extracted.
- **DefaultZipFileName**: Creates a standard default file name for .zip file extracts.

ProjectItems is a list structure that contains the project items to extract. No attributes needed.

ProjectItem reflects what is needed to extract from OneStream or to load from the file system. It has five attributes:

- **ProjectItemType**: The type of the project item. Includes the following types:
  - BusinessRule
  - Cube
  - CubeViewGroup
  - CubeView

- CubeViewProfile
- DashboardWorkspace
- DashboardMaintenanceUnit
- DashboardFile
- DashboardString
- DashboardParameter
- DashboardGroup
- DashboardAdapter
- DashboardComponent
- Dashboard
- WorkspaceAssembly
- DashboardProfile
- DataManagementGroup
- DataManagementSequence
- DataSource
- Dimension
- TransformationRuleGroup
- TransformationRuleProfile

- **FolderPath**: The name of the subfolder where the project item type is extracted.
- **Workspace**: The name of the workspace for the project item.
- **Name**: The name of the project item.

- **IncludeDescendants**: The default is True and only affects the following project item types:
    - CubeViewGroup
    - DashboardWorkspace
    - DashboardGroup
    - DashboardMaintenanceUnit
    - DataManagementGroup

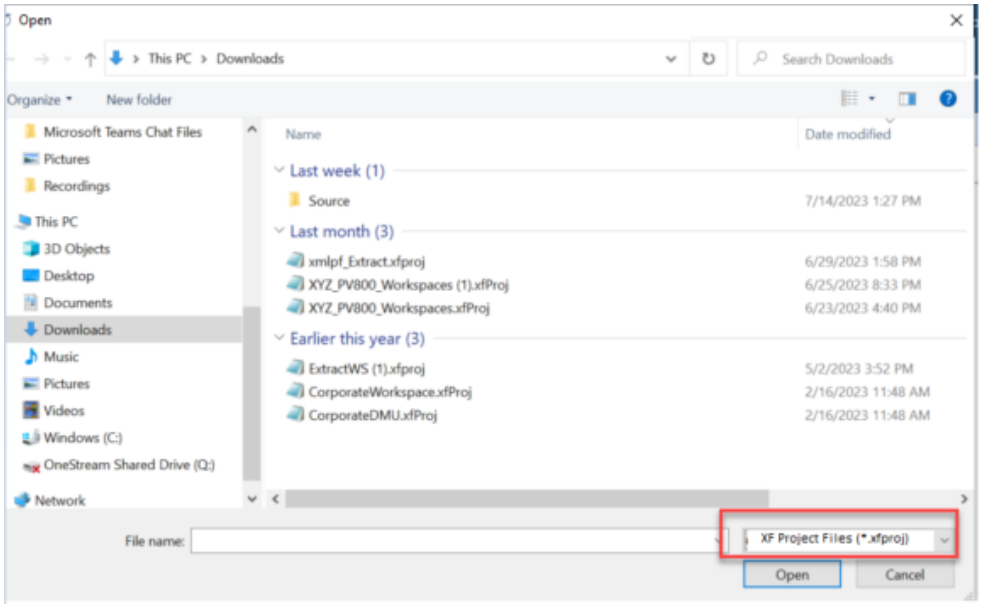
File Extract

You can place the .xfProj file into a local folder, such as your desktop. The defined folder path folders will be generated there as the target location for application exports and loads. There are two file extract options available:

- **.zip**: The export option will collect all objects defined in the .xfProj file as a zip file to the location of the .xfProj file.
- **File**: Exports all objects defined in the .xfProj file to the folder path locations defined in the .xfproj file.

To use file extract:

1. Navigate to **Application** > **Tools** > **Load/Extract**.
2. Click the **Extract** tab.
3. Under File Type, select XF Project.
4. Click the ellipses (...) and navigate to the .xfProj file.



5. (Optional) Select the **Extract to Zip** checkbox to create an application zip file that contains all defined objects.
6. Click **Extract** on the toolbar.

> **NOTE:** For XML extract, if you select the file type of “Application Workspaces,” the system will update the third line of code from <applicationDashboardsRoot> to <applicationWorkspacesRoot> on the extract.
>
> For XML load, the system properly loads <applicationWorkspacesRoot>. To support backward compatibility, if you have an XML extract that contains <applicationDashboardsRoot>, the system still loads the older version.

File Load .xfProj

File loading using the defined .xfProj file provides a seamless link to the project files. When you load an .xfProj file you will see the option to merge or replace the target files. The only files affected are those defined by the .xfProj file.

If you select replace, only files that differ for CubeViewGroups, DashboardWorkspaces, DataManagementGroups, DashboardMaintenanceUnits, and DashboardGroups are removed. For all other items, such as business rules or extensibility rules, replace acts as a merge. You can remove files that differ with Assembly file folders as well. If you select Merge, the content of the files are merged and added into the application.

To load an .xfProj file:

1. Navigate to **Application** > **Tools** > **Load/Extract**.
2. Click the **Load** tab.
3. Under File Name, browse to the .xfProj file and select it.
4. Click **Load** on the toolbar.
5. Select the load method, either Merge or Replace.

> **NOTE:** For XF Project, when extracting a file, the folder structure is updated from "Application Dashboards" to "Application Workspaces."

Zip Load

Zip file load functions like any other application file load. The contents of the file are merged into the application. Zip file load is not supported by alternative merge or replace file load options.

Build Cube Views in Workspaces

Cube views are used for reporting, analysis, and data entry. You can create cube views on the Cube Views page under the Application tab and also build and use them within the Workspaces page.

Depending on your security roles, cube views created on the Cube View Page can be used through the Default Maintenance Unit in the Default Workspace. You can create additional cube views in other workspaces but they cannot be accessed by the Cube View page.

You can load and extract cube views, cube view groups, and cube view profiles using the traditional XML process. The file type of Application Dashboards is renamed to Application Workspaces.

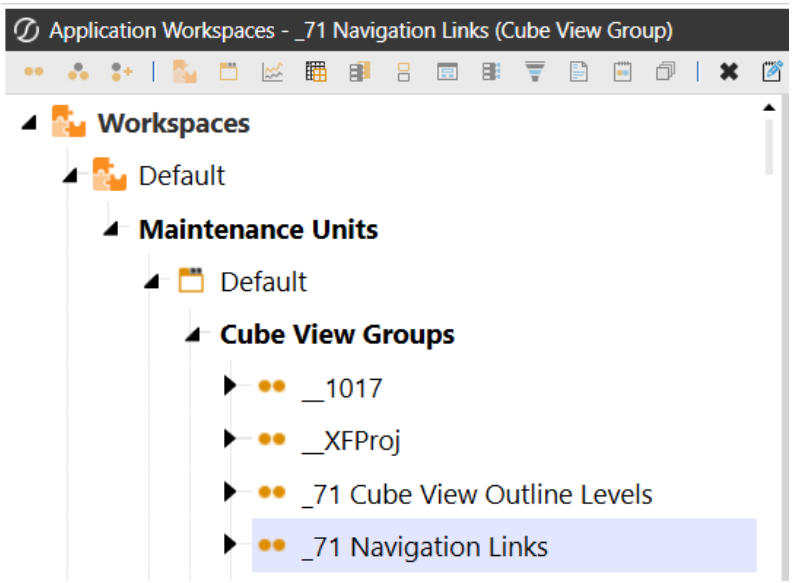
> **NOTE:** When working with cube views that are in the non-default workspace, the workspace must be shareable.

Display Cube Views Page and Groups

There is a new Default folder under Maintenance Units within the Workspaces Default Maintenance Units section. This is where the cube views page and cube view groups reside.

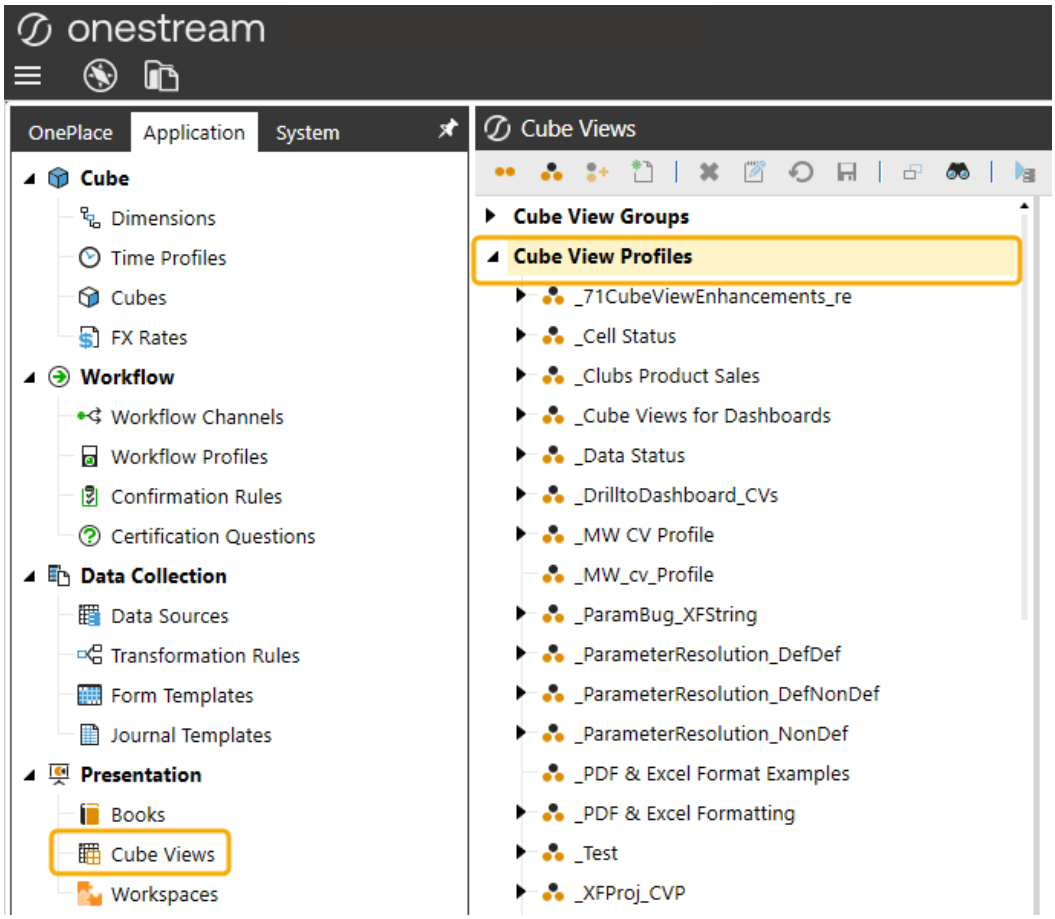
1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. In the **Application Workspaces**pane, under **Workspaces**, expand **Default** >

Maintenance Unit > **Default** > **Cube View Groups**.



Display Cube View Profiles and Groups

1. On the **Application** tab, under **Presentation**, click **Cube Views**.
2. In the **Cube Views** pane, expand **Cube View Groups** or **Cube View Profiles**.



Configure the Service Factory

Assemblies can only be written inside Workspaces. This allows for greater flexibility, extraction, and packaging, and it provides the ability to create Dynamic Dashboards, Dynamic Cubes, Dynamic Cube Views, and Dynamic Grids.

You can implement a Service Factory, as described in the following sections, to use Workspaces capabilities.

Runtime Logic

The Service Factory is a dispatcher object that routes requests to run rules to specific objects called Services that can satisfy these requests.

In a basic setup, a Service Factory can have a single service for each type of rule. For example, one is for Data Management calls, and one is for XFBR calls. However, a Service Factory can also invoke multiple services of a specific type, depending on context variables. For example, it could route calls for Data Management jobs to one service from Monday to Friday and then to a different service from Saturday to Sunday.

It is not required to provide a service for every supported type.

> **Example:** If the Service Factory is not used for requests of that type, it may use the traditional approach, and everything will function correctly.

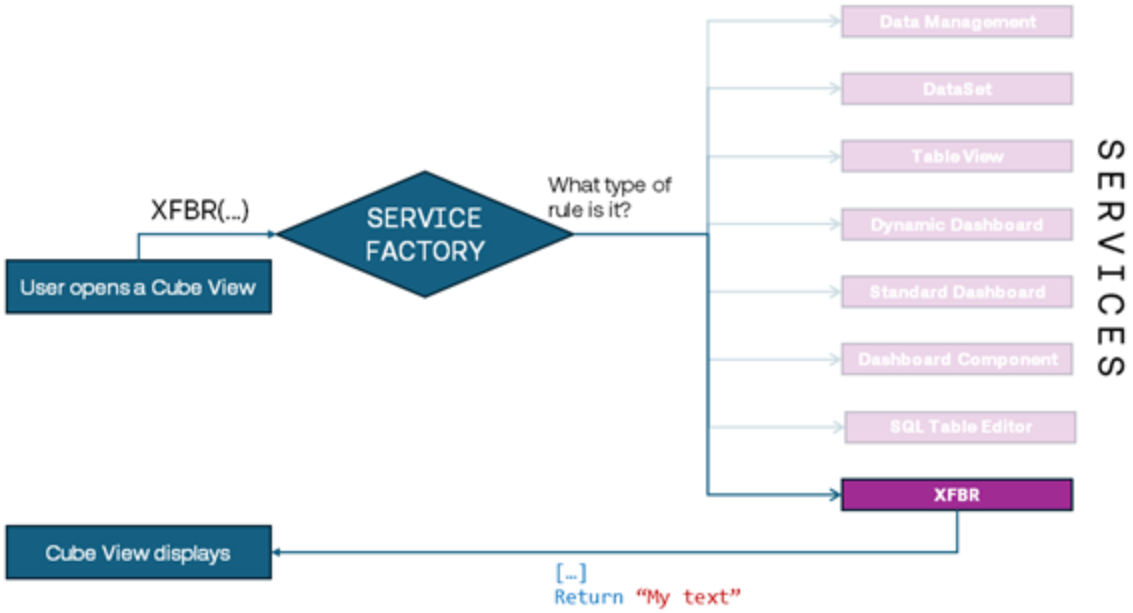
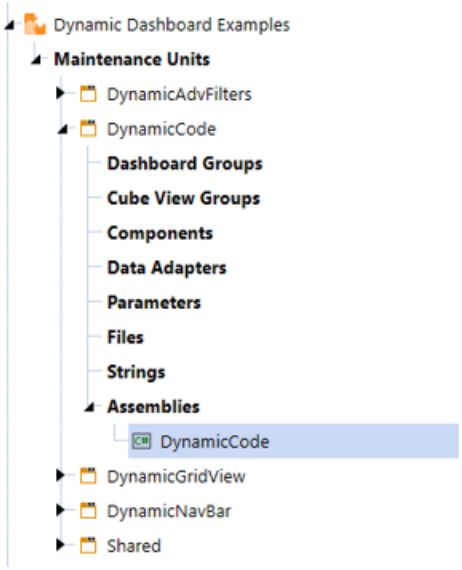


Figure 1 - Example Service Factory execution

Configuration

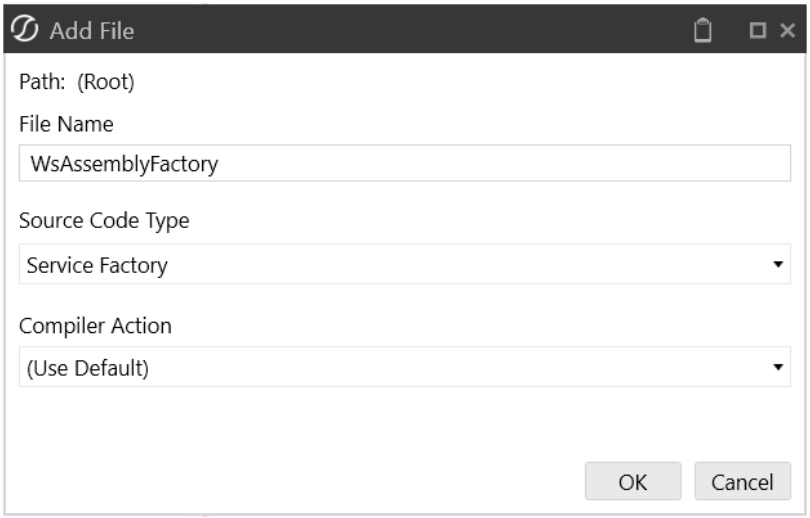
Step 1: Create a Maintenance Unit for all assemblies within a Workspace.

The first step is organizational, designed to keep all rules within a single location. Create a Maintenance Unit within the Workspace to hold all necessary assemblies. The following image displays multiple Maintenance Units in the Workspace, but only *DynamicCode* Maintenance Unit contains assemblies. This will be the central location for all rules.

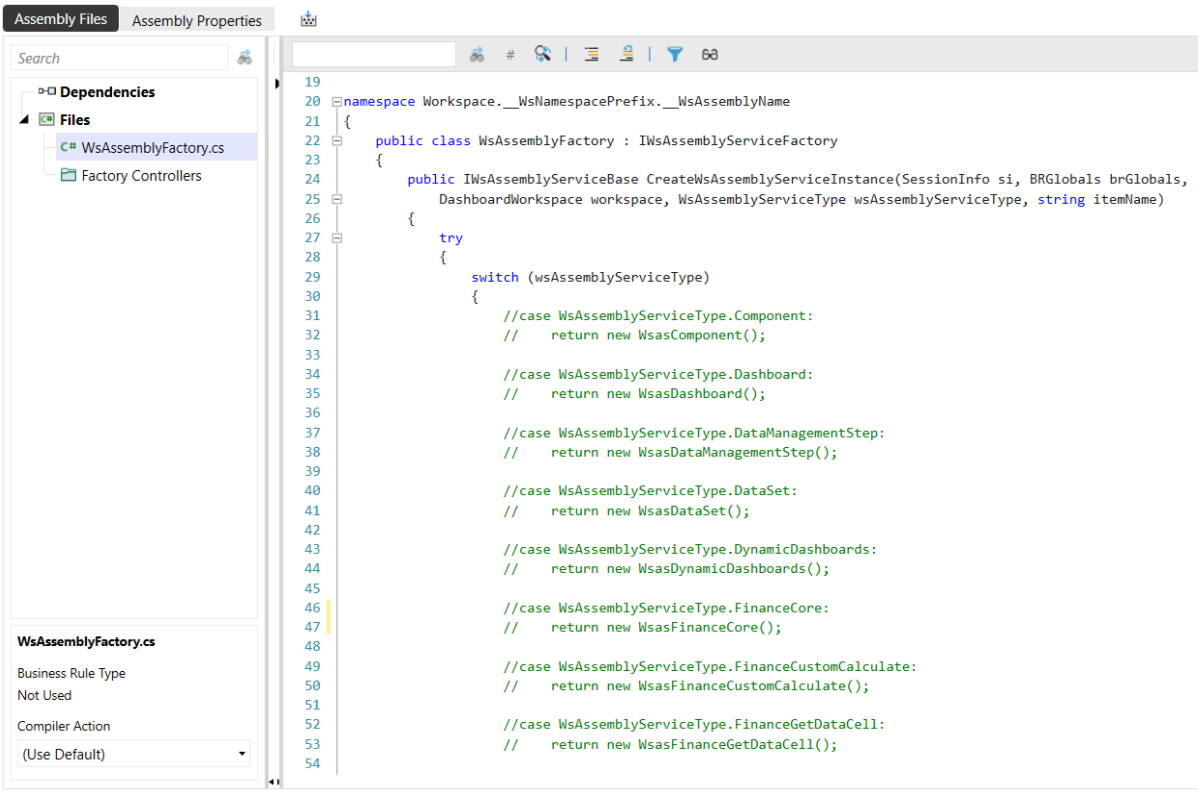


Step 2: Create an assembly file for the Service Factory.

The second step is to create an assembly file with the source code type of Service Factory.



The Service Factory assembly file is needed to take advantage of all items used within assemblies. It should look similar to the following image (the example uses C#):

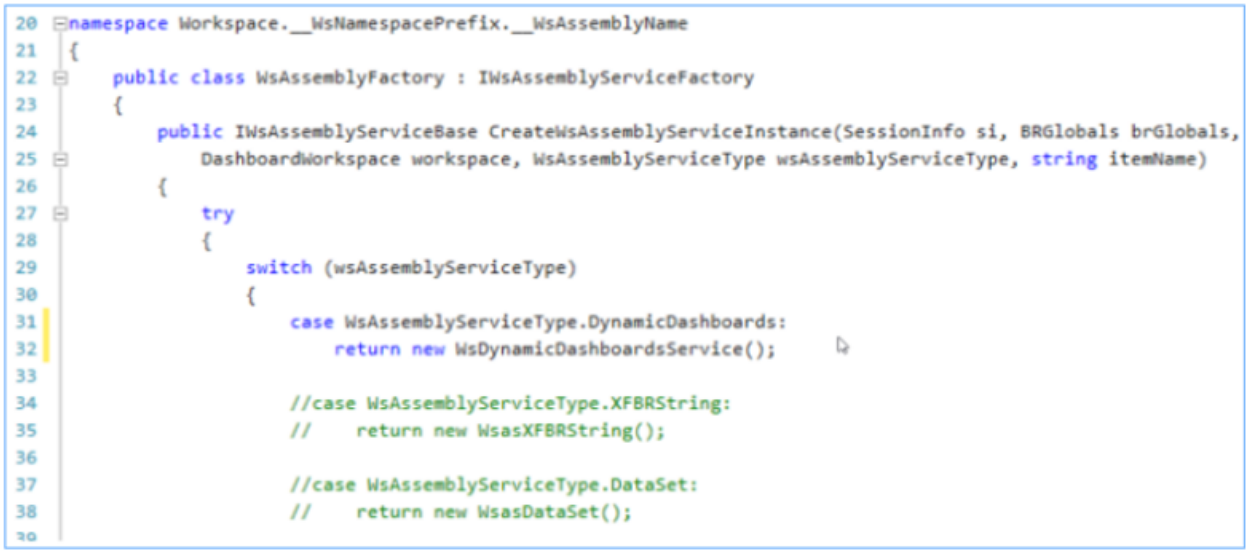


The code allows you to create the following objects:

- Components
- Dashboards
- DataManagementStep
- DataSet
- Dynamic Dashboard
- Dynamic Cube View
- Finance Business Rules
- SqlTableEditor

- TableViews
- XFBRString

If you are not using any of the objects listed, keep them commented out in the code. Otherwise, you will receive a syntax error when compiling. For example, if you only need the *DynamicDashboard* call, the Service Factory would look similar to the following image:

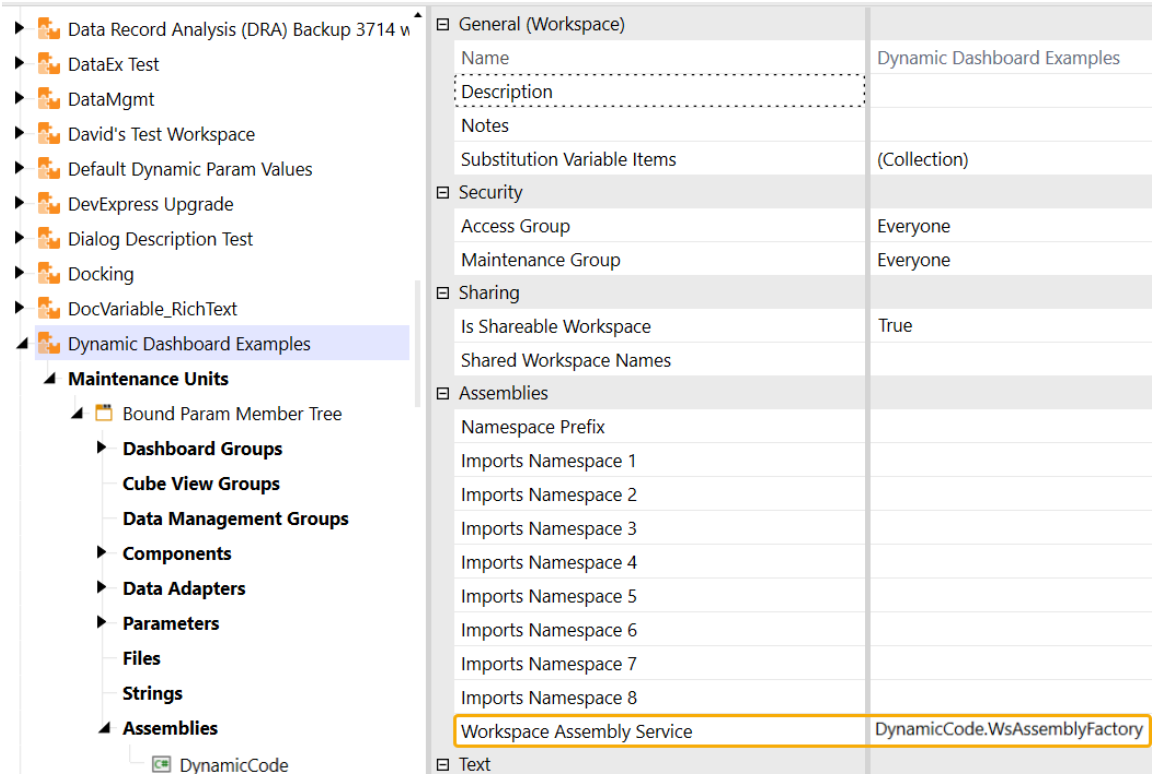


Step 3: Configure the Workspace Assembly Service property.

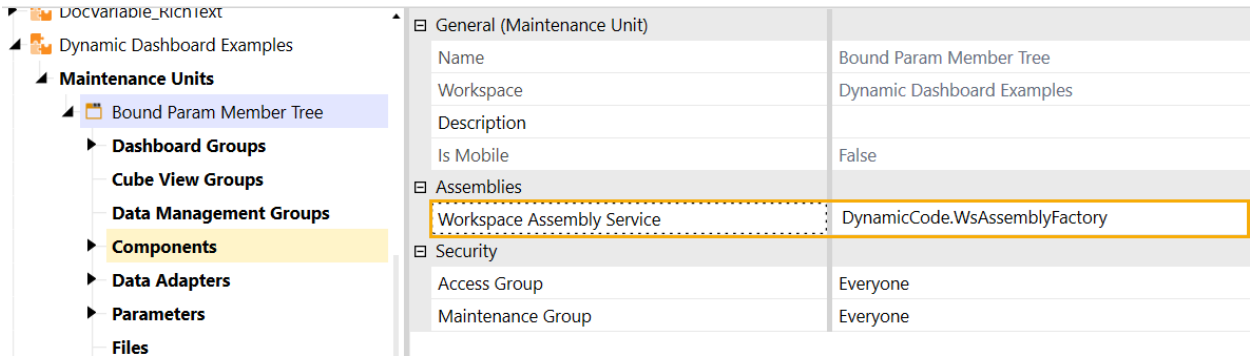
This last step allows the entire Workspace to use the new Service Factory file so that your rules run correctly.

> **IMPORTANT:** Assembly files will not run correctly if you skip this step.

Go to the Workspace, and update the Workspace Assembly Service to {WsAssemblyName}.{ServiceFactoryName}. For this example, that would be *DynamicCode.WsAssemblyFactory*.



You can also do this on any given Maintenance Unit. As a best practice, you can keep all assembly files in a single Maintenance Unit. You can designate the Workspace and Workspace Maintenance Unit level in all actions.



Reference the Service Factory in Properties

Service Factories can be referenced from configuration properties in different ways, depending on their location.

assemblyName.factoryName

This will look for a Service Factory File called factoryName in the Assembly called assemblyName.

This can be used on properties of:

- Maintenance Unit
- Workspace

WSMU

This will look for the Service Factory specified on the Maintenance Unit containing the object. If it cannot find it, it will search for the Service Factory specified on the Workspace containing the object. It will fail if it is empty.

This can be used on properties of:

- Dashboard Components

WS

This will look for the Service Factory specified on the Workspace containing the object. It will fail if it is empty.

This can be used on properties of:

- Dashboard Components

Workspace.workspaceName.maintenanceUnitName.WSMU

This will look for the Service Factory specified on the Maintenance Unit called maintenanceUnitName in Workspace called workspaceName. It will fail if it is empty.

> **NOTE:** You can specify a Namespace Prefix as workspaceName.

If the specified Maintenance Unit is not in the Workspace containing the object that originates the call, or this syntax is used in Cube Views, the specified Workspace must be set to Is Shareable Workspaces.

This can be used on properties of:

- Cube Views
- Dashboard Components
- Data Management Steps

Workspace.workspaceName.WS

This will look for the Service Factory specified on Workspace workspaceName. It will fail if it is empty.

> **NOTE:** You can specify a Namespace Prefix as workspaceName.

If the specified Workspace is not the one containing the object that originates the call, or this syntax is used in Cube Views or Cube Properties, the specified Workspace must be set to Is Shareable.

This can be used on properties of:

- Cube Properties
- Cube Views
- Dashboard Components
- Data Management Steps

Workspace.Current.maintenanceUnitName.WSMU

Workspace.Current.Current.WSMU

Workspace.Current.WS

The "Current" keyword can be used to reference the Workspace or Maintenance Unit, without having to fully spell out their names when they contain the object originating the call. This makes the configuration more robust because references will not break if these containers are renamed or duplicated. This syntax will still follow the same Workspace "Is Shareable" and application restrictions and behave the same as the explicit syntax.

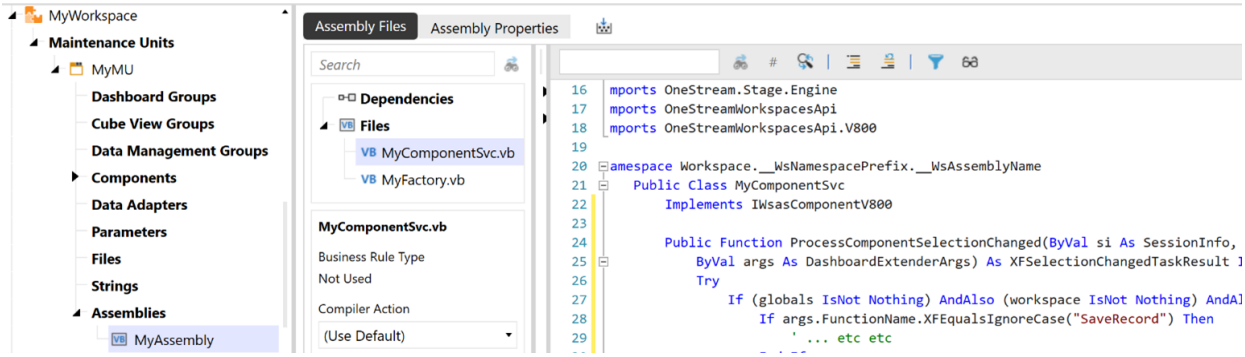
These can be used on properties of:

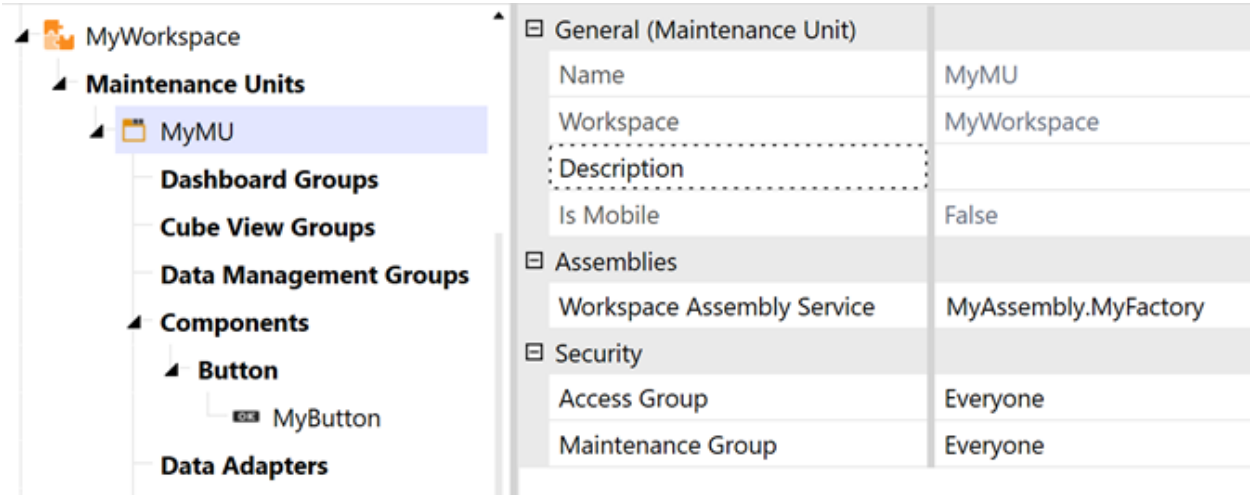
- Cube Views
- Dashboard Components
- Data Management Steps

Examples

Dashboard Components

In this example, you implemented a component service with a function name of “SaveRecord”. This service is enabled by the "MyFactory" Service Factory - Assembly File specified on the same "MyMU" Maintenance Unit within the "MyWorkspace" Workspace, which contains the button component which will invoke the function when selected on a dashboard.





In the property Selection Changed Server Task Arguments of the "MyButton" component, you can have:

{WSMU}{SaveRecord}{}

To invoke the same rule, but from a button component that lives in another Maintenance Unit named "MyMU" in the same Workspace, use:

{Workspace.Current.MyMU.WSMU}{SaveRecord}{}

To invoke the same action from a button component in another Workspace named "MyWorkspace", you must set the property of “MyWorkspace”, and change the Workspace’s “Is Shareable Workspace” field to True, and then use:

{Workspace.MyWorkspace.MyMU.WSMU}{SaveRecord}{}

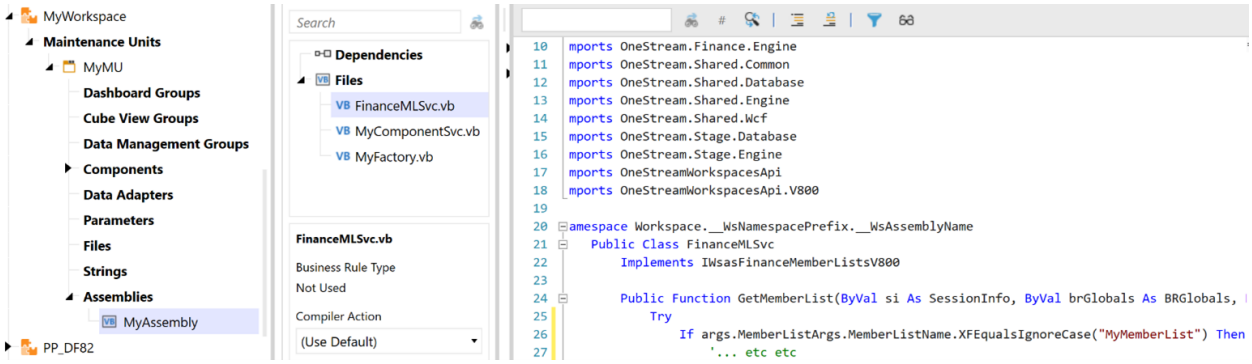
Data Management Steps

Data Management steps require the expanded syntax. For example, Workspace.MyWorkspace.MyMU.WSMU. However, to make the configuration more robust, if the factory is specified on the same Maintenance Unit that contains the step, use the "Current" keyword:

Member Lists

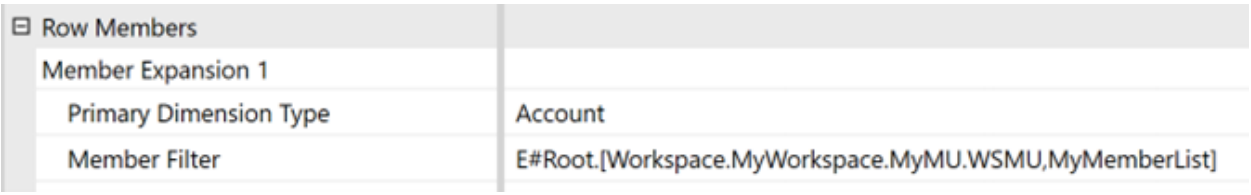
Cube Views require all referenced factories to be in a shareable Workspaces, even if Service Factories are in the same Workspace as the Cube View. The syntax must be explicit.

In this example, a Member List parameter called “MyMemberList” was created, provided by a Service Factory specified in the properties of Maintenance Unit “MyMU” in Workspace “MyWorkspace”.



To invoke it from a Cube View, use this syntax:

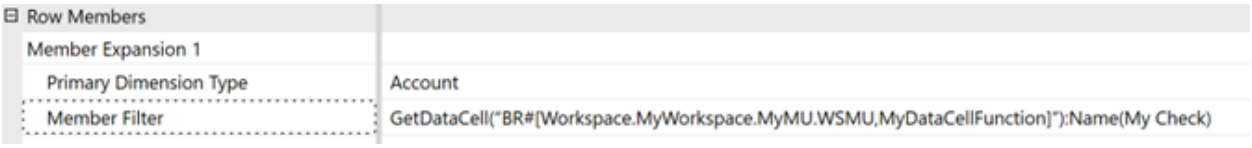
[Workspace.MyWorkspace.MyMU.WSMU,MyMemberList]



GetDataCell

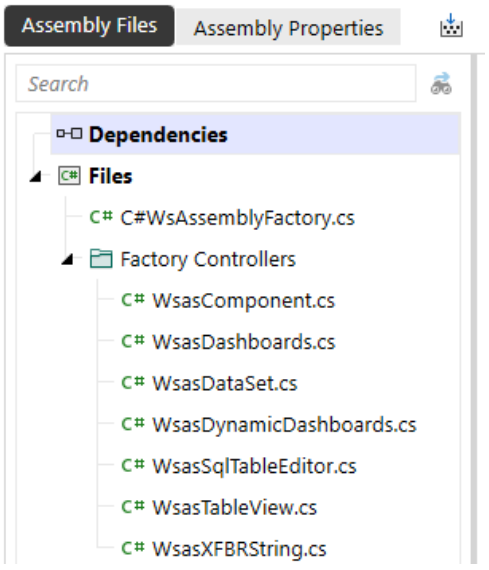
The same considerations apply as for Member Lists.

> **Example:** GetDataCell("BR#[Workspace.MyWorkspace.MyMU.WSMU,MyDataCellFunction]"):Name(My Check)



Organize Service Factory Assembly Files

With the Service Factory set up, you can build any necessary assembly files to bring the workspace to life. The image below shows a sample organization structure of assembly files.



Notice that the Service Factory assembly file is separate, and a folder holds all necessary factory controller files. The workspaces use one of each calls from the Service Factory assembly file.

While you can refer to business rules created as assemblies in other workspaces, you cannot refer to service types from other workspaces. This means a lighter syntax is required when calling business rules throughout the workspace. The following sections highlight the syntax used to reference the service types.

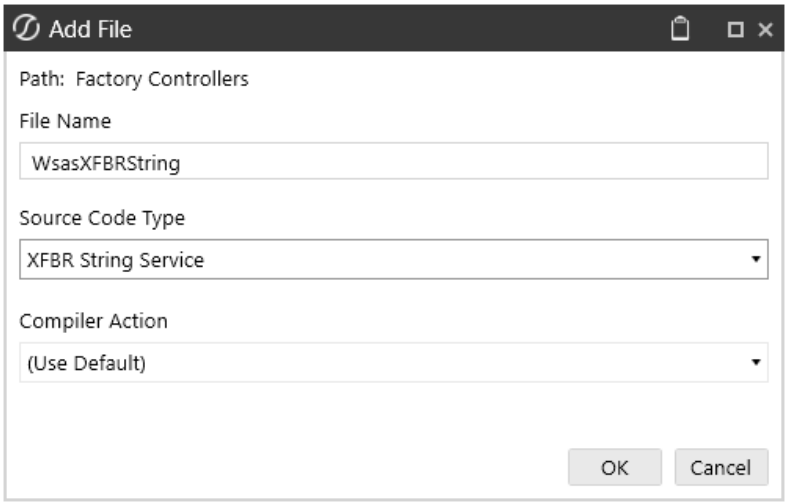
XFBRString Service Type

This service type lets you create an XFBR String within assemblies. You can do this by choosing this service type or by choosing the Dashboard String Function business rule. These rules are generally used to create effects similar to parameters and substitution variables but allow greater flexibility by using code to return string values in dashboards, cube views, and extensible documents.

You must create the XFBR service type file after creating the service factory file and after uncommenting the appropriate lines of code. The following is an example of what this looks like in C#:

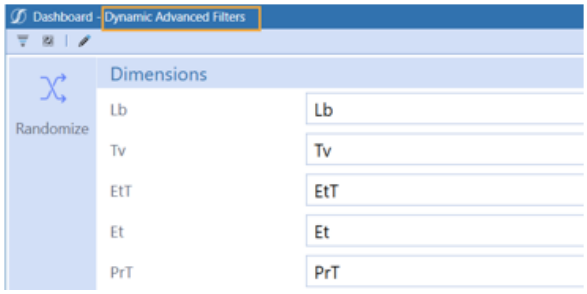
```
21              switch (WsAssemblyServiceType)
22              {
23                  //case WsAssemblyServiceType.DynamicDashboards:
24                  //    return new WsasDynamicDashboards();
25
26                  case WsAssemblyServiceType.WFBRString:
27                      return new WsasXFBRString();
28
29                  //case WsAssemblyServiceType.DataSet:
30                  //    return new WsasDataSet();
31
32                  //case WsAssemblyServiceType.Dashboard:
33                  //    return new WsasDashboard();
34
35                  //case WsAssemblyServiceType.Component:
36                  //    return new WsasComponent();
37
38                  //case WsAssemblyServiceTyle.TableView:
39                  //    return new WsasTableView();
40
41                  //case WsAssemblyServiceType.SqlTableEditor:
42                  //    return mew WsasSqlTableEditor();
43
44                  default:
45                      return null;
```

Given the changes in the code, you would configure the assembly file in the following way:

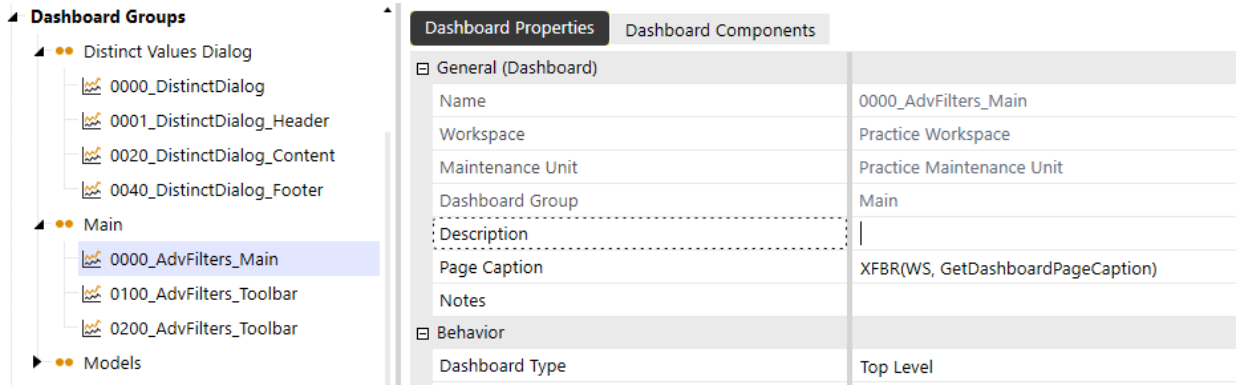


NOTE: The file name should match the return function in the Service Factory file.

The image below shows an example of an assembly file returning a page caption in a dashboard. This is commonly used to query an object or to create a dynamic page caption.



This assembly is then referenced in the Page Caption of the dashboard.



The syntax for referencing this object is XFBR(WS,FunctionName).

The syntax for using the Assembly Service on the Maintenance level is XFBR (WSMU,GetRandomText).

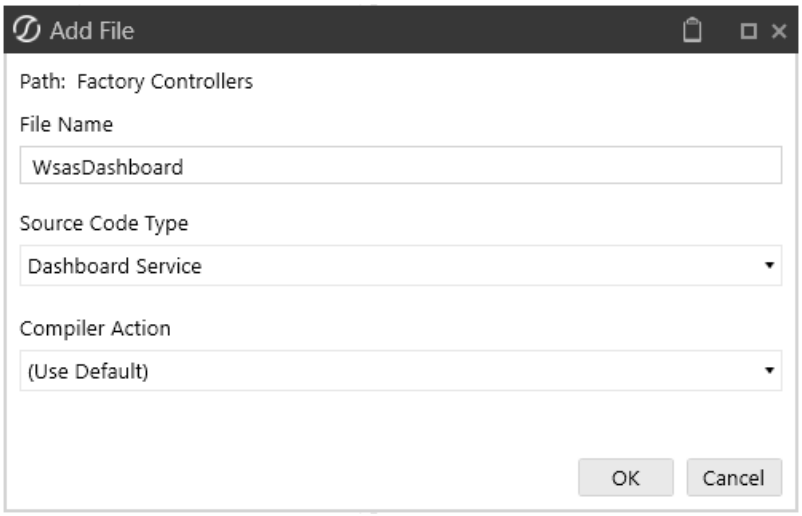
Dashboard Service Type

This service type lets you create a LoadDashboard function type in a dashboard extender rule within assemblies. You can do this by choosing this service type or by choosing the Dashboard Extender business rule type. These rules are usually used to perform custom tasks within workspaces. Developers often use the Dashboard service type with selection components, such as combo boxes to set parameters to a default value or the last selected value.

Create this file after the Service Factory file has been created and the appropriate lines of code have been uncommented. Here is an example of a dashboard service type in C#:

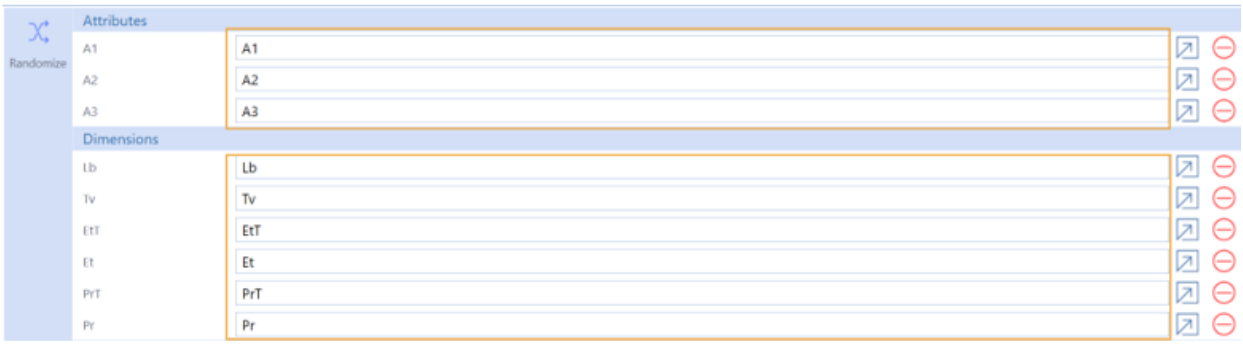
```
29              switch (wsAssemblyServiceType)
30              {
31                  //case WsAssemblyServiceType.DynamicDashboards:
32                  //    return new WsasDynamicDashboards();
33
34                  //case WsAssemblyServiceType.XFBRString:
35                  //    return new WsasXFBRString();
36
37                  //case WsAssemblyServiceType.DataSet:
38                  //    return new WsasDataSet();
39
40                  case WsAssemblyServiceType.Dashboard:
41                      return new WsasDashboard();
42
43                  //case WsAssemblyServiceType.Component:
44                  //    return new WsasComponent();
45
46                  //case WsAssemblyServiceType.TableView:
47                  //    return new WsasTableView();
48
49                  //case WsAssemblyServiceType.SqlTableEditor:
50                  //    return new WsasSqlTableEditor();
51
52                  default:
53                      return null;
```

Given the changes in the code, you would configure the dashboard service file in the following way:

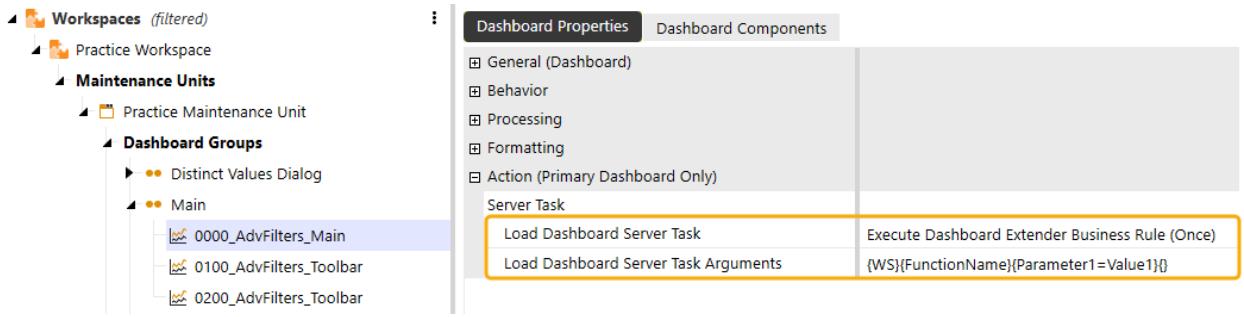


NOTE: The file name should match the return function in the Service Factory file.

The following is an example of an assembly file that is populating default values inside labels:



The labels should be populated at runtime, so the assembly file is referenced in the following manner:



The syntax for referencing this object is {WS}{FunctionName}{Parameter1=Value1}.

The syntax for using the Assembly Service on the Maintenance level is {WSMU}{FunctionName}{Parameter1=Value1}.

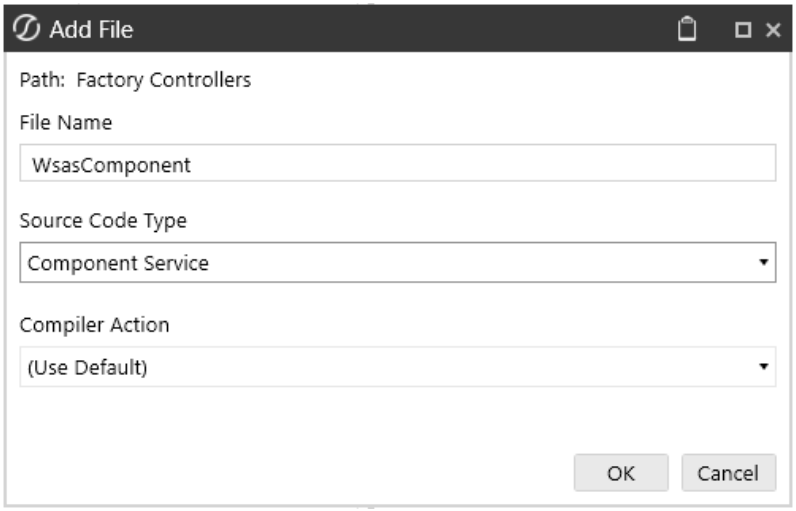
Component Service Type

This service type lets you create a ComponentSelectionChanged function type in a dashboard extender rule within assemblies. You can do this by choosing this service type or by choosing the Dashboard Extender business rule type. These rules are usually used to perform custom tasks within workspaces. Developers often use this Dashboard extender rule as it handles all actions occurring due to interactions with dashboard components, such as grids or combo boxes. These rules are referenced in the action properties of a given component.

Create this file after the Service Factory file has been created and the appropriate lines of code have been uncommented. Here is an example of a dashboard service type in C#:

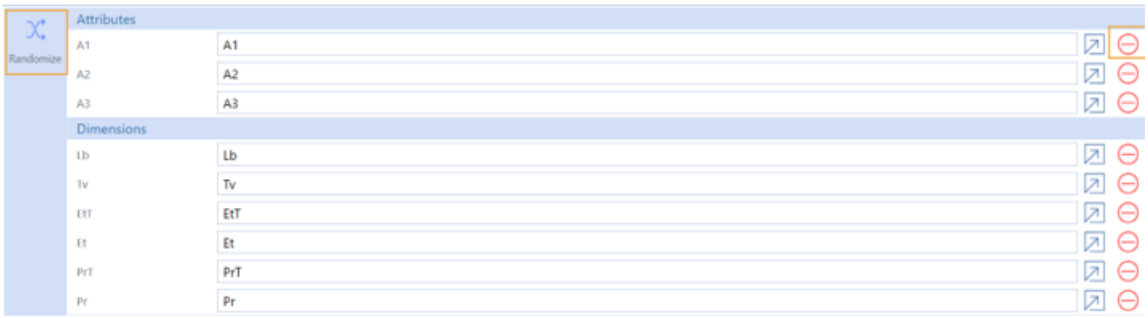
```
29                  switch (wsAssemblyServiceType)
30                  {
31                      //case WsAssemblyServiceType.DynamicDashboards:
32                      //    return new WsasDynamicDashboards();
33
34                      //case WsAssemblyServiceType.XFBRString:
35                      //    return new WsasXFBRString();
36
37                      //case WsAssemblyServiceType.DataSet:
38                      //    return new WsasDataSet();
39
40                      //case WsAssemblyServiceType.Dashboard:
41                      //    return new WsasDashboard();
42
43                      case WsAssemblyServiceType.Component:
44                          return new WsasComponent();
45
46                      //case WsAssemblyServiceType.TableView:
47                      //    return new WsasTableView();
48
49                      //case WsAssemblyServiceType.SqlTableEditor:
50                      //    return new WsasSqlTableEditor();
51
52                      default:
53                          return null;
```

Given the changes in the code, you would configure the dashboard service file in the following way:

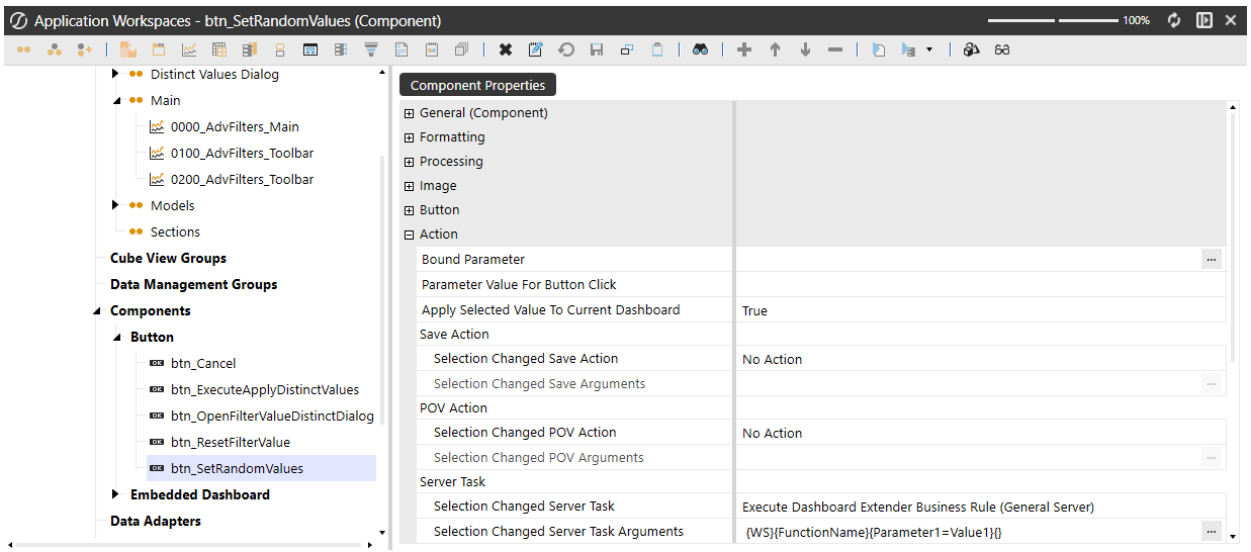


NOTE: The file name should match the return function in the Service Factory file.

The following is an example of an assembly file that is operating the two highlighted icons in the dashboard:



The Randomize icon fills the labels with random values, while the second icon resets the values to their original state. The logic, populated within the assembly file, runs when you click a button.



The syntax for referencing this object is {WS}{FunctionName}{Parameter1=Value1}.

The syntax for using the Assembly Service on the Maintenance level is {WSMU}{FunctionName} {Parameter1=Value1}.

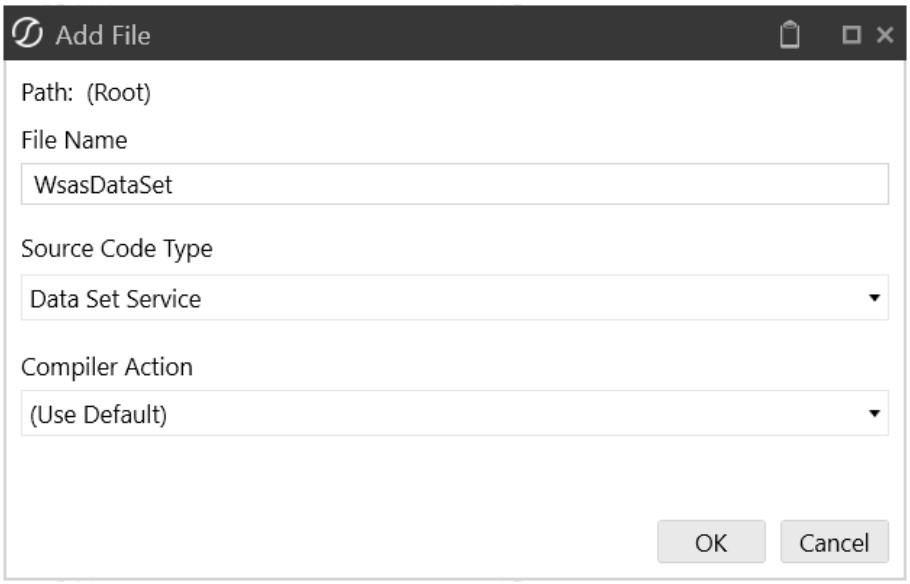
Data Set Service Type

This service type lets you create Data Set logic within assemblies. You can do this by choosing this service type or by choosing the Dashboard Data Set business rule type. These rules are usually queries and return data to data adapters and parameters using a method query.

Create this file after creating the Service Factory file and after the appropriate lines of code have been uncommented. Here is an example of a data set service type in C#:

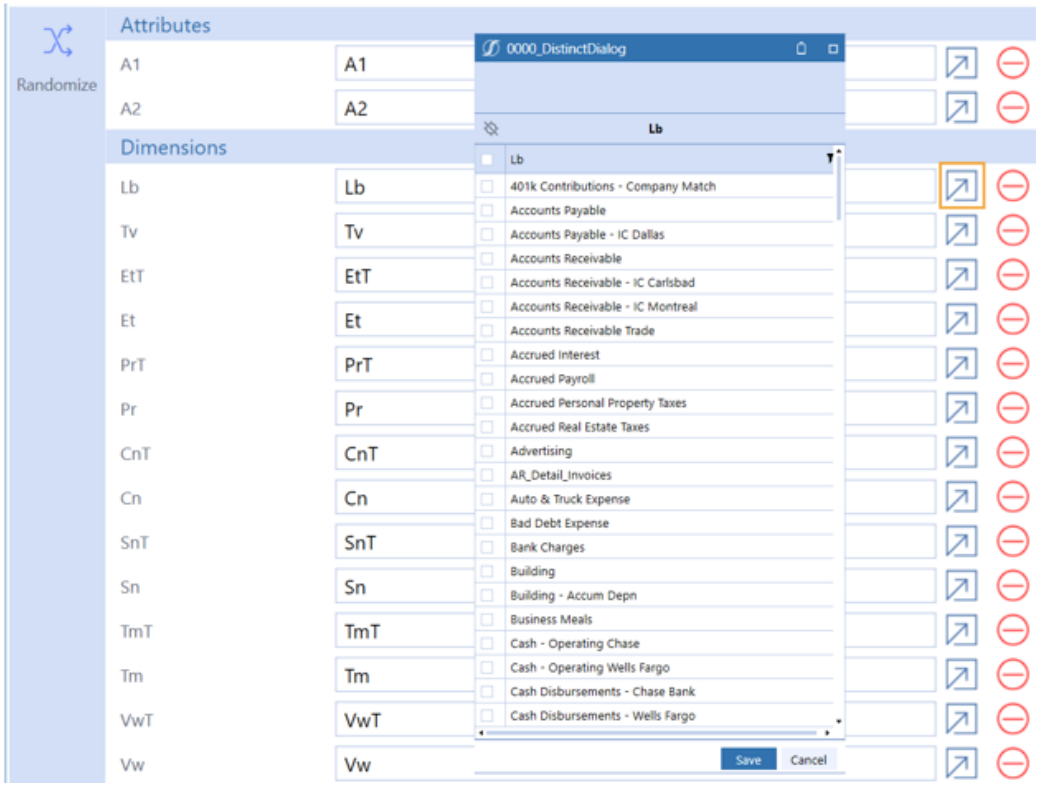
```
29              switch (wsAssemblyServiceType)
30              {
31                  //case WsAssemblyServiceType.DynamicDashboards:
32                  //    return new WsasDynamicDashboards();
33
34                  //case WsAssemblyServiceType.XFBRString:
35                  //    return new WsasXFBRString();
36
37                  case WsAssemblyServiceType.DataSet:
38                      return new WsasDataSet();
39
40                  //case WsAssemblyServiceType.Dashboard:
41                  //    return new WsasDashboard();
42
43                  //case WsAssemblyServiceType.Component:
44                  //    return new WsasComponent();
45
46                  //case WsAssemblyServiceType.TableView:
47                  //    return new WsasTableView();
48
49                  //case WsAssemblyServiceType.SqlTableEditor:
50                  //    return new WsasSqlTableEditor();
51
52                  default:
53                      return null;
```

Given the changes in the code, you would configure the dashboard service file in the following way:

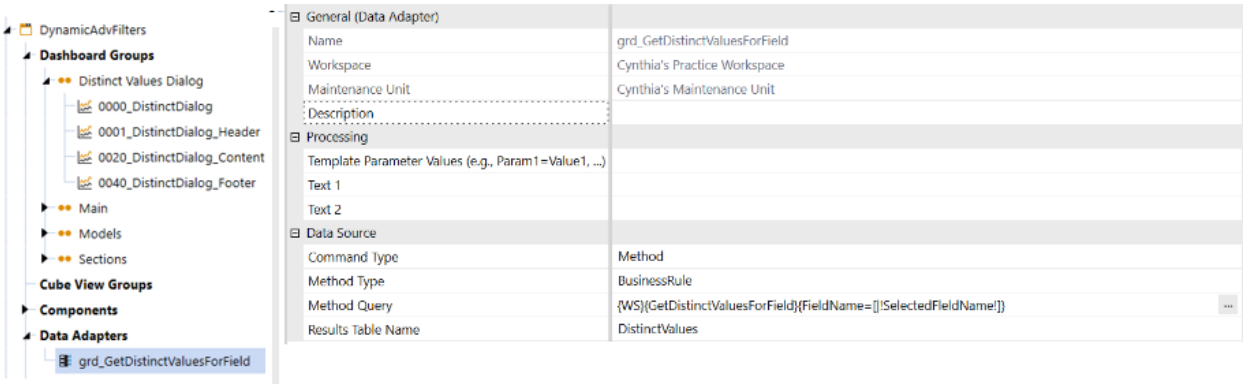


NOTE: The file name should match the return function in the Service Factory file.

The following is an example of an assembly file that creates a grid of potential values you might see based on a selection. The assembly file activates when you click on an icon.



This runs through a method query data adapter that populates the grid component.



The syntax for referencing this object is {WS}{DataSet}{Parameter1=Value1}.

The syntax for using the Assembly Service on the Maintenance level is {WSMU}{DataSet} {Parameter1=Value1}.

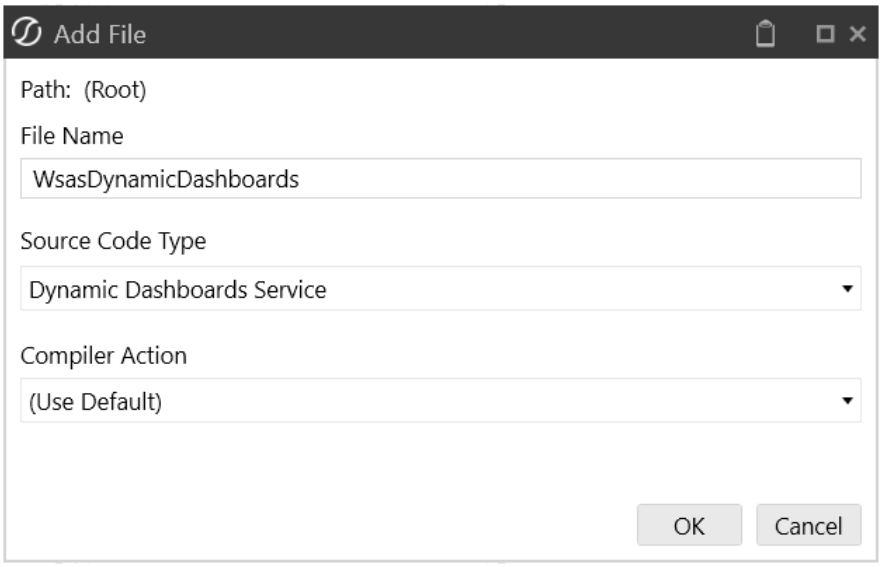
Dynamic Dashboards Service Type

Dynamic Dashboards allow you to modify components within the dashboard as well as modifying the dashboard itself.

Create a Dynamic Dashboard Service Type assembly file after the Service Factory file has been created. Here is an example of a Service Factory file referencing a Dynamic Dashboard Service type in C#:

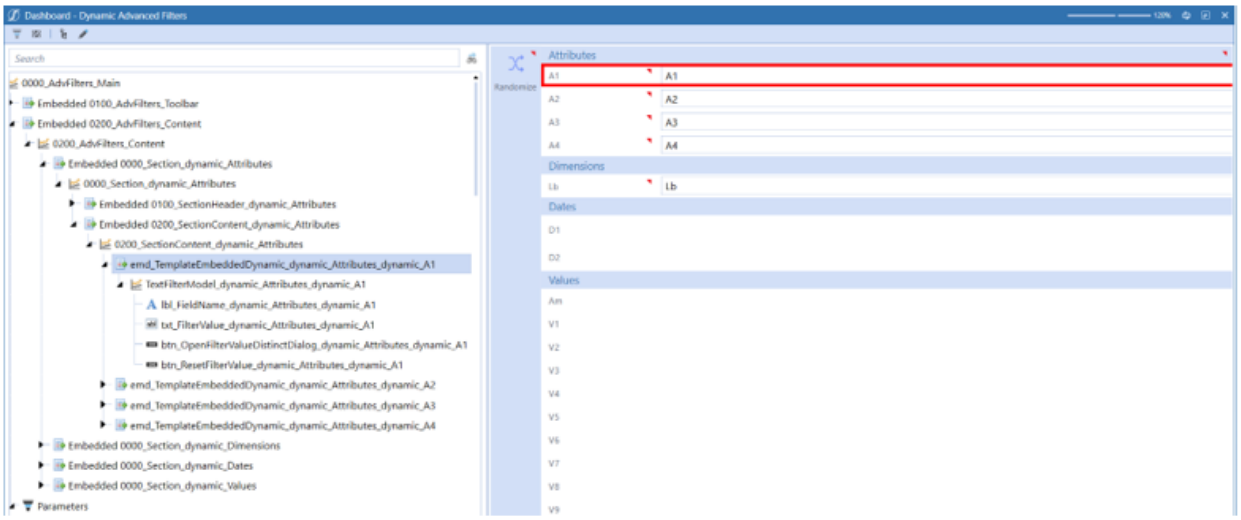
```
29              switch (wsAssemblyServiceType)
30              {
31                  case WsAssemblyServiceType.DynamicDashboards:
32                      return new WsasDynamicDashboards();
33
34                  //case WsAssemblyServiceType.XFBRString:
35                  //    return new WsasXFBRString();
36
37                  //case WsAssemblyServiceType.DataSet:
38                  //    return new WsasDataSet();
39
40                  //case WsAssemblyServiceType.Dashboard:
41                  //    return new WsasDashboard();
42
43                  //case WsAssemblyServiceType.Component:
44                  //    return new WsasComponent();
45
46                  //case WsAssemblyServiceType.TableView:
47                  //    return new WsasTableView();
48
49                  //case WsAssemblyServiceType.SqlTableEditor:
50                  //    return new WsasSqlTableEditor();
51
52                  default:
53                      return null;
```

Given the changes in the code, configure the dashboard service file in the following way:



NOTE: The file name should match the return function in the Service Factory file.

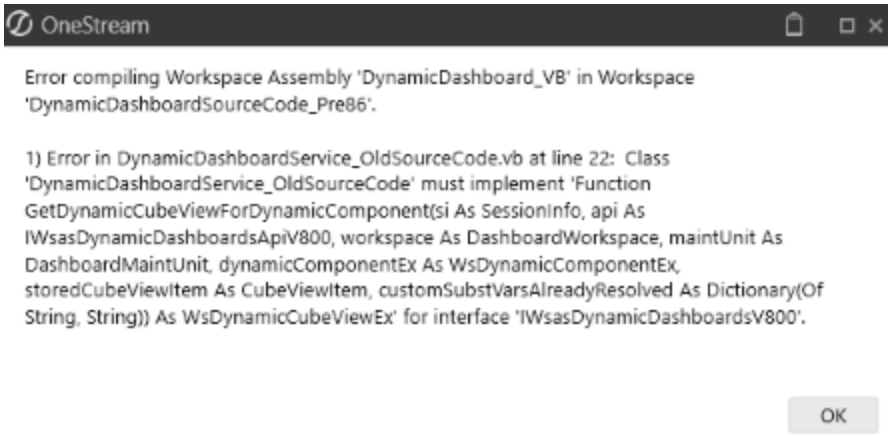
The example below shows embedded components that have not been created manually in the workspaces interface but are generated by the assembly file.



Before writing the assembly, ensure the Dashboard Type property is set to **Embedded Dynamic**. The rule will not run if you forget this step. This is also where you should ensure the Workspace Assembly Service property is set to the Service Factory assembly file.

There is no special syntax required for referencing this service type.

If you created Dynamic Dashboard Service Type Assembly Files using VB.Net before version 9.0, you need to manually add the new GetDynamicCubeViewForDynamicComponent function to the source code or you will receive this error:



To update and compile pre-Version 9.0 VB.Net Dynamic Dashboard Service files, complete the following steps:

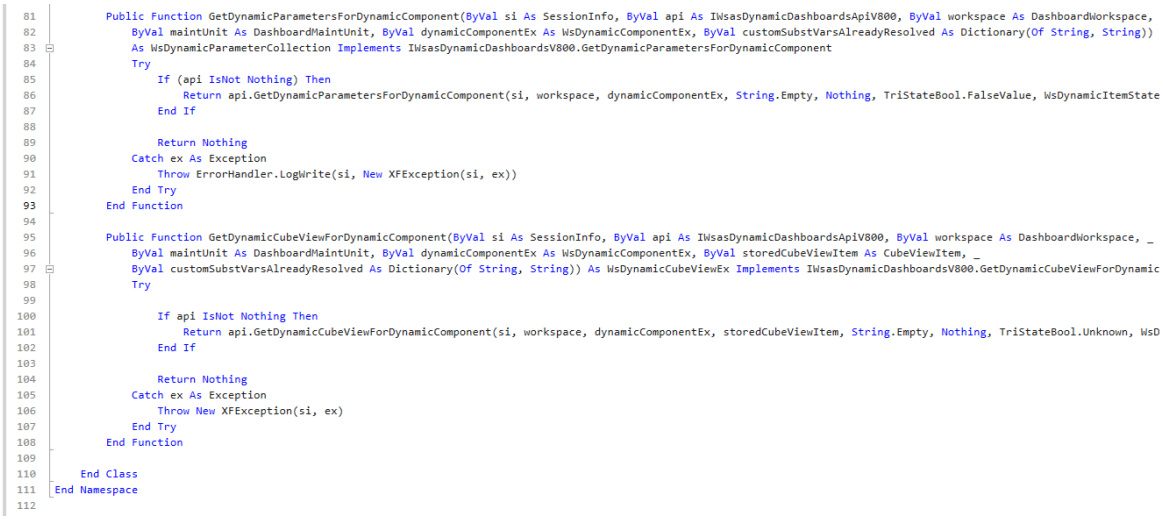
1. Copy the following lines of code to add the GetDynamicCubeViewForDynamicComponent method.

GetDynamicCubeViewForDynamicComponent VB.Net Code

```
Public Function GetDynamicCubeViewForDynamicComponent(ByVal si As SessionInfo, ByVal
api As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
        ByVal maintUnit As DashboardMaintUnit, ByVal dynamicComponentEx As
WsDynamicComponentEx, ByVal storedCubeViewItem As CubeViewItem, _
        ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As
WsDynamicCubeViewEx Implements
IWsasDynamicDashboardsV800.GetDynamicCubeViewForDynamicComponent
```

```
        Try

            If api IsNot Nothing Then
                Return api.GetDynamicCubeViewForDynamicComponent(si, workspace,
dynamicComponentEx, storedCubeViewItem, String.Empty, Nothing, TriStateBool.Unknown,
WsDynamicItemStateType.Unknown)
            End If

            Return Nothing
        Catch ex As Exception
            Throw New XFException(si, ex)
        End Try
    End Function
```

2. Paste the code in the Dynamic Dashboard Service File that is producing the error. You can paste the full function after the GetDynamicParametersForDynamicComponent method. This is after line 93 in the original source code.



```
81          Public Function GetDynamicParametersForDynamicComponent(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace,
82              ByVal maintUnit As DashboardMaintUnit, ByVal dynamicComponentEx As WsDynamicComponentEx, ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String))
83              As WsDynamicParameterCollection Implements IWsasDynamicDashboardsV800.GetDynamicParametersForDynamicComponent
84              Try
85                  If (api IsNot Nothing) Then
86                      Return api.GetDynamicParametersForDynamicComponent(si, workspace, dynamicComponentEx, String.Empty, Nothing, TriStateBool.FalseValue, WsDynamicItemState
87                  End If
88
89                  Return Nothing
90              Catch ex As Exception
91                  Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
92              End Try
93          End Function
94
95          Public Function GetDynamicCubeViewForDynamicComponent(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
96              ByVal maintUnit As DashboardMaintUnit, ByVal dynamicComponentEx As WsDynamicComponentEx, ByVal storedCubeViewItem As CubeViewItem, _
97              ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As WsDynamicCubeViewEx Implements IWsasDynamicDashboardsV800.GetDynamicCubeViewForDynamic
98              Try
99
100                 If api IsNot Nothing Then
101                     Return api.GetDynamicCubeViewForDynamicComponent(si, workspace, dynamicComponentEx, storedCubeViewItem, String.Empty, Nothing, TriStateBool.Unknown, WsD
102                 End If
103
104                 Return Nothing
105             Catch ex As Exception
106                 Throw New XFException(si, ex)
107             End Try
108         End Function
109
110     End Class
111 End Namespace
112
```

3. Select **Compile Assembly** to compile the Assembly File.

> **IMPORTANT:** You can utilize new functionalities within Platform Release 8.2 and later versions for service types in C#. While new coding options will work successfully in C#, these new functionalities in Platform Release 8.2 or later may not be compatible with VB.Net.

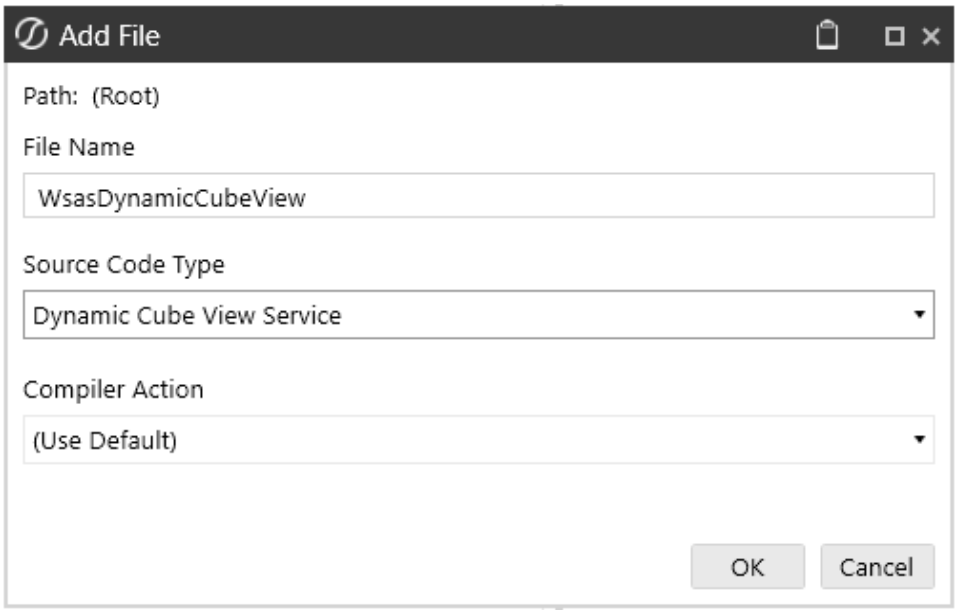
Dynamic Cube View Service Type

This service type enables you to manipulate Cube View formatting and data at runtime by modifying the CubeViewItem object. Create a Dynamic Cube View Service type assembly file after the Service Factory has been created. The following example is a Service Factory file with the case statement for Dynamic Cube Views uncommented. See Implementing Dynamic Cube View Services.

```
24          public IWsAssemblyServiceBase CreateWsAssemblyServiceInstance(
25              DashboardWorkspace workspace, WsAssemblyServiceType wsAsse
26          {
27              try
28              {
29                  switch (wsAssemblyServiceType)
30                  {
31                      //case WsAssemblyServiceType.Component:
32                      //    return new WsasComponent();
33
34                      //case WsAssemblyServiceType.Dashboard:
35                      //    return new WsasDashboard();
36
37                      //case WsAssemblyServiceType.DataManagementStep:
38                      //    return new WsasDataManagementStep();
39
40                      //case WsAssemblyServiceType.DataSet:
41                      //    return new WsasDataSet();
42
43                      //case WsAssemblyServiceType.DynamicDashboards:
44                      //    return new WsasDynamicDashboards();
45
46                      case WsAssemblyServiceType.DynamicCubeView:
47                          return new WsasDynamicCubeView();
48
49                      //case WsAssemblyServiceType.FinanceCore:
50                      //    return new WsasFinanceCore();
```

Given the changes in the code, configure the assembly file in the following way from the **Add File** window:

1. In the **File Name** field, enter a file name.
2. From the **Source Code Type** drop-down menu, select **Dynamic Cube View Service**.
3. From the **Compiler Action** drop-down menu, select a compiler action. You can choose to Compile or Disable the file. Selecting Disable ignores the file when the assembly is compiled. (Use Default) is the same as Compile.



NOTE: The file name should match the return function in the Service Factory file.

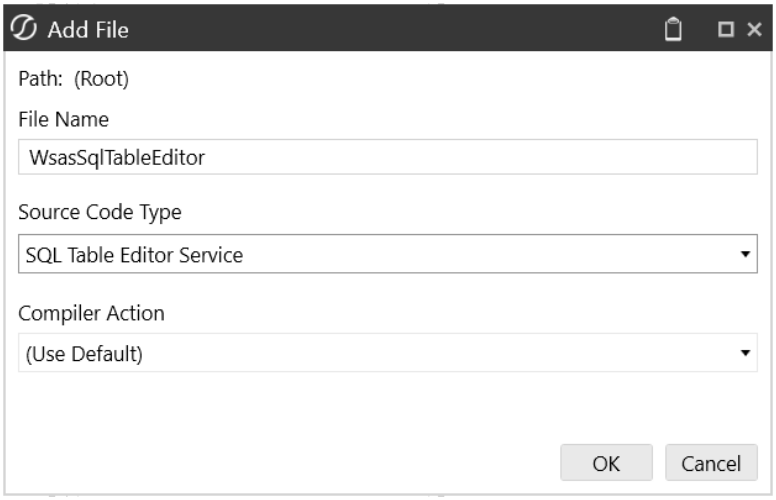
SQL Table Editor Service Type

This service type lets you create SqlTableEditorSaveData function type in a dashboard extender rule within assemblies. You can do this either by choosing this service type or by choosing the Dashboard Extender business rule type. These rules are generally used to perform custom tasks in workspaces, such as saving actions within the SQL Table Editor component.

Create this file after the Service Factory file has been created and after the appropriate lines of code have been uncommented. Here is an example of a data set service type in C#:

```
29              switch (wsAssemblyServiceType)
30              {
31                  //case WsAssemblyServiceType.DynamicDashboards:
32                  //    return new WsasDynamicDashboards();
33
34                  //case WsAssemblyServiceType.XFBRString:
35                  //    return new WsasXFBRString();
36
37                  //case WsAssemblyServiceType.DataSet:
38                  //    return new WsasDataSet();
39
40                  //case WsAssemblyServiceType.Dashboard:
41                  //    return new WsasDashboard();
42
43                  //case WsAssemblyServiceType.Component:
44                  //    return new WsasComponent();
45
46                  //case WsAssemblyServiceType.TableView:
47                  //    return new WsasTableView();
48
49                  case WsAssemblyServiceType.SqlTableEditor:
50                      return new WsasSqlTableEditor();
51
52                  default:
53                      return null;
```

Given the changes in the code, you would configure the assembly file in the following way:



NOTE: The file name should match the return function in the Service Factory file.

The syntax for referencing this object is {WS}{FunctionName}{Parameter1=Value1}, similar to the Dashboard service type and the Component service type.

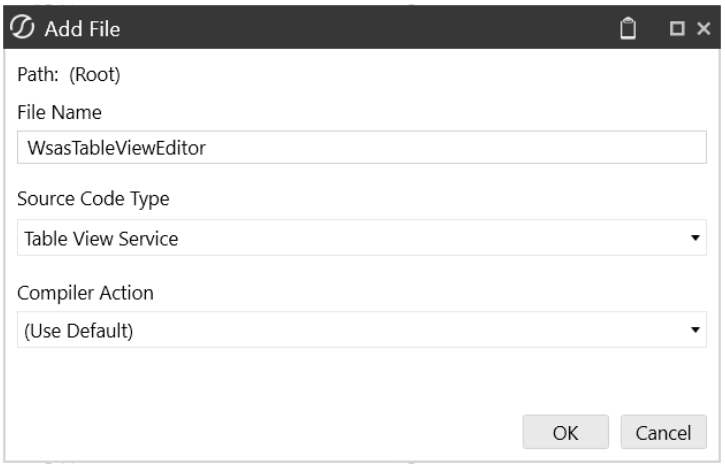
Table View Service Type

This service type lets you create SqlTableEditorSaveData function type in a dashboard extender rule within assemblies. You can do this either by choosing this service type or by choosing the Spreadsheet business rule type. These rules query and edit data that does not live within a cube in the spreadsheet or the Excel Add-In.

Create this file after the Service Factory file has been created and after the appropriate lines of code have been uncommented. Here is an example of a data set service type in C#:

```
29              switch (wsAssemblyServiceType)
30              {
31                  //case WsAssemblyServiceType.DynamicDashboards:
32                  //    return new WsasDynamicDashboards();
33
34                  //case WsAssemblyServiceType.XFBRString:
35                  //    return new WsasXFBRString();
36
37                  //case WsAssemblyServiceType.DataSet:
38                  //    return new WsasDataSet();
39
40                  //case WsAssemblyServiceType.Dashboard:
41                  //    return new WsasDashboard();
42
43                  //case WsAssemblyServiceType.Component:
44                  //    return new WsasComponent();
45
46                  case WsAssemblyServiceType.TableView:
47                      return new WsasTableView();
48
49                  //case WsAssemblyServiceType.SqlTableEditor:
50                  //    return new WsasSqlTableEditor();
51
52                  default:
```

Given the changes in the code, you would configure the assembly file in the following way:



> **NOTE:** The file name should match the return function in the Service Factory file.

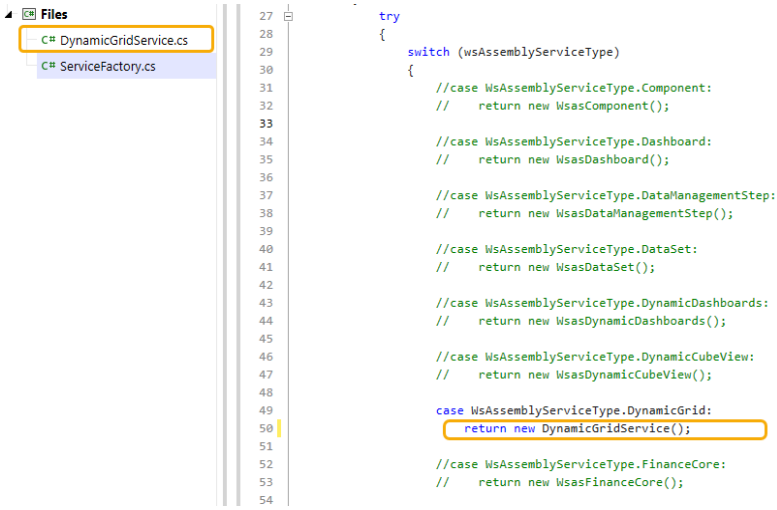
The syntax for referencing this object is Workspace.WorkspaceName.AssemblyName.FileName.

Dynamic Grid Service Type

Dynamic Grids enable you to read, write, and transform data from various sources, offering a flexible and high-performance solution for managing large datasets.

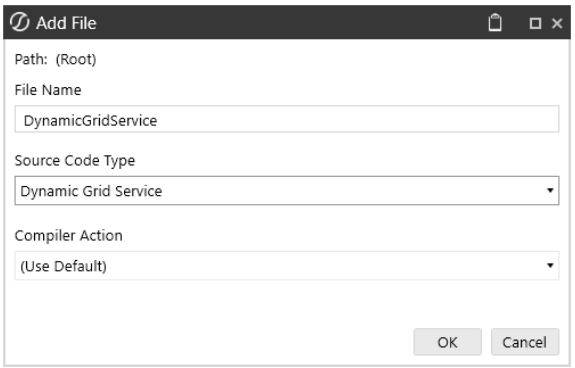
Create the Service Factory Assembly File and then the Dynamic Grid Service Assembly File. Then, uncomment the appropriate lines of code. The following example is in the Service Factory Assembly File with the lines for the Dynamic Grid uncommented and highlighted.

> **NOTE:** The file name must match the name of the return function in the Service Factory file.



Configure the Dynamic Grid Service Assembly File with the following options:

- In the **File Name** field, enter a name for the file (for example, DynamicGridService).
- In the **Source Code Type** drop-down menu, select **Dynamic Grid Service**.
- In the **Compiler Action** drop-down menu, select **(Use Default)**.



There is no special syntax required for referencing this service type.

See Dynamic Grid and Dynamic Grid Service.

API Reference

The following common arguments are accepted or made available by many functions.

Object	Object Type	Description
si	SessionInfo	Generic information about the current user session, including the type of client. See the *API Overview Guide* for details.
brGlobals	BRGlobals	Generic object to store and retrieve data between calls in the same execution thread.See the *API Overview Guide* for details.
args	DashboardDataSetArgs, DashboardExtenderArgs, ExtenderArgs	Run-specific context properties.
workspace	DashboardWorkspace	The Workspace containing the object that is being processed.
maintUnit	DashboardMaintUnit	The Maintenance Unit containing the object that is being processed.
dynamicDashboardEx	WsDynamicDashboardEx	The dashboard that is being processed.
dynamicComponentEx	WsDynamicComponentEx	The component that is being processed.
dynamicAdapterEx	WsDynamicAdapterEx	The adapter that is being

Object	Object Type	Description
		processed.
customSubstVarsAlreadyResolved	Dictionary (Of String, String)	Populated dashboard Parameters.
nextLevelNameSuffixToAdd	String	In a Repeater configuration, this string is attached to the name of the original component to create the dynamic instance. For example, generated instances are called <originalName>_ dynamic_ <nextLevelNameSuffixToAdd>.
nextLevelTemplateSubstVarsToAdd	Dictionary (Of String, String)	In a Repeater configuration, this dictionary contains the Template Parameters available to repeated objects.
convertStoredComponentToDynamic	TriStateBool	To mark the original component as a dynamic version.
dynamicItemStateType	WsDynamicItemStateType	How an object is persisted when saving or replicating state across servers.

Service Factory

The Service Factory class must implement interface IWsAssemblyServiceFactory and contain the following CreateWsAssemblyServiceInstance method:

C# Code

```
 public IWsAssemblyServiceBase CreateWsAssemblyServiceInstance(SessionInfo si, BRGlobals
brGlobals,
            DashboardWorkspace workspace, WsAssemblyServiceType wsAssemblyServiceType,
string itemName)
```

VB.Net Code

```
Public Function CreateWsAssemblyServiceInstance(ByVal si As SessionInfo, ByVal globals As
BRGlobals, ByVal workspace As DashboardWorkspace, ByVal wsasType As WsAssemblyServiceType,
ByVal itemName As String) As IWsAssemblyServiceBase Implements
IWsAssemblyServiceFactory.CreateWsAssemblyServiceInstance
```

The object returned must implement interface IWsAssemblyServiceBase. All supported service types inherit the IWsAssemblyServiceBase interface through their specific interfaces.

wsasType
The type of service that the Service Factory expects to provide. This is used to determine which Service object to return.

itemName
This contains the name of the invoked function if it is present. For example, a Data Set name or a Custom Calculation Function name.

Component Service

The Component Service responds to component-triggered actions in dashboards. It is similar to a feature found in traditional Dashboard Extenders, such as DashboardExtenderFunctionType.ComponentSelectionChanged.

It must implement interface IWsasComponentV800, which requires the following method:

C# Code

```
public XFSelectionChangedTaskResult ProcessComponentSelectionChanged(SessionInfo si,
BRGlobals brGlobals, DashboardWorkspace workspace, DashboardExtenderArgs args)
```

VB.Net Code

```
Public Function ProcessComponentSelectionChanged(ByVal si As SessionInfo, ByVal globals As
BRGlobals, ByVal workspace As DashboardWorkspace, _
            ByVal args As DashboardExtenderArgs) As XFSelectionChangedTaskResult Implements
IWsasComponentV800.ProcessComponentSelectionChanged
```

Dashboard Service

The Dashboard Service responds to Dashboard-triggered actions. It is similar to a feature found in traditional Dashboard Extenders, for example DashboardExtenderFunctionType.LoadDashboard.

It must implement interface IWsasDashboardV800, which requires the following method:

C# Code

```
public XFLoadDashboardTaskResult ProcessLoadDashboardTask(SessionInfo si, BRGlobals
brGlobals, DashboardWorkspace workspace, DashboardExtenderArgs args)
```

VB.Net Code

```
 Public Function ProcessLoadDashboardTask(ByVal si As SessionInfo, ByVal globals As
BRGlobals, ByVal workspace As DashboardWorkspace, _
            ByVal args As DashboardExtenderArgs) As XFLoadDashboardTaskResult Implements
IWsasDashboardV800.ProcessLoadDashboardTask
```

Data Management Step Service

The Data Management Step Service runs when invoked by a Data Management Step of an Execute Business Rule type. This is similar to a feature found in traditional Extensibility rules, for example ExtenderFunctionType.ExecuteDataMgmtBusinessRuleStep.

It must implement interface IWsasDataManagementStepV800, which requires the following method:

C# Code

```
public void ProcessDataManagementStep(SessionInfo si, BRGlobals brGlobals,
DashboardWorkspace workspace, ExtenderArgs args)
```

VB.Net Code

```
 Public Sub ProcessDataManagementStep(ByVal si As SessionInfo, ByVal brGlobals As BRGlobals,
ByVal workspace As DashboardWorkspace, _
            ByVal args As ExtenderArgs) Implements
IWsasDataManagementStepV800.ProcessDataManagementStep
```

Data Set Service

The Data Set Service returns a Data Set or DataTable object, often to be processed by a Data Adapter or Bound List Parameter. It is similar to a feature found in traditional Dashboard Data Set rules, for example, DashboardDataSetFunctionType.GetDataSet.

It must implement interface IWsasDataSetV800, which requires the following method:

C# Code

```
 public object GetDataSet(SessionInfo si, BRGlobals brGlobals, DashboardWorkspace workspace,
DashboardDataSetArgs args)
```

VB.Net Code

```
Public Function GetDataSet(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal
workspace As DashboardWorkspace, _
            ByVal args As DashboardDataSetArgs) As Object Implements
IWsasDataSetV800.GetDataSet
```

Dynamic Grid Service

This service type feeds data to the Dynamic Grid Component. It implements interface IWsasDynamicGridV800, which has the following two methods:

This method is called whenever the dynamic grid requests data, such as on the initial page load.

C# Code

```
public XFDynamicGridGetDataResult GetDynamicGridData(SessionInfo si, BRGlobals brGlobals,
DashboardWorkspace workspace, DashboardDynamicGridArgs args)
```

VB.Net Code

```
Public Function GetDynamicGridData(ByVal si As SessionInfo, ByVal brGlobals As BRGlobals,
ByVal workspace As DashboardWorkspace, ByVal args As DashboardDynamicGridArgs) As
XFDynamicGridGetDataResult Implements IWsasDynamicGridV800.GetDynamicGridData
```

DashboardDynamicGridArgs has everything needed to request data, such as start row index, page size, and column filters.

This method is called whenever the user clicks the Save button on the toolbar to save modified rows.

C# Code

```
public XFDynamicGridSaveDataResult SaveDynamicGridData(SessionInfo si, BRGlobals brGlobals,
DashboardWorkspace workspace, DashboardDynamicGridArgs args)
```

VB.Net Code

```
Public Function SaveDynamicGridData(ByVal si As SessionInfo, ByVal brGlobals As BRGlobals,
ByVal workspace As DashboardWorkspace, ByVal args As DashboardDynamicGridArgs) As
XFDynamicGridSaveDataResult Implements IWsasDynamicGridV800.SaveDynamicGridData
```

The DashboardDynamicGridArgs has everything needed to save data, including a list of modified rows.

Dynamic Cube View Service

This service type enables you to manipulate Cube View formatting and data at runtime by modifying the CubeViewItem object. It must implement interface IWsasDynamicCubeViewV800, which requires the following method:

C# Code

```
public CubeViewItem GetDynamicCubeViewItem(SessionInfo si, IWsasDynamicCubeViewApiV800 api,
DashboardWorkspace workspace, CubeViewItem cubeViewItem, DynamicCubeViewArgs args)
```

VB.Net Code

```
Public Function GetDynamicCubeViewItem(ByVal si As SessionInfo, ByVal api As
IWsasDynamicCubeViewApiV800, ByVal workspace As DashboardWorkspace, _ ByVal cubeViewItem As
CubeViewItem, ByVal args As DynamicCubeViewArgs) As CubeViewItem Implements
IWsasDynamicCubeViewV800.GetDynamicCubeViewItem
```

Dynamic Dashboard Service

The Dynamic Dashboard Service is responsible for preprocessing dashboards at runtime. It provides several hooks to manipulate objects created by dashboards of an Embedded Dynamic Dashboard type or an Embedded Dynamic Repeater Dashboard type. For more information on general strategies, see Dynamic Dashboards

It must implement interface IWsasDynamicDashboardV800, which requires the following method:

C# Code

```
 public WsDynamicDashboardEx GetEmbeddedDynamicDashboard(SessionInfo si,
IWsasDynamicDashboardsApiV800 api, DashboardWorkspace workspace, DashboardMaintUnit
maintUnit,
            WsDynamicComponentEx parentDynamicComponentEx, Dashboard storedDashboard,
Dictionary<string, string> customSubstVarsAlreadyResolved)
```

VB.Net Code

```
        Public Function GetEmbeddedDynamicDashboard(ByVal si As SessionInfo, ByVal api As
IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal parentDynamicComponentEx As
WsDynamicComponentEx, ByVal storedDashboard As Dashboard, _
            ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As
WsDynamicDashboardEx Implements IWsasDynamicDashboardsV800.GetEmbeddedDynamicDashboard
```

This method returns an object representing a completed dashboard of an Embedded Dynamic Dashboard type or an Embedded Dynamic Repeater Dashboard type. Use this function to modify dashboard-specific properties, such as layout or data used by Repeaters. It is considered similar to the LoadDashboard approach of traditional Dashboard Extenders with these significant differences:

- The method will always be triggered, regardless of embedded status.
- The method will trigger before any other event is processed. For example, before LoadDashboard extenders would run.
- The API allows for extensive access to Dashboard Properties.

parentDynamicComponentEx
The dashboard that will embed the dashboard generated by this function.

storedDashboard
The stored dashboard from which the function will generate a dynamic instance.

Implement the GetDynamicComponentsForDynamicDashboard method to modify any component property. This method returns a collection of dynamic components to use on an Embedded Dynamic Dashboard type or an Embedded Dynamic Repeater Dashboard type.

C# Code

```
public WsDynamicComponentCollection GetDynamicComponentsForDynamicDashboard(SessionInfo si,
IWsasDynamicDashboardsApiV800 api, DashboardWorkspace workspace,
            DashboardMaintUnit maintUnit, WsDynamicDashboardEx dynamicDashboardEx,
Dictionary<string, string> customSubstVarsAlreadyResolved)
```

VB.Net Code

```
Public Function GetDynamicComponentsForDynamicDashboard(ByVal si As SessionInfo, ByVal api
As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal dynamicDashboardEx As
WsDynamicDashboardEx, ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String))
_
            As WsDynamicComponentCollection Implements
IWsasDynamicDashboardsV800.GetDynamicComponentsForDynamicDashboard
```

Implement the GetDynamicAdaptersForDynamicComponent method to modify any adapter. This method returns a collection of Data Adapters to use on an Embedded Dynamic Dashboard type or an Embedded Dynamic Repeater Dashboard type.

C# Code

```
  public WsDynamicAdapterCollection GetDynamicAdaptersForDynamicComponent(SessionInfo si,
IWsasDynamicDashboardsApiV800 api, DashboardWorkspace workspace,
            DashboardMaintUnit maintUnit, WsDynamicComponentEx dynamicComponentEx,
Dictionary<string, string> customSubstVarsAlreadyResolved)
```

VB.Net Code

```
    Public Function GetDynamicAdaptersForDynamicComponent(ByVal si As SessionInfo, ByVal api
As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal dynamicComponentEx As
WsDynamicComponentEx, ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String))
_
            As WsDynamicAdapterCollection Implements
IWsasDynamicDashboardsV800.GetDynamicAdaptersForDynamicComponent
```

Implement the GetDynamicCubeViewForDynamicAdapter method to modify the Cube View configuration, for example, changing row filters. This method is used to return a Cube View that will be used by a Data Adapter on an Embedded Dynamic Dashboard type or an Embedded Dynamic Repeater Dashboard type.

storedCubeViewItem

The stored Cube View referenced by the Adapter configuration.

C# Code

```
  public WsDynamicCubeViewEx GetDynamicCubeViewForDynamicAdapter(SessionInfo si,
IWsasDynamicDashboardsApiV800 api, DashboardWorkspace workspace,
            DashboardMaintUnit maintUnit, WsDynamicAdapterEx dynamicAdapterEx, CubeViewItem
storedCubeViewItem, Dictionary<string, string> customSubstVarsAlreadyResolved)
```

VB.Net Code

```
    Public Function GetDynamicCubeViewForDynamicAdapter(ByVal si As SessionInfo, ByVal api
As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal dynamicAdapterEx As
WsDynamicAdapterEx, ByVal storedCubeViewItem As CubeViewItem, _
            ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As
WsDynamicCubeViewEx Implements
IWsasDynamicDashboardsV800.GetDynamicCubeViewForDynamicAdapter
```

Implement the GetDynamicParametersForDynamicComponent method to modify the configuration or state of any Parameter referenced by components or by the dashboard itself. This method returns a collection of Parameter instances used on an Embedded Dynamic Dashboard type or an Embedded Dynamic Repeater Dashboard type.

C# Code

```
  public WsDynamicParameterCollection GetDynamicParametersForDynamicComponent(SessionInfo
si, IWsasDynamicDashboardsApiV800 api, DashboardWorkspace workspace,
            DashboardMaintUnit maintUnit, WsDynamicComponentEx dynamicComponentEx,
Dictionary<string, string> customSubstVarsAlreadyResolved)
```

VB.Net Code

```
  Public Function GetDynamicParametersForDynamicComponent(ByVal si As SessionInfo, ByVal api
As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal dynamicComponentEx As
WsDynamicComponentEx, ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String))
_
            As WsDynamicParameterCollection Implements
IWsasDynamicDashboardsV800.GetDynamicParametersForDynamicComponent
```

Implement the GetDynamicCubeViewForDynamicComponent method to modify the WsDynamicCubeViewEx that is returned as part of the Dynamic Dashboard creation.

C# Code

```
   WsDynamicCubeViewEx GetDynamicCubeViewForDynamicComponent(SessionInfo si,
IWsasDynamicDashboardsApiV800 api, DashboardWorkspace workspace, DashboardMaintUnit
maintUnit,
            WsDynamicComponentEx dynamicComponentEx, CubeViewItem storedCubeViewItem,
Dictionary<string, string> customSubstVarsAlreadyResolved)
```

VB.Net Code

```
Public Function GetDynamicCubeViewForDynamicComponent(ByVal si As SessionInfo, ByVal api As
IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal dynamicComponentEx As
WsDynamicComponentEx, ByVal storedCubeViewItem As CubeViewItem, _
            ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As
WsDynamicCubeViewEx Implements
IWsasDynamicDashboardsV800.GetDynamicCubeViewForDynamicComponent
```

IWsasDynamicDashboardsApiV800

Dynamic Dashboard Services make an API instance of the IWsasDynamicDashboardsApiV800 interface available to their methods. This object has several methods and properties.

Property Name	Type	Notes
DbConnFW	DbConnInfo	Open connection to the Framework database. Use this to avoid opening a new connection.
DbConnAppOrFW	DbConnInfo	Open connection to the Application database or the Framework database, depending on which dashboard it is run in. Use this to avoid opening a new connection.
IsSystemLevel	Boolean	Whether this code is being run as part of a System dashboard.
BRGlobals	BRGlobals	Can be used to cache data between runs.

These are the main types of methods provided by this API object:

GetDynamic*/GetEmbedded*
This is used to load in one or more dynamic instances, typically to modify them.

SaveDynamic*State
This is used to save the state of a particular element.

GetStored*
This is used to load in the configuration of specific elements, typically to modify them.

GetDynamic*/GetEmbedded* Functions

The call provided in the following example will load in memory any instance of the specified storedDashboard dashboard, typically so that some of its properties or components can be modified. convertStoredDashboardToDynamic will determine whether the resulting dashboard should be considered dynamic. This will impact available properties and further running of Services.

C# Code

```
WsDynamicDashboardEx GetEmbeddedDynamicDashboard(SessionInfo si, DashboardWorkspace
workspace, WsDynamicComponentEx parentDynamicComponentEx, Dashboard storedDashboard, string
nextLevelNameSuffixToAdd, Dictionary<string, string> nextLevelTemplateSubstVarsToAdd,
TriStateBool convertStoredDashboardToDynamic, WsDynamicItemStateType dynamicItemStateType);
```

The call provided in the following example will load in memory the collection of Component instances necessary to display the dashboard. convertStoredComponentsToDynamic will determine whether the resulting instances should be considered Dynamic. This will impact available properties and further running of Services.

C# Code

```
WsDynamicComponentCollection GetDynamicComponentsForDynamicDashboard(SessionInfo si,
DashboardWorkspace workspace, WsDynamicDashboardEx dynamicDashboardEx, string
nextLevelNameSuffixToAdd, Dictionary<string, string> nextLevelTemplateSubstVarsToAdd,
TriStateBool convertStoredComponentsToDynamic, WsDynamicItemStateType dynamicItemStateType);
```

The call provided in the following example will generate all repeated dashboards and components necessary to display a Dynamic Repeater dashboard. repeatArgsList is a list containing instances of type WsDynamicComponentRepeatArgs. This is a representation of key value pairs that should be used for each iteration. See Service-based Dynamic Repeater.

C# Code

```
WsDynamicComponentCollection GetDynamicComponentsRepeatedForDynamicDashboard( SessionInfo
si, DashboardWorkspace workspace, WsDynamicDashboardEx dynamicDashboardEx,
List<WsDynamicComponentRepeatArgs> repeatArgsList, TriStateBool
convertStoredComponentsToDynamic, WsDynamicItemStateType dynamicItemStateType);
```

The call provided in the following example will load in memory an instance of the specified storedComponent Component. This is typically so that some of its properties or components can be modified. convertStoredComponentToDynamic will determine whether the resulting instances should be considered Dynamic. This will impact available properties.

C# Code

```
WsDynamicComponentEx GetDynamicComponentForDynamicDashboard(SessionInfo si,
DashboardWorkspace workspace, WsDynamicDashboardEx dynamicDashboardEx, DashboardComponent
storedComponent, string nextLevelNameSuffixToAdd, Dictionary<string, string>
nextLevelTemplateSubstVarsToAdd, TriStateBool convertStoredComponentToDynamic,
WsDynamicItemStateType dynamicItemStateType);
```

The call provided in the following example will load in memory a collection of Data Adapters necessary to display the specified component instances.

C# Code

```
WsDynamicAdapterCollection GetDynamicAdaptersForDynamicComponent(SessionInfo si,
DashboardWorkspace workspace, WsDynamicComponentEx dynamicComponentEx, string
nextLevelNameSuffixToAdd, Dictionary<string, string> nextLevelTemplateSubstVarsToAdd,
TriStateBool convertStoredAdaptersToDynamic, WsDynamicItemStateType dynamicItemStateType);
```

The call provided in the following example will load in memory a single Data Adapter necessary to display a component. storedAdapter must be the source Adapter configuration.

C# Code

```
WsDynamicAdapterEx GetDynamicAdapterForDynamicComponent(SessionInfo si, DashboardWorkspace
workspace, WsDynamicComponentEx dynamicComponentEx, DashboardAdapter storedAdapter, string
nextLevelNameSuffixToAdd, Dictionary<string, string> nextLevelTemplateSubstVarsToAdd,
```

```
TriStateBool convertStoredAdapterToDynamic, WsDynamicItemStateType dynamicItemStateType);
```

The call provided in the following example will load in memory a single Cube View, which will be used to provide records for a Data Adapter. dynamicAdapterEx is the source Adapter configuration. storedCubeViewItem is the source Cube View configuration.

C# Code

```
WsDynamicCubeViewEx GetDynamicCubeViewForDynamicAdapter(SessionInfo si, DashboardWorkspace
workspace, WsDynamicAdapterEx dynamicAdapterEx, CubeViewItem storedCubeViewItem, string
nextLevelNameSuffixToAdd, Dictionary<string, string> nextLevelTemplateSubstVarsToAdd,
TriStateBool convertStoredCubeViewToDynamic, WsDynamicItemStateType dynamicItemStateType);
```

The call provided in the following example will load in memory a collection of Parameter instances necessary to display the provided component.

C# Code

```
WsDynamicParameterCollection GetDynamicParametersForDynamicComponent(SessionInfo si,
DashboardWorkspace workspace, WsDynamicComponentEx dynamicComponentEx, string
nextLevelNameSuffixToAdd, Dictionary<string, string> nextLevelTemplateSubstVarsToAdd,
TriStateBool convertStoredParametersToDynamic, WsDynamicItemStateType dynamicItemStateType);
```

GetStored* Functions

- List<DashboardDbrdCompMemberEx> GetStoredComponentsForDynamicDashboard (SessionInfo si, DashboardWorkspace workspace, WsDynamicDashboard dynamicDashboard);
- DashboardDbrdCompMemberEx GetStoredComponentForDynamicDashboard (SessionInfo si, DashboardWorkspace workspace, WsDynamicDashboard dynamicDashboard, string componentName);

- DashboardDbrdCompMemberEx GetStoredComponentForDynamicDashboard (SessionInfo si, DashboardWorkspace workspace, WsDynamicDashboard dynamicDashboard, Guid componentID);

Save*State Functions

- void SaveDynamicDashboardState(SessionInfo si, WsDynamicComponent parentComponent, WsDynamicDashboardEx dynamicDashboardEx, WsDynamicItemStateType dynamicItemStateType);
- void SaveDynamicComponentState(SessionInfo si, WsDynamicDashboard parentDashboard, WsDynamicComponentEx dynamicComponentEx, WsDynamicItemStateType dynamicItemStateType);
- void SaveDynamicAdapterState(SessionInfo si, WsDynamicComponent parentComponent, WsDynamicAdapterEx dynamicAdapterEx, WsDynamicItemStateType dynamicItemStateType);

Dynamic Dashboards

Dynamic Dashboards give developers the ability to build and modify dashboards through code. You use Workspace Assemblies, which enhance the flexibility of your dashboards and reduce the time spent in the user interface. This enables you to build more of your solutions within Workspaces.

> **NOTE:** Dynamic Dashboards should be created by developers.

Implementing Dynamic Dashboard Services

Dynamic Dashboard Services offer unique capabilities because unlike other service types, they do not have an equivalent in traditional Business Rules. Dynamic Dashboard Services provide additional connections in the loading mechanism of a dashboard, enabling developers to customize the dashboard and its components.

The most commonly implemented methods in a Dynamic Dashboard Service are GetEmbeddedDynamicDashboard and GetDynamicComponentsForDynamicDashboard.

Dynamic Dashboard Service Methods

GetEmbeddedDynamicDashboard

This method is run when you display an Embedded Dynamic Dashboard type or an Embedded Dynamic Repeater Dashboard type. The Dynamic Dashboard Service is returned by the Service Factory set in the Maintenance Unit or Workspace containing the dashboard. In the body of this function, developers can access or modify their properties and instantiate in memory all objects necessary to display the dashboard.

In the following example, users can load a dashboard that uses a certain border color by default and change it according to a user-specific property:

```
Public Function GetEmbeddedDynamicDashboard(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
    ByVal maintUnit As DashboardMaintUnit, ByVal parentDynamicComponentEx As WsDynamicComponentEx, ByVal storedDashboard As Dashboard, _
    ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As WsDynamicDashboardEx Implements IWsasDynamicDashboardsV800.GetEmbeddedDynamicDashboard
    Try
        If (api IsNot Nothing) Then
            ' check on dashboard name to execute only when necessary
            If storedDashboard.Name.XFEqualsIgnoreCase("My Embedded Dashboard") Then
                ' create the dashboard in memory
                Dim myDash As WsDynamicDashboardEx = api.GetEmbeddedDynamicDashboard(si, _
                workspace, parentDynamicComponentEx, _
                storedDashboard, String.Empty,Nothing, _
                TriStateBool.Unknown, WsDynamicItemStateType.Unknown)
                ' modify formatting if necessary
                If si.UserName.XFEqualsIgnoreCase("Richmond Avenal") Then
                    myDash.DynamicDashboard.DisplayFormat &= ", BorderColor = Black"
                End If
            End If
        End If

        Return Nothing
    Catch ex As Exception
        Throw New XFException(si, ex)
    End Try
End Function
```

VB.Net Code

```
Public Function GetEmbeddedDynamicDashboard(ByVal si As SessionInfo, ByVal api As
IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal parentDynamicComponentEx As
WsDynamicComponentEx, ByVal storedDashboard As Dashboard, _
            ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As
WsDynamicDashboardEx Implements IWsasDynamicDashboardsV800.GetEmbeddedDynamicDashboard
            Try
                If (api IsNot Nothing) Then
                       ' check on dashboard name to execute only when necessary
                    If storedDashboard.Name.XFEqualsIgnoreCase("My Embedded Dashboard") Then
                        ' create the dashboard in memory
                        Dim myDash As WsDynamicDashboardEx = api.GetEmbeddedDynamicDashboard
(si, _
                        workspace, parentDynamicComponentEx, _
                        storedDashboard, String.Empty,Nothing, _
                        TriStateBool.Unknown, WsDynamicItemStateType.Unknown)
                        ' modify formatting if necessary
                        If si.UserName.XFEqualsIgnoreCase("Richmond Avenal") Then
                            myDash.DynamicDashboard.DisplayFormat &= ", BorderColor = Black"
                        End If
                    End If
                End If

                Return Nothing
            Catch ex As Exception
                Throw New XFException(si, ex)
            End Try
        End Function
```

Users can also create new Repeater items to generate a large number of new components on the page without having to set them up manually. See the use case in Service-based Dynamic Repeater.

```
Public Function GetEmbeddedDynamicDashboard(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
    ByVal maintUnit As DashboardMaintUnit, ByVal parentDynamicComponentEx As WsDynamicComponentEx, ByVal storedDashboard As Dashboard, _
    ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As WsDynamicDashboardEx Implements IWsasDynamicDashboardsV800.GetEmbeddedDynamicDashboard
    Try
        If (api IsNot Nothing) Then
            ' check on dashboard name to execute only when necessary
            If storedDashboard.Name.XFEqualsIgnoreCase("My Embedded Dashboard") Then

                ' get a list of users, and wrap them in repeatArgs
                Dim users As List(Of UserSummaryInfo) = BRApi.Security.Admin.GetUsers(si)
                Dim repeatArgs As New List(Of WsDynamicComponentRepeatArgs)
                For Each user As UserSummaryInfo In users:
                    Dim nextLevelTemplateSubstVarsToAdd As New Dictionary(Of String, String) _
                        From {{"name",user.Name},{"description",user.Description}}
                                repeatArgs.Add( _
                                    New WsDynamicComponentRepeatArgs( _
                                        user.UniqueID, nextLevelTemplateSubstVarsToAdd))
                Next

                ' create the dashboard in memory
                Dim myDash As WsDynamicDashboardEx = api.GetEmbeddedDynamicDashboard(si, _
                    workspace, parentDynamicComponentEx, _
                    storedDashboard, String.Empty,Nothing, _
                    TriStateBool.Unknown, WsDynamicItemStateType.Unknown)

                ' Attach our List of repeaters.
                ' This is equivalent to populating property "Component Template Repeat Items"
                '    on the Dashboard
                    myDash.DynamicDashboard.Tag = repeatArgs
                Return myDash

            Else
                ' default behaviour for other dashboards
                Return api.GetEmbeddedDynamicDashboard(si, _
                    workspace, parentDynamicComponentEx, _
                    storedDashboard, String.Empty,Nothing, _
                    TriStateBool.Unknown, WsDynamicItemStateType.Unknown)
            End If
```

VB.Net Code

```
 Public Function GetEmbeddedDynamicDashboard(ByVal si As SessionInfo, ByVal api As
IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal parentDynamicComponentEx As
WsDynamicComponentEx, ByVal storedDashboard As Dashboard, _
            ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As
WsDynamicDashboardEx Implements IWsasDynamicDashboardsV800.GetEmbeddedDynamicDashboard
            Try
                If (api IsNot Nothing) Then
                    ' check on dashboard name to execute only when necessary
                    If storedDashboard.Name.XFEqualsIgnoreCase("My Embedded Dashboard") Then

                        ' get a list of users, and wrap them in repeatArgs
                        Dim users As List(Of UserSummaryInfo) =
BRApi.Security.Admin.GetUsers(si)
                        Dim repeatArgs As New List(Of WsDynamicComponentRepeatArgs)
                        For Each user As UserSummaryInfo In users:
                            Dim nextLevelTemplateSubstVarsToAdd As New Dictionary(Of String,
String) _
                                From {{"name",user.Name},{"description",user.Description}}
                                        repeatArgs.Add( _
                                            New WsDynamicComponentRepeatArgs( _
                                                user.UniqueID,
nextLevelTemplateSubstVarsToAdd))
                        Next

                        ' create the dashboard in memory
```

```
                        Dim myDash As WsDynamicDashboardEx = api.GetEmbeddedDynamicDashboard
(si, _
                            workspace, parentDynamicComponentEx, _
                            storedDashboard, String.Empty,Nothing, _
                            TriStateBool.Unknown, WsDynamicItemStateType.Unknown)

                        ' Attach our List of repeaters.
                        ' This is equivalent to populating property "Component Template
Repeat Items"
                        '    on the Dashboard
                            myDash.DynamicDashboard.Tag = repeatArgs
                        Return myDash

                    Else
                        ' default behaviour for other dashboards
                        Return api.GetEmbeddedDynamicDashboard(si, _
                            workspace, parentDynamicComponentEx, _
                            storedDashboard, String.Empty,Nothing, _
                            TriStateBool.Unknown, WsDynamicItemStateType.Unknown)
                    End If

                End If

                Return Nothing
            Catch ex As Exception
                Throw New XFException(si, ex)
            End Try
        End Function
```

GetDynamicComponentsForDynamicDashboards

This method is run when a Dynamic Dashboard generates the necessary components in memory. Users can specify custom logic to modify the generated components rather than modifying the dashboard as a whole. GetEmbeddedDynamicDashboard and GetDynamicComponentsForDynamicDashboard are often coupled, particularly when using a Repeater configuration. GetEmbeddedDynamicDashboard sets up values for each iteration of the Repeater Template, and GetDynamicComponentsForDynamicDashboard uses these values to configure the repeated components.

For example, users may want to dynamically change an Embedded Component property to display a different Embedded Dashboard. Instead of having to use Parameters that reference Legacy Business Rules within the Maintenance Unit, users can use code in a Dynamic Dashboard Service. See the following example:

```
Public Function GetDynamicComponentsForDynamicDashboard(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
    ByVal maintUnit As DashboardMaintUnit, ByVal dynamicDashboardEx As WsDynamicDashboardEx, ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) _
    As WsDynamicComponentCollection Implements IWsasDynamicDashboardsV800.GetDynamicComponentsForDynamicDashboard
    Try
        If (api IsNot Nothing) Then
            ' if we are executing the specific dashboard...
            If dynamicDashboardEx.DynamicDashboard.Name.XFEqualsIgnoreCase("MyEmbeddedDashboard") Then

                ' first we create the components ...
                Dim comps As WsDynamicComponentCollection = _
                api.GetDynamicComponentsForDynamicDashboard(si, workspace, _
                dynamicDashboardEx, String.Empty, Nothing, TriStateBool.Unknown, _
                WsDynamicItemStateType.Unknown)

                ' ... then find the Embedded Component that we will manipulate …
                Dim embComp As WsDynamicDbrdCompMemberEx = _
                comps.GetComponentUsingBasedOnName("My Embedded Component")

                ' ... then make the change …
                embComp.DynamicComponentEx.DynamicComponent.Component.EmbeddedDashboardName = _
                dashboardName
                ' ... and return all components!
                Return comps
```

VB.Net Code

```
Public Function GetDynamicComponentsForDynamicDashboard(ByVal si As SessionInfo, ByVal api
As IWsasDynamicDashboardsApiV800, ByVal workspace As DashboardWorkspace, _
            ByVal maintUnit As DashboardMaintUnit, ByVal dynamicDashboardEx As
WsDynamicDashboardEx, ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String))
_
            As WsDynamicComponentCollection Implements
IWsasDynamicDashboardsV800.GetDynamicComponentsForDynamicDashboard
            Try
                If (api IsNot Nothing) Then
                    ' if we are executing the specific dashboard...
                    If dynamicDashboardEx.DynamicDashboard.Name.XFEqualsIgnoreCase
("MyEmbeddedDashboard") Then

                        ' first we create the components ...
                        Dim comps As WsDynamicComponentCollection = _
                        api.GetDynamicComponentsForDynamicDashboard(si, workspace, _
                        dynamicDashboardEx, String.Empty, Nothing, TriStateBool.Unknown, _
                        WsDynamicItemStateType.Unknown)

                        ' ... then find the Embedded Component that we will manipulate …
                        Dim embComp As WsDynamicDbrdCompMemberEx = _
                        comps.GetComponentUsingBasedOnName("My Embedded Component")

                        ' ... then make the change …
                        embComp.DynamicComponentEx.DynamicComponent.Component.EmbeddedDashbo
ardName = _
                        dashboardName
                        ' ... and return all components!
                        Return comps
                    End If
                End If

                Return Nothing
            Catch ex As Exception
                Throw New XFException(si, ex)
            End Try
        End Function
```

These are the other methods implemented in a Dynamic Dashboard Service:

- GetDynamicAdaptersForDynamicComponent
- GetDynamicCubeViewForDynamicAdapter
- GetDynamicParametersForDynamicComponent
- GetDynamicCubeViewForDynamicComponent

These methods are similar to GetDynamicComponentsForDynamicDashboard. They are run when the relevant elements, such as Data Adapters and parameters, are created in memory. To implement these methods, instantiate the elements in memory with the relevant API method, modify them as necessary, and return them. For more information on these methods and their arguments, see API Reference.

Stored Components and Dynamic Components

A Stored Component is any item you create on the Workspaces page, such as a button component. Its configuration is saved in the database and used whenever the component is displayed in a dashboard. When a Stored Component is used in an Embedded Dynamic Dashboard or an Embedded Dynamic Repeater Dashboard, one or more Dynamic Component instances are created in memory. You can modify these instances in the Dynamic Dashboard Service to provide a customized experience without modifying the underlying Stored Component.

Stored Components are represented by traditional classes, such as DashboardComponent and DashboardParameter. Dynamic Components are represented by classes with a Ws prefix, like WsDynamicComponent and WsDynamicParameter.

Embedded Dynamic Repeater Dashboard

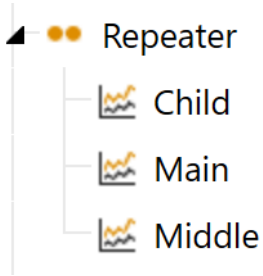
The Embedded Dynamic Repeater dashboard is a type of dashboard used to add multiple instances of the same component without having to recreate each component individually. Unlike custom controls, you are not required to make additional embedded components. An example is a dashboard that requires the following:

- Multiple labels that have differing text
- Many buttons that perform different actions
- Multiple reports and charts with slight variations in data

> **NOTE:** Embedded Dynamic Repeater dashboards are not visible outside of the Workspaces page.

Recommended Setup

To maintain organization, OneStream recommends you create your first dashboards using the Embedded Dynamic Repeater dashboard type, similar to the following image.



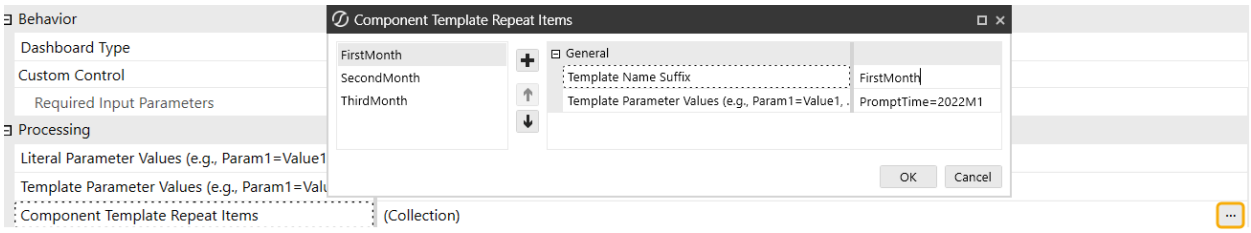
The Repeater Dashboard group contains three dashboards:

- **Child**: This dashboard contains the components needed to create a dashboard.
- **Main**: This dashboard contains the Embedded Component of the Middle dashboard. Make sure to set the Dashboard Type property to **Top Level**, **Use Default**, or **Top Level No Parameters**. This ensures that the dashboard is visible outside the Workspaces page.
- **Middle**: This dashboard has its Dashboard Type property set to Embedded Dynamic Repeater. You will also configure other properties as well. This is where you assign the Embedded Component for the Child dashboard.

Component Template Repeat Items

Whichever dashboard you designate with the Embedded Dynamic Repeater dashboard type it must have a few additional properties configured. Setting the dashboard type to Embedded Dynamic Repeater will activate the Component Template Repeat Items property.

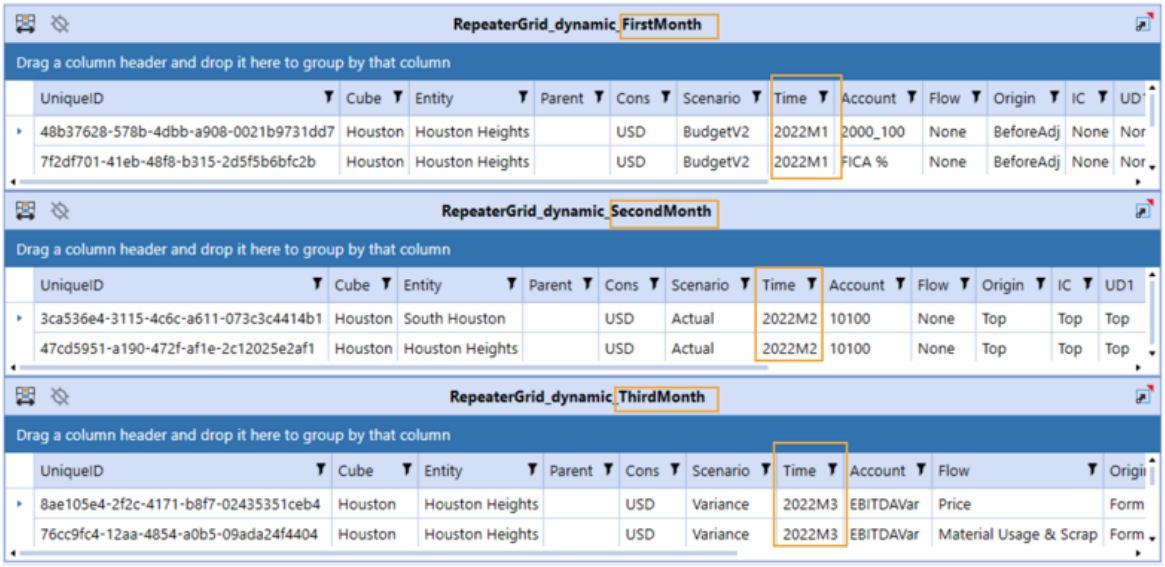
Click the **ellipses** at the right end of the Component Template Repeat Items property to repeat the component added to this dashboard and to display a dialog box where you can configure additional properties.



In the example above, three versions of the same component have been repeated: FirstMonth, SecondMonth, and ThirdMonth. You must set the following two General settings for each:

- **Template Name Suffix**: Added to the name of the dashboard when run.
- **Template Parameter Values**: Resolve any parameters in the component or data adapter.

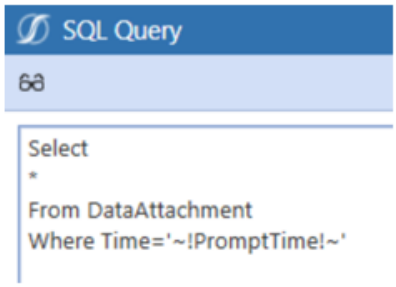
In the following image, the Time parameter is resolved differently in each repeated item.



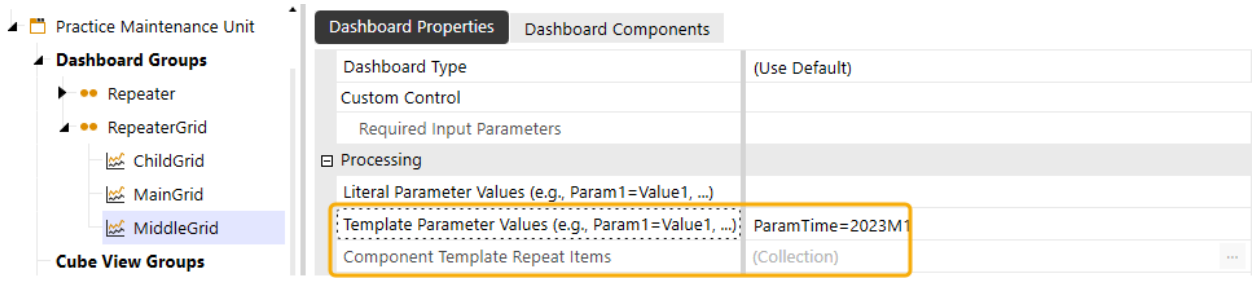
Template Parameters

Template parameters are slightly different than the typical parameters used within OneStream. Template parameters in Embedded Dynamic Repeater dashboards use the following syntax: ~!ParameterName!~.

The following is an example of a template parameter used in a SQL Data Adapter:



This parameter type does not prompt you for information and can only be resolved through the Template Parameters property.

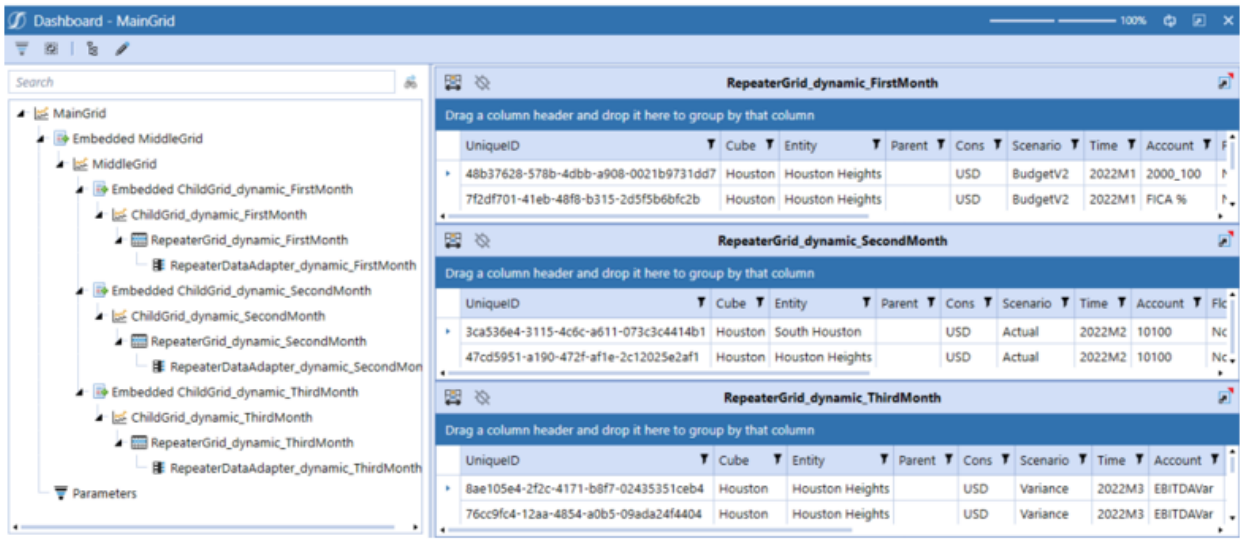


The Component Template Repeat Items property is where you could repeat the component and vary the parameter setting each time. However, the first highlighted property in the above image is where you would resolve the parameter without repeating it.

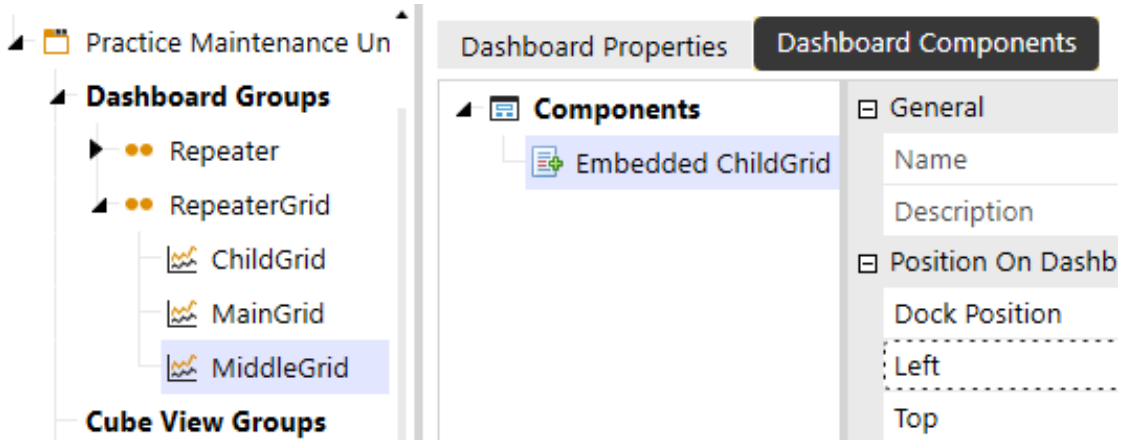
> **NOTE:** You cannot use this parameter type in cube views.

Embedded Dynamic Repeater in Design Mode

When you are using the Embedded Dynamic Repeater dashboard in design mode, your changes to the dashboard are viewable so that you can see variations. For example, in the following image, it looks as though three separate dashboards, grid components, and data adapters have been added to the Middle Grid dashboard.



But if you look at the **Workspace**> **Dashboard**> **Dashboard Component** tab , only one embedded component was added.

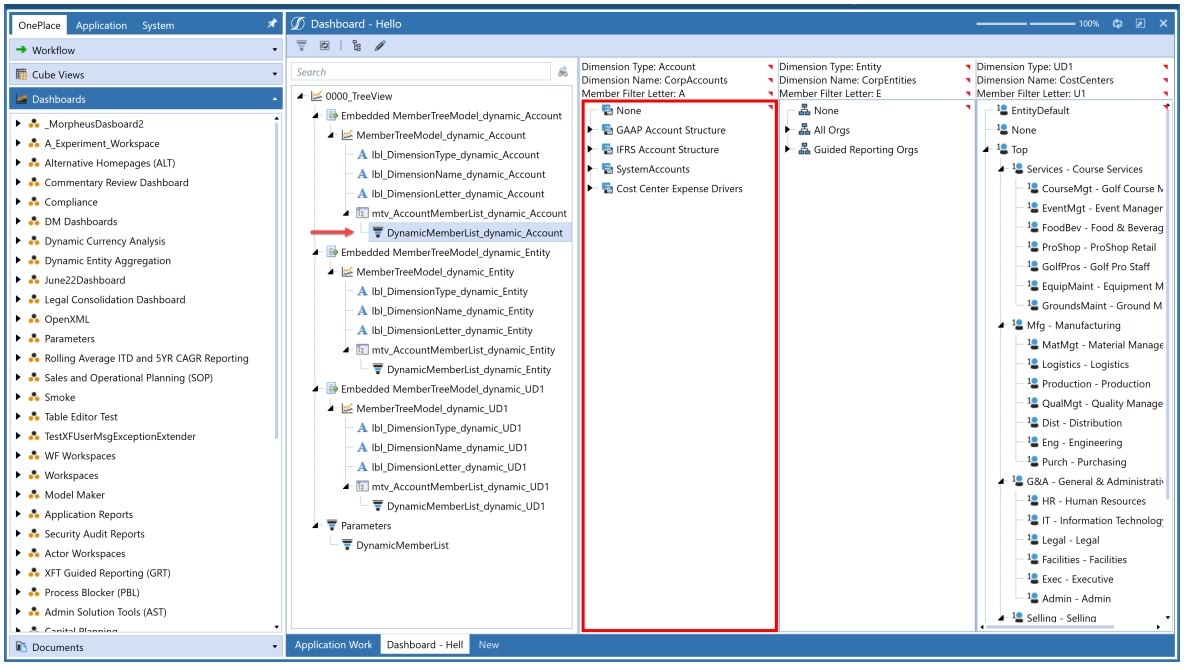


This means that the view in design mode shows different variations configured through the Embedded Dynamic Repeater dashboard components.

Dynamic Bound Parameter

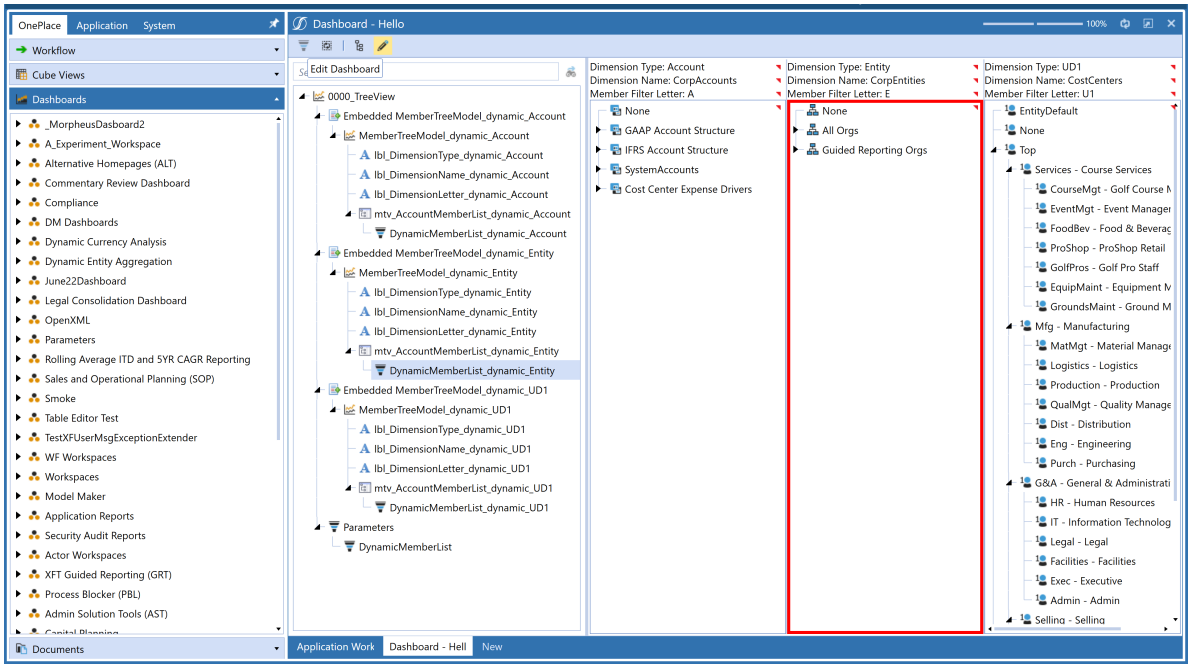
When using an Embedded Dynamic Repeater Dashboard type, any components that use a bound parameter can be dynamically created and repeated to support Template Substitution variables. The syntax is (~|SubstitutionVariable|~). Components that are set up to use bound parameters ensure that template parameters are substituted prior to the execution of a bound parameter.

While in Design mode, you can view the dashboard that displays the Dynamic Bound Parameters in the Tree View. You can click on the Dynamic Bound Parameter to highlight its parent control.

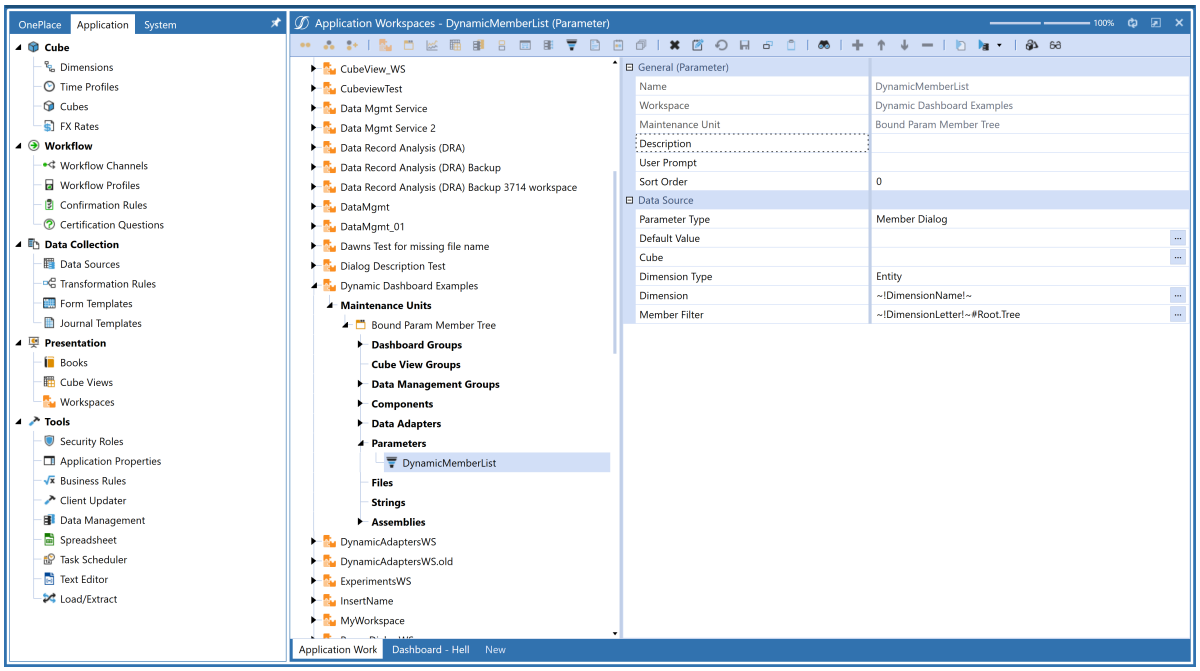


NOTE: A red border will highlight the parent control of the dynamic bound parameter.

Click the **Edit Dashboard** toolbar button in the Design Mode toolbar when a Dynamic Bound Parameter is selected to navigate to the Workspaces page.



The Parameter will be selected.



Service-based Dynamic Repeater

Dynamic Dashboards can leverage templates to display similar items in the same dashboard, such as a list of icons or multiple grids. You can also trigger this from a Dynamic Dashboard Service Assembly file, which enables you to dynamically generate the list of items. For information on the standard no-code Repeater functionality, see Embedded Dynamic Repeater Dashboard.

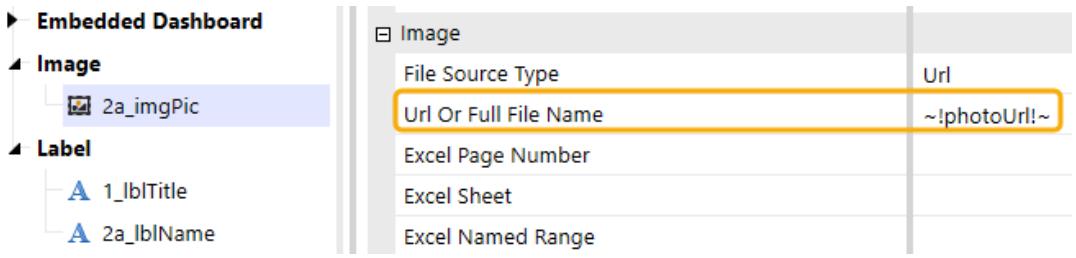
USE CASE

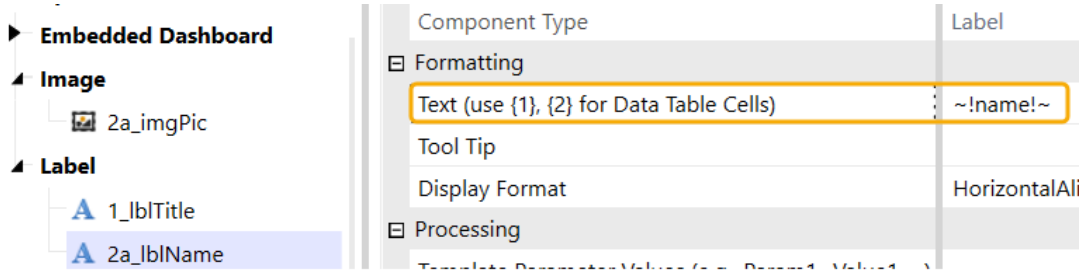
Use an Embedded Dynamic Repeater Dashboard to Display User Images and Names

Role	Benefits
Developer: Writes custom code and sets properties.	Users can set up an Embedded Dynamic Repeater Dashboard and add multiple instances of the same components without having to re-create each component individually.

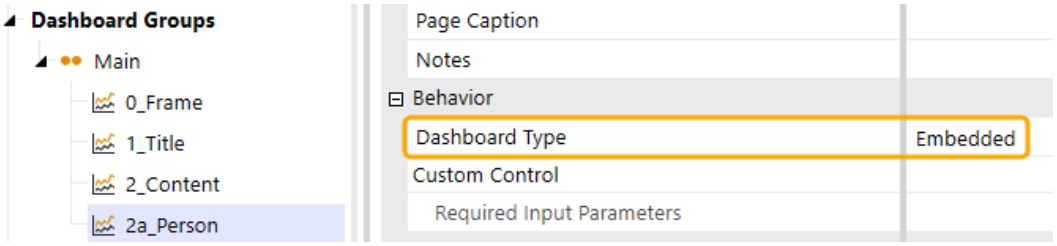
You want to display user images and names in a Dashboard. Complete the following steps:

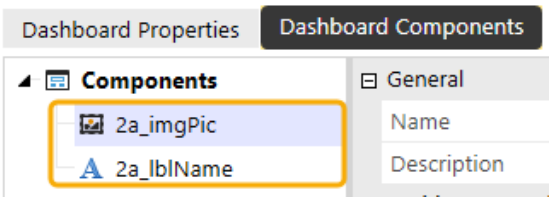
1. Create an image component and a label component. See Component Display Format Properties. In this use case, each component is configured with template parameter values variables. These are populated by the RepeatArgs in the code. When the template runs, it is replaced with the actual image URL or name.



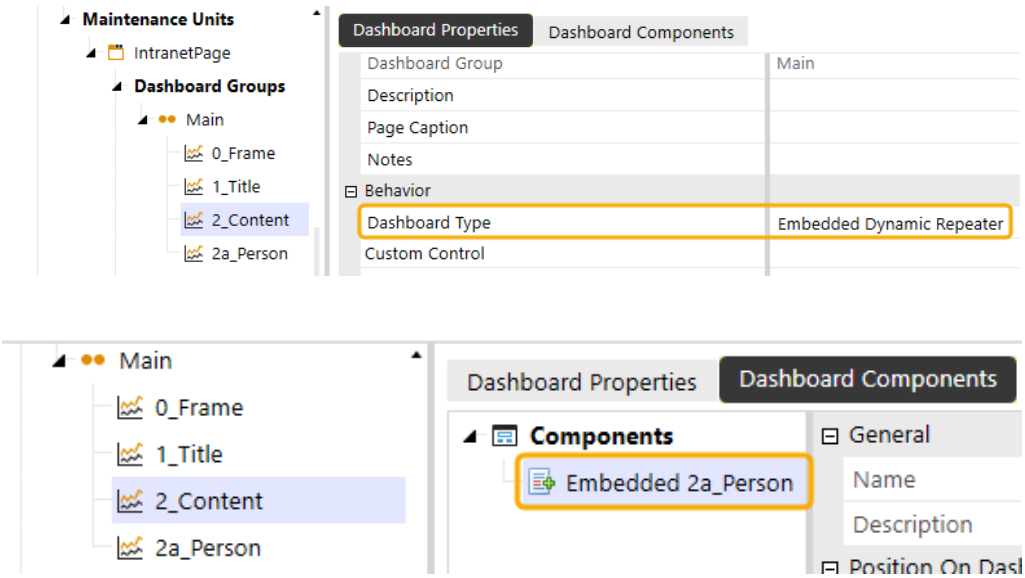


2. Create an Embedded Dashboard. See Dashboard Properties. In the Dashboard Components tab, add the image and label components to the Embedded Dashboard.

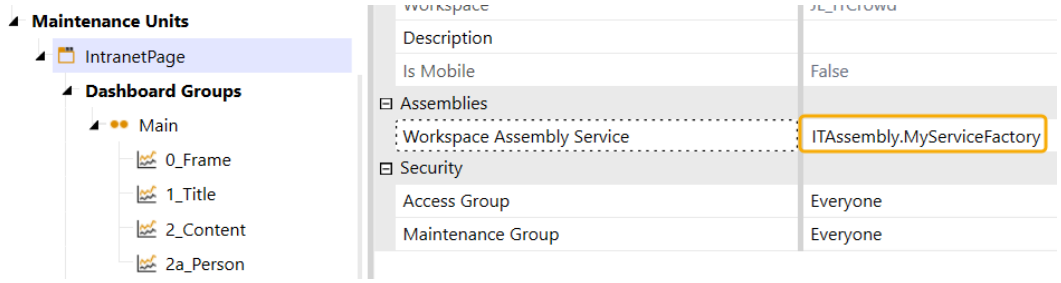


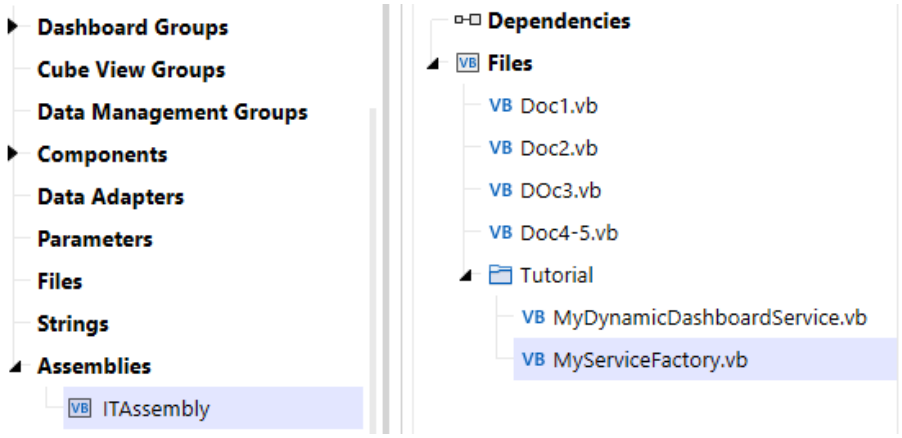


3. Create an Embedded Dynamic Repeater Dashboard. See Embedded Dynamic Repeater Dashboard. In the Dashboard Components tab, add the Embedded Dashboard Component. When the Embedded Dynamic Repeater Dashboard runs, it will repeat this Embedded Dashboard Component for each user. 0_Frame dashboard contains the Embedded Dashboard Components for 1_Title and 2_Content dashboards.



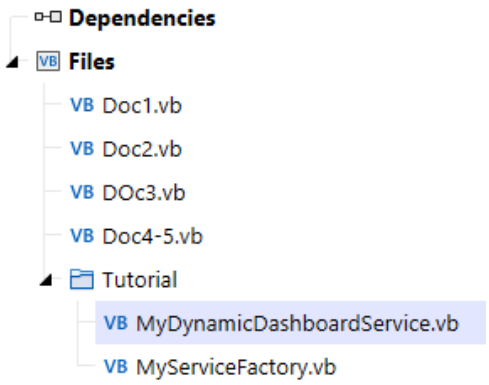
4. In this use case, the Maintenance Unit's Workspace Assembly Service property is set to ITAssembly.MyServiceFactory. This property matches the ITAssembly Assembly which contains the MyServiceFactory Assembly file. This assembly will return the services.





5. In the MyServiceFactory Assembly file, the following code is the request you need to enable the processing of Dynamic Dashboards. This code returns what is set in the MyDynamicDashboardService file in the same Assembly.

```
24          Public Function CreateWsAssemblyServiceInstance(ByVal si As SessionInfo,
25              Try
26                  Select Case wsasType
27
28                      Case Is = WsAssemblyServiceType.DynamicDashboards
29                          ' create and return an instance of our service
30                          Return New MyDynamicDashboardService()
31
```



6. The MyDynamicDashboardService Assembly file's default source code contains various automatically created methods. Fill in the sections required to display the user images and user names.

   The logic starts with GetEmbeddedDynamicDashboard. Use this to retrieve the list of users and set them in a data structure to keep track of the user images and user names.

```
23
24          Public Function GetEmbeddedDynamicDashboard(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, By
25              ByVal maintUnit As DashboardMaintUnit, ByVal parentDynamicComponentEx As WsDynamicComponentEx, ByVal storedDash
26              ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As WsDynamicDashboardEx Implements IWsa
27              Try
28                  If (api IsNot Nothing) Then
29                      ' only execute our logic on the Dynamic Repeater dashboard
30                      If storedDashboard.Name.XFEqualsIgnoreCase("2_Content") Then
31
32                          ' set up the list of objects that will contain our repeated variables
33                          Dim repeatArgs As New List(Of WsDynamicComponentRepeatArgs)
34
35                          ' retrieve user details and loop on each user, retrieving details
36                          ' Note: if the group doesn't exist, it will error out!
37                          Dim objGroupInfo As GroupInfo = BRApi.Security.Admin.GetGroup(si, "ITCrowd")
38                          For Each userPrincipal As Principal In objGroupInfo.ChildPrincipals
39                              Try
40                                  Dim userObj As User = brapi.Security.Admin.GetUser(si, userPrincipal.UniqueID).User
41
42                                  ' create a dictionary where the key is the name of the template variable to populate,
43                                  ' and the value is the actual string it will contain in this iteration
44                                  Dim nextLevelTemplateSubstVarsToAdd As New Dictionary(Of String, String) _
45                                      From {{"name", userObj.Name}, {"photoUrl", userObj.Text1}}
```

7. Build a dictionary with a list of these RepeatArgs.

```
Public Function GetEmbeddedDynamicDashboard(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, ByVal workspace A
    ByVal maintUnit As DashboardMaintUnit, ByVal parentDynamicComponentEx As WsDynamicComponentEx, ByVal storedDashboard As Dashbo
    ByVal customSubstVarsAlreadyResolved As Dictionary(Of String, String)) As WsDynamicDashboardEx Implements IWsasDynamicDashboar
    Try
        If (api IsNot Nothing) Then
            ' only execute our logic on the Dynamic Repeater dashboard
            If storedDashboard.Name.XFEqualsIgnoreCase("2_Content") Then

                ' set up the list of objects that will contain our repeated variables
                Dim repeatArgs As New List(Of WsDynamicComponentRepeatArgs)

                ' retrieve user details and loop on each user, retrieving details
                ' Note: if the group doesn't exist, it will error out!
                Dim objGroupInfo As GroupInfo = BRApi.Security.Admin.GetGroup(si, "ITCrowd")
                For Each userPrincipal As Principal In objGroupInfo.ChildPrincipals
                    Try
                        Dim userObj As User = brapi.Security.Admin.GetUser(si, userPrincipal.UniqueID).User

                        ' create a dictionary where the key is the name of the template variable to populate,
                        ' and the value is the actual string it will contain in this iteration
                        Dim nextLevelTemplateSubstVarsToAdd As New Dictionary(Of String, String) _
                            From {{"name", userObj.Name}, {"photoUrl", userObj.Text1}}

                        ' wrap the dictionary in a WsDynamicComponentRepeatArgs
                        Dim repeatInstance As New WsDynamicComponentRepeatArgs( _
                                userObj.UniqueID.ToString, _        ' suffix unique to this repetition (in this case a User ID)
                                nextLevelTemplateSubstVarsToAdd)    ' dictionary containing populated variables

                        ' add our instance to the list of WsDynamicComponentRepeatArgs objects
                        repeatArgs.Add( repeatInstance )
                    Catch e As Exception
                        brapi.ErrorLog.LogMessage(si, "Error with " & userPrincipal.UniqueID.ToString)
                    End Try
                Next
```

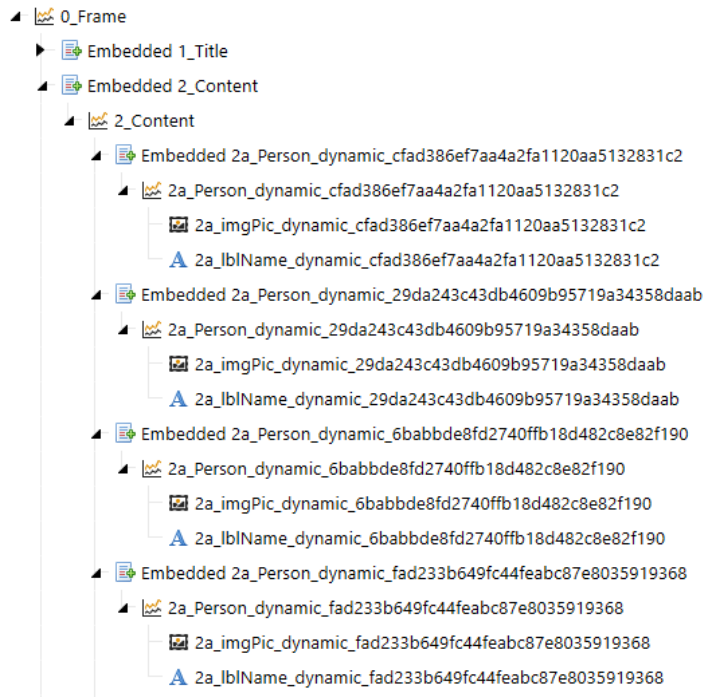
8. Create the dashboard in memory and add that RepeatArgs, the list of objects containing the user images and user names, to this Tag object.

```
                ' create the dashboard in memory
                Dim myDash As WsDynamicDashboardEx = api.GetEmbeddedDynamicDashboard(si, _
                                                workspace, parentDynamicComponentEx, _
                                                storedDashboard, String.Empty,Nothing, _
                                                TriStateBool.Unknown, WsDynamicItemStateType.Unknown)

                ' Attach our List of repeaters to the dashboard.
                ' This is equivalent to populating property "Component Template Repeat Items"
                '    on the Dashboard.
                myDash.DynamicDashboard.Tag = repeatArgs

                ' return the dashboard. This will eventually trigger GetDynamicComponentsForDynamicDashboard below.
                Return myDash
            Else
                Return api.GetEmbeddedDynamicDashboard(si, workspace, parentDynamicComponentEx, storedDashboard, Str
            End If

        End If

        Return Nothing
    Catch ex As Exception
        Throw New XFException(si, ex)
    End Try
End Function
```

9. After the dashboard has built the list and ran the method, it calls the GetDynamicComponentsForDynamicDashboard method, which generates each individual object.

```
Public Function GetDynamicComponentsForDynamicDashboard(ByVal si As SessionInfo, ByVal api As IWsasDynamicDashboardsApiV800, ByVal workspac
    ByVal maintUnit As DashboardMaintUnit, ByVal dynamicDashboardEx As WsDynamicDashboardEx, ByVal customSubstVarsAlreadyResolved As Dictio
    As WsDynamicComponentCollection Implements IWsasDynamicDashboardsV800.GetDynamicComponentsForDynamicDashboard
    Try
        If (api IsNot Nothing) Then
            ' only on the dynamic repeater
            If dynamicDashboardEx.DynamicDashboard.Name.XFEqualsIgnoreCase("2_Content") Then

                ' retrieve the list of items to repeat
                Dim repeatArgsList As List(Of WsDynamicComponentRepeatArgs) = dynamicDashboardEx.DynamicDashboard.Tag
                ' create and return the components, passing the list as argument
                Return api.GetDynamicComponentsRepeatedForDynamicDashboard(si, workspace, dynamicDashboardEx, _
                    repeatArgsList, _           ' our list of repeated items
                    TriStateBool.TrueValue, _   ' set this to False if you DON'T want to generate individual Parameters for each iteration
                    WsDynamicItemStateType.MinimalWithTemplateParameters)   ' what to sync across servers
```

This is the result. When this dashboard is run, it will display the corresponding user images and user names.



Dynamic Cube Views

Dynamic Cube Views give developers the ability to build and modify Cube Views through code. Developers can modify existing Cube Views at runtime to configure what is presented. Any property a developer can set for a Cube View in the user interface can also be set in the Assembly file when configuring a Dynamic Cube View.

> **NOTE:** Dynamic Cube Views should be created by developers.

Complete the following steps to set a Cube View as dynamic:

1. Navigate to the Workspace and Maintenance Unit.
2. From **Cube View Groups**, select the Cube View.
3. Navigate to **Advanced** > **Cube View Properties** > **Dynamic** and set the **Is Dynamic** field to **True**. If you are using the **Designer** tab, go to **General Settings** > **Common** > **Dynamic**.

   > **NOTE:** If the Is Dynamic property is set to False, the Cube View will display what is set in the Designer and Advanced tabs' user interfaces and ignore anything set in the assembly file.

Complete the following steps to set up a new assembly for a Dynamic Cube View:

1. Create an assembly under the same Maintenance Unit that contains the Dynamic Cube View.
2. Create a file under the new assembly. See Create an Assembly.
   a. In the **Add File** dialog box, enter a file name. From the **Source Code Type** drop-down menu, select **Service Factory**. See Configure the Service Factory.
3. Create another assembly file. This will be used in the Service Factory.

a. In the **Add File** dialog box, enter a file name.

b. From the **Source Code Type** drop-down menu, select **Dynamic Cube View Service**.

4. Open the Service Factory file you created in step 2.

   a. Uncomment the case statement for Dynamic Cube Views.

   b. Replace the returned class with the name of the file created in step 3.

```
24          public IWsAssemblyServiceBase CreateWsAssemblyServiceInstance(
25              DashboardWorkspace workspace, WsAssemblyServiceType wsAsse
26          {
27              try
28              {
29                  switch (wsAssemblyServiceType)
30                  {
31                      //case WsAssemblyServiceType.Component:
32                      //    return new WsasComponent();
33
34                      //case WsAssemblyServiceType.Dashboard:
35                      //    return new WsasDashboard();
36
37                      //case WsAssemblyServiceType.DataManagementStep:
38                      //    return new WsasDataManagementStep();
39
40                      //case WsAssemblyServiceType.DataSet:
41                      //    return new WsasDataSet();
42
43                      //case WsAssemblyServiceType.DynamicDashboards:
44                      //    return new WsasDynamicDashboards();
45
46                      case WsAssemblyServiceType.DynamicCubeView:
47                          return new WsasDynamicCubeView();
48
49                      //case WsAssemblyServiceType.FinanceCore:
50                      //    return new WsasFinanceCore();
```

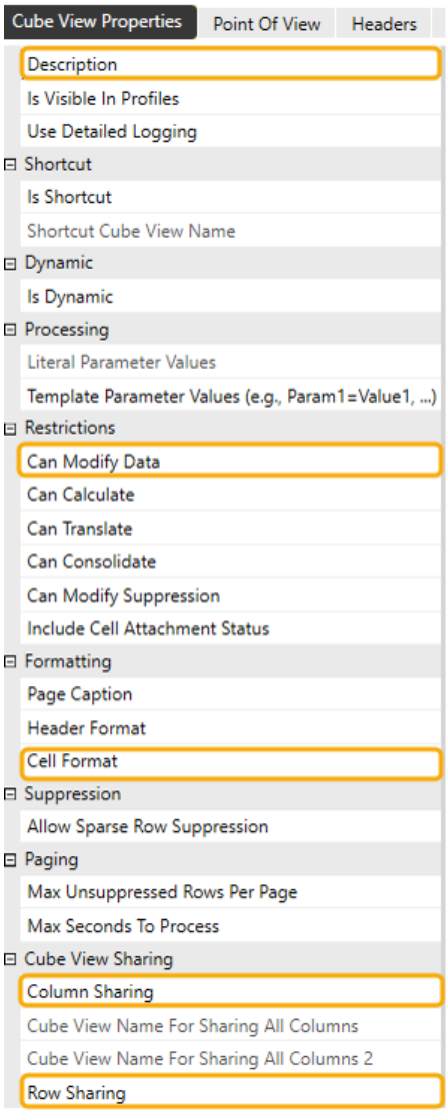
5. Select the Workspace. Under **Assemblies**, update the **Workspace Assembly Service** property to {WsAssemblyName}.{ServiceFactoryName}.
6. The Service Factory will return an instance of the WsasDynamicCubeView class you created in step 3 for any Cube View that has its Is Dynamic property set to True. To target a specific Cube View in the Dynamic Cube View Service, enter the CubeViewItem's Name property.

```
20 namespace Workspace.__WsNamespacePrefix.__WsAssemblyName
21 {
22     public class WsasDynamicCubeView : IWsasDynamicCubeViewV800
23     {
24         public CubeViewItem GetDynamicCubeViewItem(SessionInfo si, IWsasDynamicCubeViewApiV800 api,
25         {
26             if (cubeViewItem.Name.XFEqualsIgnoreCase("MyDynamicCubeView"))
27             {
28                 CubeView cubeView = cubeViewItem.GetCubeView(si);
```

Implementing Dynamic Cube View Services

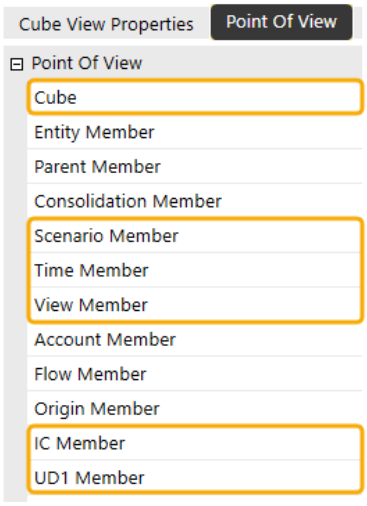
Any property that you can set in the Cube View Advanced and Designer tabs can be set in code through your Dynamic Cube View Service.

This example displays sample configurations of common Cube View Properties.



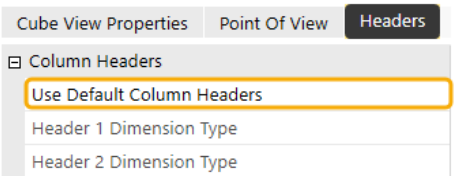

```
// General
cubeViewItem.Description = "Dynamic Description";
cubeViewItem.IsVisibleInProfiles = true;
cubeView.UseDetailedLogging = false;

// Shortcut
cubeViewItem.IsShortcut = false;
cubeViewItem.ShortcutCubeViewName = string.Empty;
//cubeViewItem.LiteralParameterValues = string.Empty;

// Processing
//cubeViewItem.TemplateParameterValues = string.Empty;

// Restrictions
cubeView.CanModifyData = true;
cubeView.CanCalculate = true;
cubeView.CanTranslate = true;
cubeView.CanConsolidate = true;
cubeView.CanModifySuppression = false;
cubeView.IncludeCellAttachmentStatus = false;

// Formatting
cubeView.PageCaption = string.Empty;
cubeView.HeaderFormat = "RowExpansionMode = ExpandAll";
cubeView.CellFormat = "ShowCurrency = True";

// Suppression
cubeView.AllowSparseRowSuppression = false;

// Paging
cubeView.MaxNumUnsuppressedRowsPerCubeViewPage = -1;
cubeView.MaxNumSecondsToProcessCubeView = -1;

// Cube View Sharing
cubeView.ColSharingType = CubeViewRowColSharingType.NotUsed;
cubeView.CubeViewNameForSharingAllCols = string.Empty;
cubeView.CubeViewNameForSharingAllCols2 = string.Empty;
cubeView.RowSharingType = CubeViewRowColSharingType.NotUsed;
cubeView.CubeViewNameForSharingAllRows = string.Empty;
cubeView.CubeViewNameForSharingAllRows2 = string.Empty;
```

This example displays the Point Of View properties configured in code, which includes a Literal Parameter containing the string Actual set to the Scenario Member.



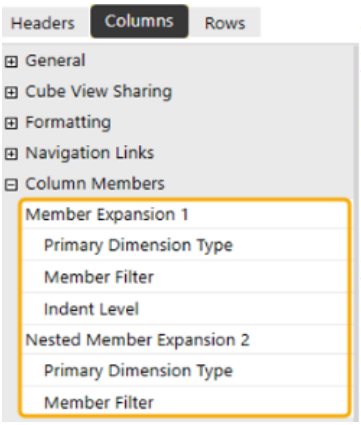

```
// Set Cube View POV.
cubeView.CubeViewCube.Name = "Level1";
cubeView.CubeViewPovMembers.Scenario.Name = "|!Actual!|";
cubeView.CubeViewPovMembers.Time.Name = "2022M1";
cubeView.CubeViewPovMembers.View.Name = "YTD";
cubeView.CubeViewPovMembers.Flow.Name = "None";
cubeView.CubeViewPovMembers.IC.Name = "Top";
cubeView.CubeViewPovMembers.UD1.Name = "CC100";
cubeView.CubeViewPovMembers.UD2.Name = "None";
cubeView.CubeViewPovMembers.UD3.Name = "None";
cubeView.CubeViewPovMembers.UD4.Name = "None";
cubeView.CubeViewPovMembers.UD5.Name = "None";
cubeView.CubeViewPovMembers.UD6.Name = "None";
cubeView.CubeViewPovMembers.UD7.Name = "None";
cubeView.CubeViewPovMembers.UD8.Name = "None";
```

This example displays configuration to the Column Header properties.



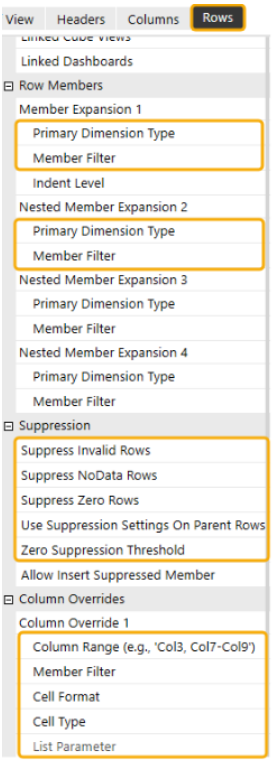

```
// Column Headers.
cubeView.UseDefaultColumnHeaders = true;
//cubeView.ColumnHeaders.Add(new CubeViewHeader("Entity"));
//cubeView.ColumnHeaders.Add(new CubeViewHeader("Consolidation"));
```

This example displays configuration to the Column Members properties.




```
// Column Members
// Member Expansion 1.
var columnMemberExpansionOne = new CubeViewMemberExpansion();
columnMemberExpansionOne.PrimaryDimTypeName = "Consolidation";
columnMemberExpansionOne.MemberFilter = "C#Local";
columnMemberExpansionOne.InitialIndentLevel = 0;

cubeViewColumn.MemberExpansions.Add(columnMemberExpansionOne);

// Nested Member Expansion 2.
var columnMemberExpansionTwo = new CubeViewMemberExpansion();
columnMemberExpansionTwo.PrimaryDimTypeName = "Origin";
columnMemberExpansionTwo.MemberFilter = "O#Top, O#BeforeElim, O#Elimination";
columnMemberExpansionTwo.InitialIndentLevel = 0;

cubeViewColumn.MemberExpansions.Add(columnMemberExpansionTwo);
```

This example displays the Row Members, Suppression, and Column Overrides properties configured in code.




```
// Row Members.
// Member Expansion 1.
var rowMemberExpansionOne = new CubeViewMemberExpansion();
rowMemberExpansionOne.PrimaryDimTypeName = "Entity";
rowMemberExpansionOne.MemberFilter = "E#TotalCompany_L1.TreeDescendantsInclusive";
rowMemberExpansionOne.InitialIndentLevel = 0;

cubeViewRow.MemberExpansions.Add(rowMemberExpansionOne);

// Nested Member Expansion 2.
var rowMemberExpansionTwo = new CubeViewMemberExpansion();
rowMemberExpansionTwo.PrimaryDimTypeName = "Account";
rowMemberExpansionTwo.MemberFilter = "A#400IC, A#500IC,A#5000IC_L4 , A#CommonICPlug";
rowMemberExpansionTwo.InitialIndentLevel = 0;

cubeViewRow.MemberExpansions.Add(rowMemberExpansionTwo);

// Suppression
cubeViewRow.SuppressInvalidRows = false;
cubeViewRow.SuppressNoDataRows = false;
cubeViewRow.SuppressZeroRows = false;
cubeViewRow.UseSuppressionSettingsOnParentRows = true;
cubeViewRow.ZeroSuppressionThreshold = string.Empty;

// Column Overrides
var cubeViewColumnOverrideOne = new CubeViewOverride();
cubeViewColumnOverrideOne.RowsOrColsToOverride = new StringRange();
cubeViewColumnOverrideOne.MemberFilter = string.Empty;
cubeViewColumnOverrideOne.CellFormat = string.Empty;
cubeViewColumnOverrideOne.CellType = CubeViewCellType.Unknown;
cubeViewColumnOverrideOne.ListParameter = string.Empty;
```

Use the cubeViewItem.GetCubeView and cubeViewItem.SetCubeView methods to retrieve and set Cube View data for your Dynamic Cube View. Enter these methods to use an existing Cube View and add the defined properties for configuration.

```
25          {
26              if (cubeViewItem.Name.XFEqualsIgnoreCase("MyDynamicCubeView"))
27              {
28                  CubeView cubeView = cubeViewItem.GetCubeView(si);
29
30                  Cube View Properties
107
108                 Point of View
127
128                 Headers
189
190                 Columns
256
257                 Rows
324
325                 cubeViewItem.SetCubeView(si, cubeView);
326             }
327
328             return cubeViewItem;
```

Build Data Management in Workspaces

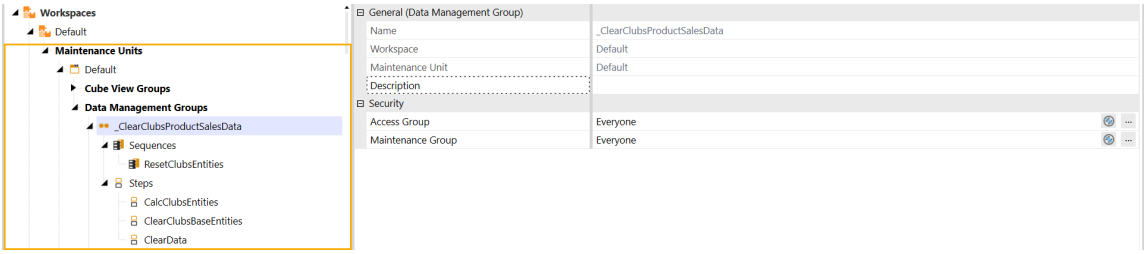
Data Management objects, such as groups, sequences, and steps are used to run common tasks and processes through a series of steps and sequences. You can create data management objects on the Data Management page under the Application tab and also build and use them on the Workspaces page.

> **NOTE:** Data Management Profiles will not be displayed in the user interface.

Display Data Management Groups

There is a Default folder under Maintenance Units in the Default Workspace and Default Maintenance Units section. This is where the legacy data management objects from the Data Management groups page reside.

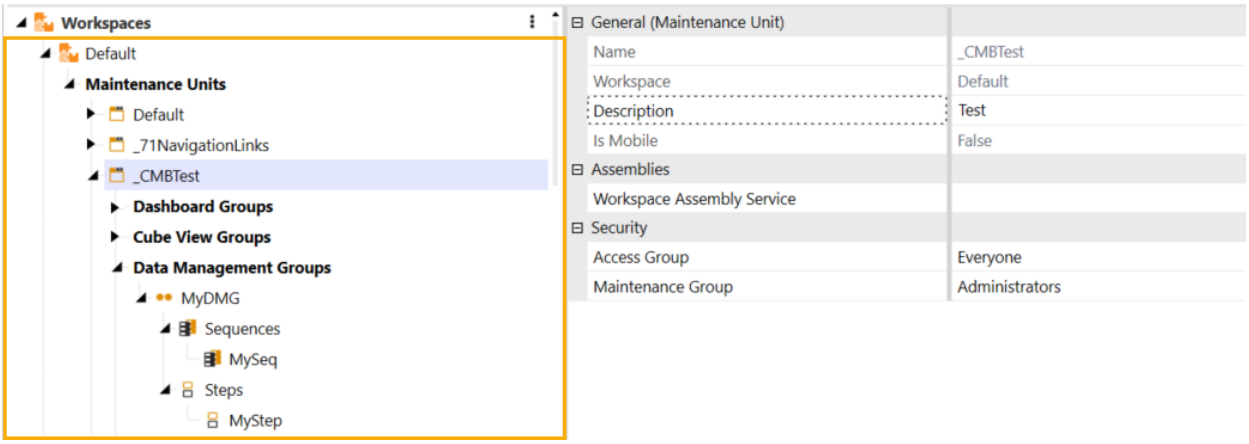
1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. In the **Application Workspaces** pane, under **Workspaces**, expand **Default** > **Maintenance Units** > **Default** > **Data Management Groups**.



Display Data Management Groups (Non-Default Maintenance Unit)

Data management groups also display under a Default Workspace, under the Non-Default Maintenance Unit.

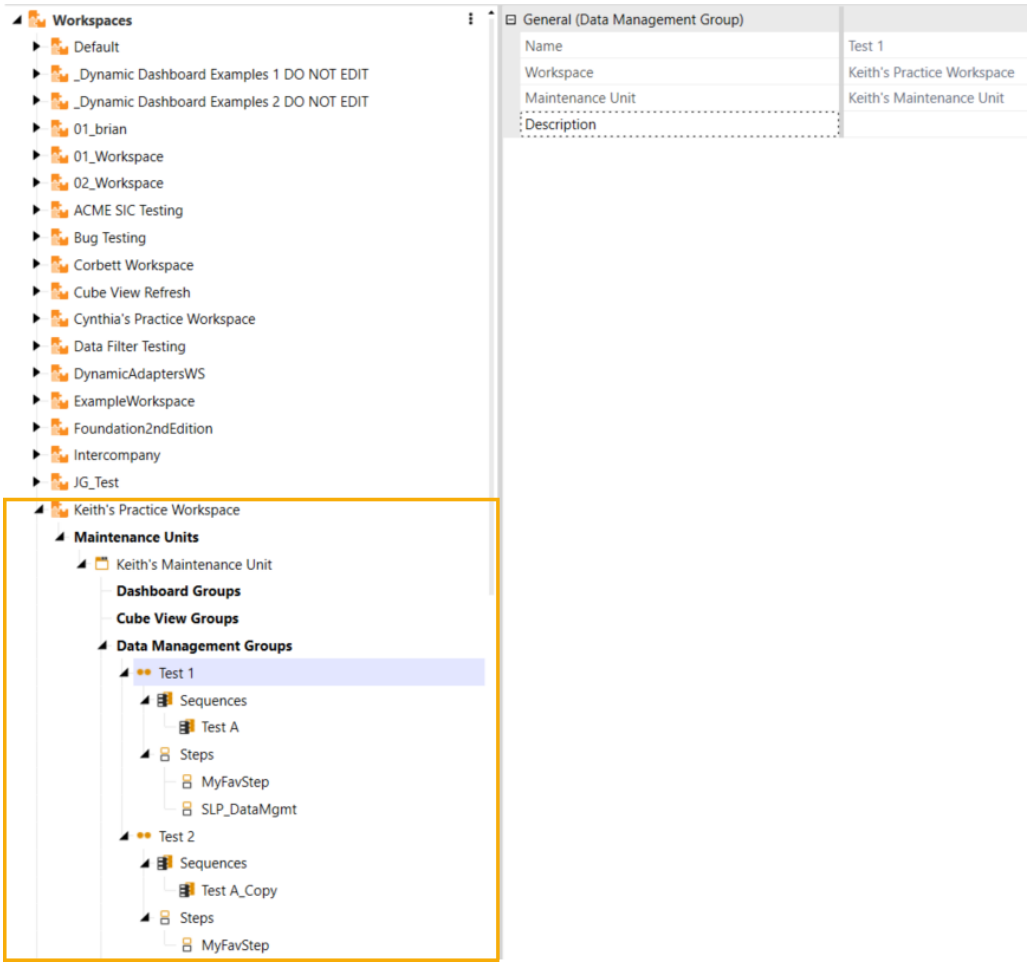
1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. In the **Application Workspaces** pane, under **Workspaces**, expand **Maintenance Units**, locate the maintenance unit, and expand **Data Management Groups**.



Display Data Management Groups (Non-Default Workspace)

Data management groups also display under a Non-Default Workspace.

1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. In the **Application Workspaces** pane, under **Workspaces**, locate the workspace and expand it. Then expand **Maintenance Units**, locate the maintenance unit, and expand **Data Management Groups**.



Display Data Management Groups Profiles

Data management group profiles will not be displayed in the user interface under the Data Management page under the Application tab or on the Workspaces page.

> **NOTE:** The following Data Management Profile tables still exist if you are using Data Management Profiles: DataMgmtProfile, DataMgmtProfileMember, AuditDataMgmtGroup and AuditDataMgmtProfileMember.

Build Data Management Sequences and Steps in Workspaces

Data Management objects, such as sequences and steps, can be copied, pasted, and deleted in the workspace to maintain existing applications and create new solutions.

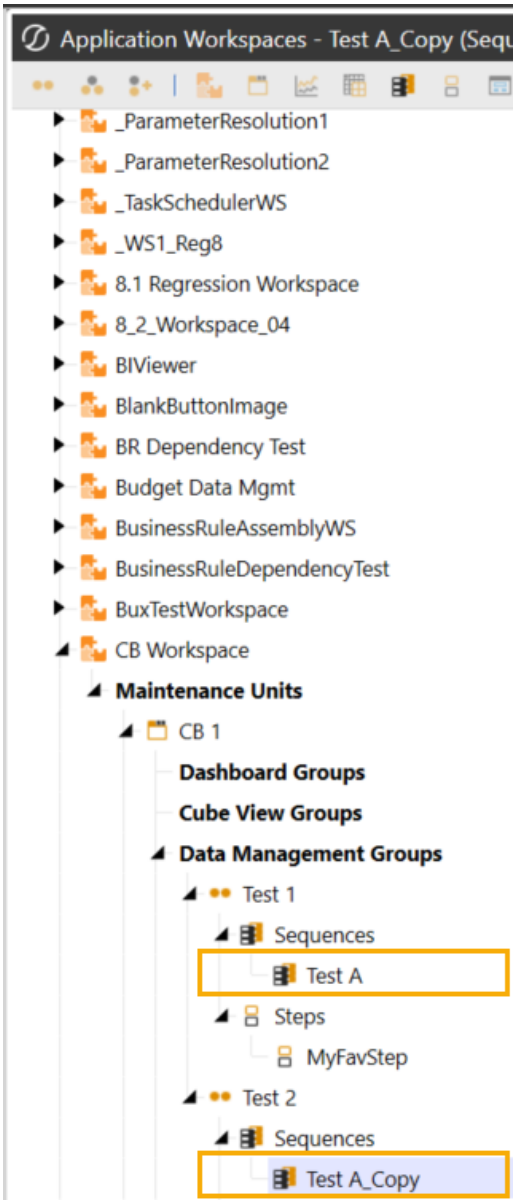
Copy and Paste Sequences and Steps in Workspaces

When copying a data management sequence, it will copy and paste the associated data management steps. When selecting a data management step to copy and paste, it will only copy the specific data management step not the associated data management sequence.

You can copy and paste data management steps and sequences to a different data management group in the same workspace and maintenance unit.

1. In the **Application Workspaces** pane, under **Workspaces**, locate the workspace and expand it. Then, expand **Maintenance Units**, select and expand the maintenance unit, and expand a specific **Data Management Groups**.
2. Select the data management sequence and then either right-click and select copy or click to copy your sequence to another data management group within the same maintenance unit.
3. Select a data management group in the same maintenance group to paste your copied sequence to and either right-click and select paste or click to paste.

If there are associated data management steps, they will also be copied and pasted to the data management group. The copied sequence name will append "_Copy" to the end of the name in the same maintenance unit and workspace.

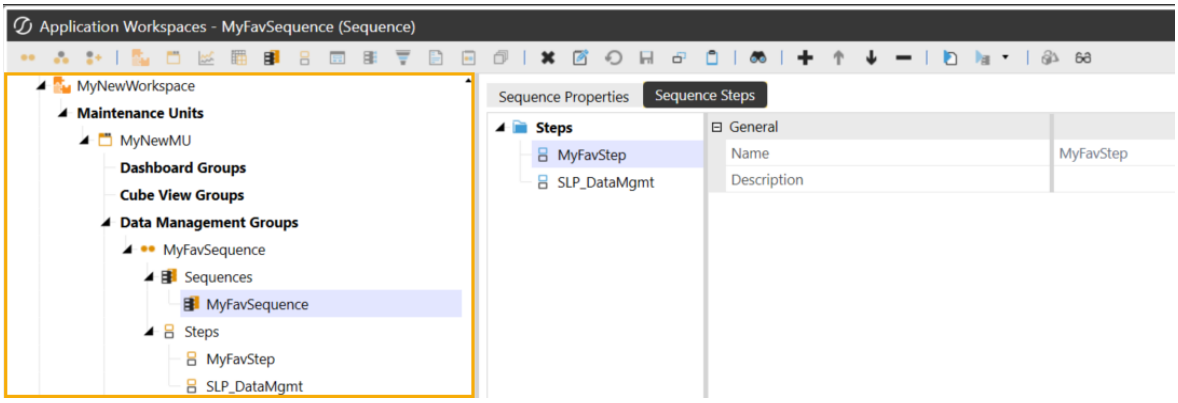


> **NOTE:** Sequence names must be unique in a workspace within the same maintenance group or in a workspace with a different maintenance unit or data management group, which will append "_Copy" to a copied sequence.

You can also copy and paste data management sequences and steps to another workspace.

1. In the **Application Workspaces** pane, under **Workspaces**, locate the workspace and expand it. Then, expand **Maintenance Units**, select and expand the maintenance unit, and expand **Data Management Groups**.
2. Select the data management sequence and then either right-click and select copy or click to copy your sequence to another workspace data management group.
3. Select the other workspace to paste your copied sequence to and then either right-click and select paste or click to paste.

The copied sequence name will display in the new Workspace data management group without "_ Copy" and as the copied sequence name.



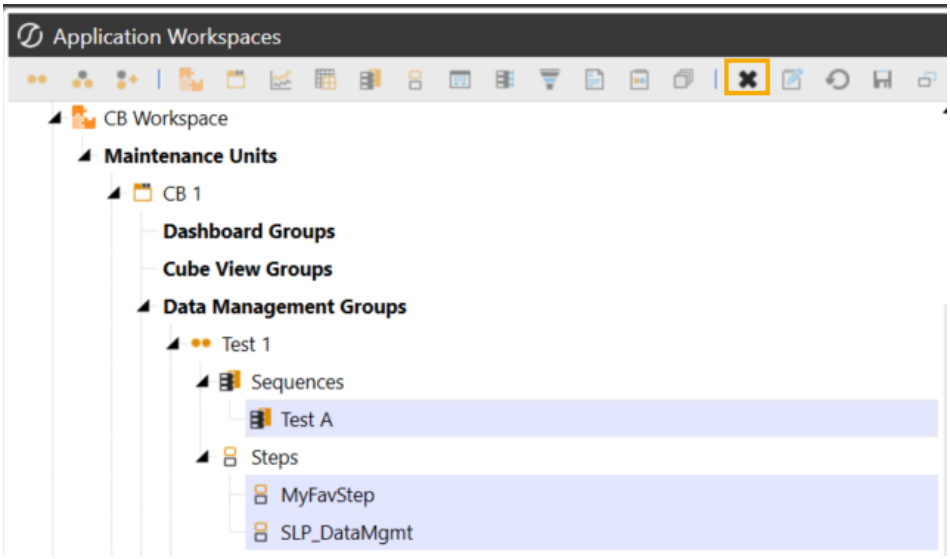
> **NOTE:** When copying and pasting steps and sequences to another workspace, the sequence name will appear as is since it is not located in the same maintenance unit and workspace.

If you multi-select and highlight a data management sequence and its associated data management steps, you will not be able to use the copy functionality. You can copy multiple sequences and steps at a time but cannot multi-select and copy both sequences and steps in the same instance.

Delete Sequences and Steps in Workspaces

You can multi-select more than one data management sequence and data management step to delete within the same data management group.

1. In the **Application Workspaces** pane, under **Workspaces**, locate the workspace and expand it. Then, expand **Maintenance Units**, select and expand the maintenance unit, and expand **Data Management Groups**.
2. Use the **CTRL** key to select multiple data management sequences and data management steps not grouped together.
3. Click to delete the selected data management sequences and steps.



NOTE: You cannot delete data management sequences or steps within different data management groups at the same time. Selecting data management items in different data management groups will disable the delete functionality.

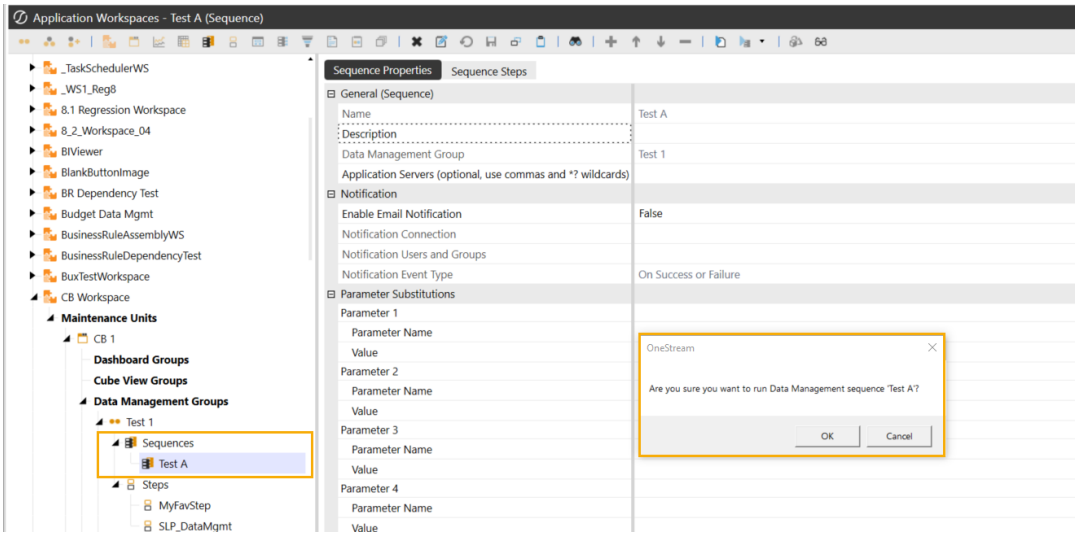
Run Sequences or Steps in Workspaces

In the Application Workspaces page, you can run or modify existing data management objects.

To run a sequence:

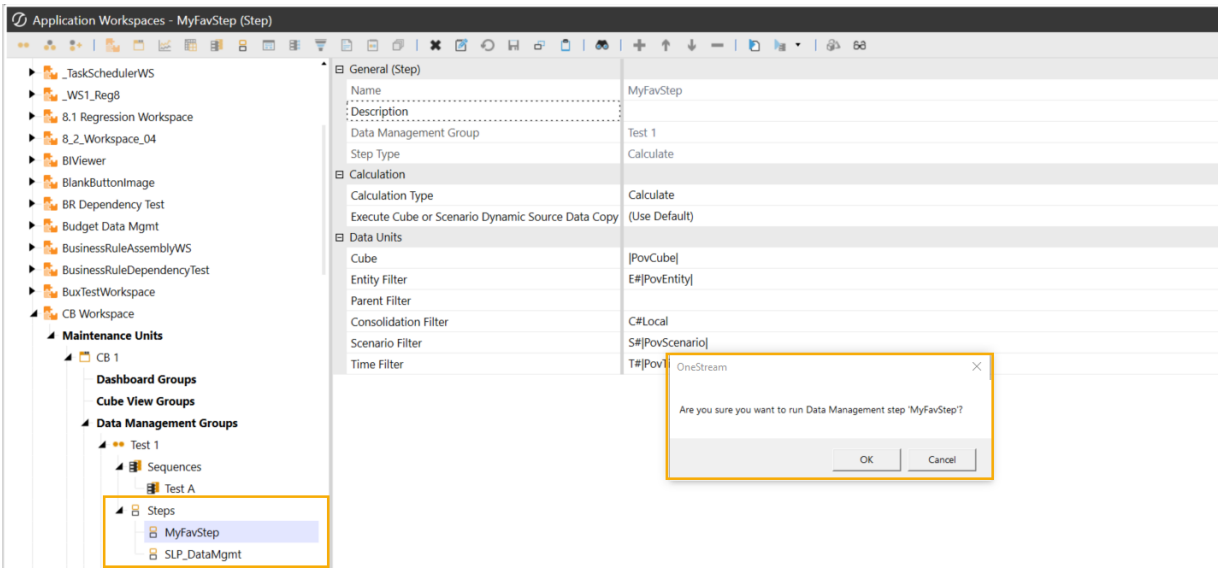
1. In the **Application Workspaces** pane, under **Workspaces**, locate the workspace and expand it. Then, expand **Maintenance Units**, select and expand the maintenance unit, and expand **Data Management Groups**.
2. Select the data management sequence in the data management group.
3. Select the **Run** button to run the data management sequence.
4. You will get a confirmation box asking if you are sure you want to run the data management sequence. Click the **OK** button.

> **NOTE:** Running a sequence without steps added to it will produce an error message.



To run a step:

1. In the **Application Workspaces** pane, under **Workspaces**, locate the workspace, and expand it. Then, expand **Maintenance Units**, select and expand the maintenance unit, and expand **Data Management Groups**.
2. Select the data management step in the data management group.
3. Select the **Run** button to run the data management step.
4. You will get a confirmation box asking if you are sure you want to run the data management step. Click the **OK** button.

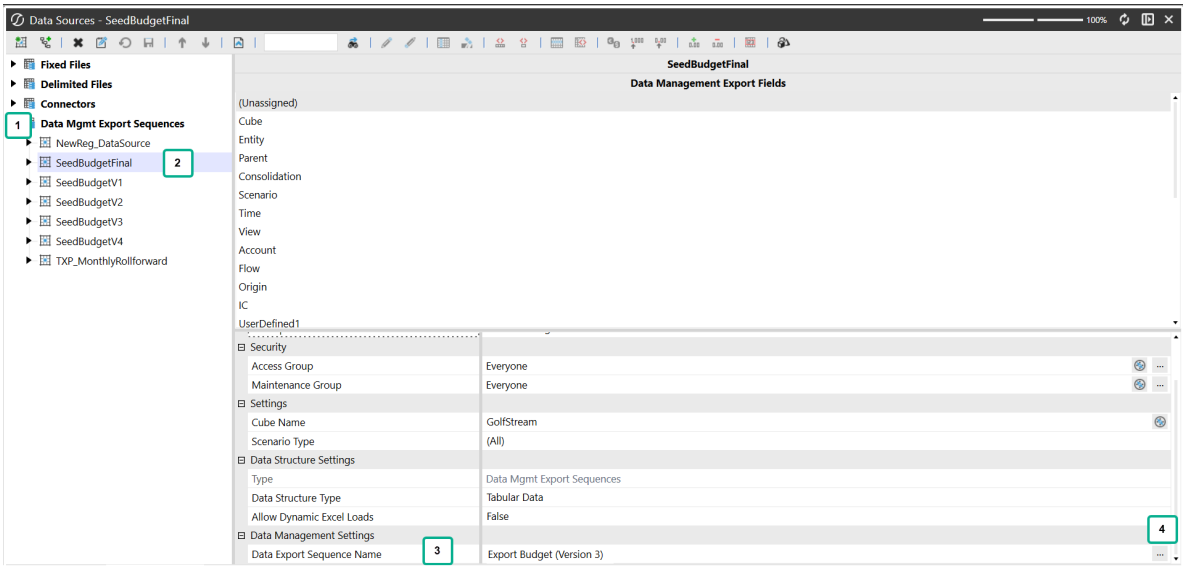


IMPORTANT: REST API Previous API 7.2.0 and 5.2.0 will accept an optional Workspace Name for Data Management Sequence. See OneStream Web API Endpoints.

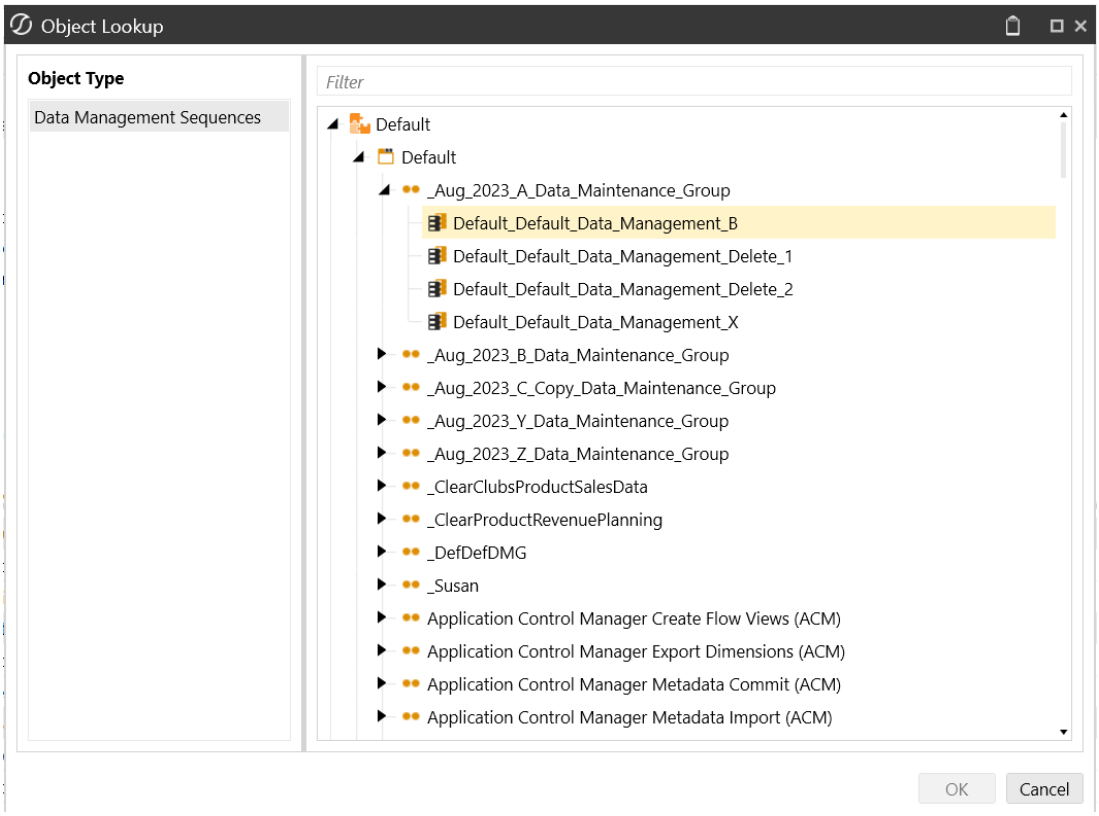
Data Management Export Sequences in Workspaces

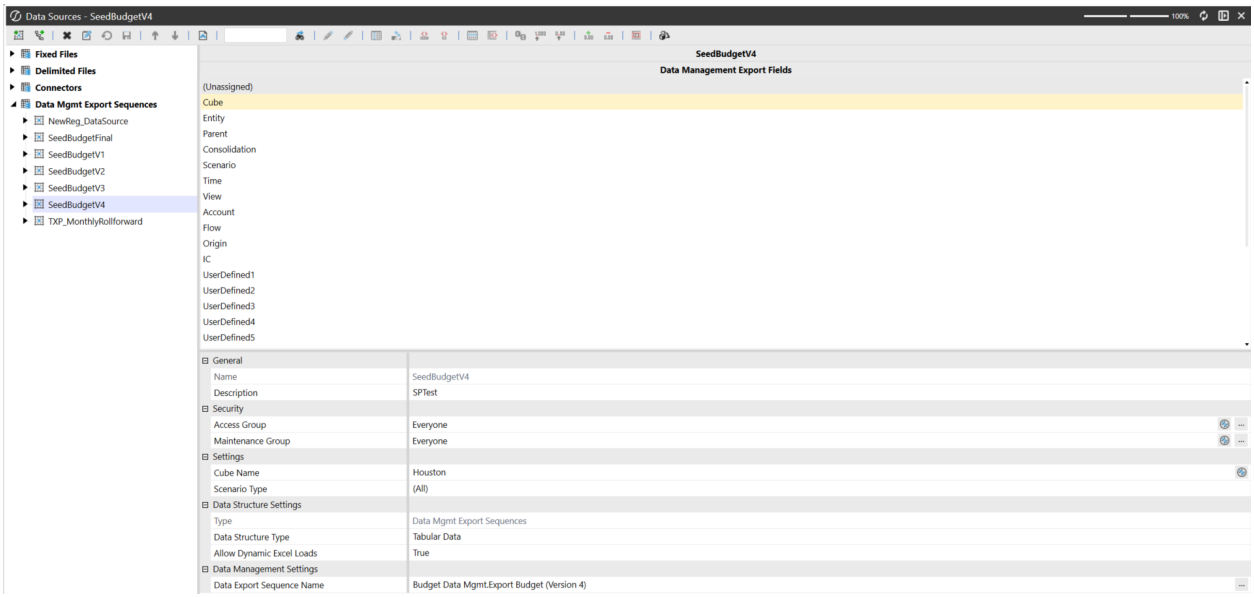
Display Data Management Export Sequences

1. On the **Application** tab, under **Data Collection**, click **Data Sources**.
2. Expand **Data Mgmt Export Sequences** and select a data management export sequence.
3. Under **Data Management Settings**, in the **Data Export Sequence Name** field, click the ellipsis (...).
4. On the Object Lookup dialog box, select the Data Management Sequence based on the workspace location.



> **NOTE:** You can select Data Management Sequences from the following areas: Default Workspace/Default Maintenance Unit, Default Workspace/Non-Default Maintenance Unit, and Non-Default Workspace for the Data Export Sequence Name.


Object Lookup
Object Type
Data Management Sequences
Filter
Default
Default
_Aug_2023_A_Data_Maintenance_Group
Default_Default_Data_Management_B
Default_Default_Data_Management_Delete_1
Default_Default_Data_Management_Delete_2
Default_Default_Data_Management_X
_Aug_2023_B_Data_Maintenance_Group
_Aug_2023_C_Copy_Data_Maintenance_Group
_Aug_2023_Y_Data_Maintenance_Group
_Aug_2023_Z_Data_Maintenance_Group
_ClearClubsProductSalesData
_ClearProductRevenuePlanning
_DefDefDMG
_Susan
Application Control Manager Create Flow Views (ACM)
Application Control Manager Export Dimensions (ACM)
Application Control Manager Metadata Commit (ACM)
Application Control Manager Metadata Import (ACM)
OK
Cancel


Data Sources - SeedBudgetV4
Fixed Files
Delimited Files
Connectors
Data Mgmt Export Sequences
NewReg_DataSource
SeedBudgetFinal
SeedBudgetV1
SeedBudgetV2
SeedBudgetV3
SeedBudgetV4
TXP_MonthlyRollforward
SeedBudgetV4
Data Management Export Fields
(Unassigned)
Cube
Entity
Parent
Consolidation
Scenario
Time
View
Account
Flow
Origin
IC
UserDefined1
UserDefined2
UserDefined3
UserDefined4
UserDefined5
General
Name
SeedBudgetV4
Description
SPTest
Security
Access Group
Everyone
Maintenance Group
Everyone
Settings
Cube Name
Houston
Scenario Type
(All)
Data Structure Settings
Type
Data Mgmt Export Sequences
Data Structure Type
Tabular Data
Allow Dynamic Excel Loads
True
Data Management Settings
Data Export Sequence Name
Budget Data Mgmt.Export Budget (Version 4)

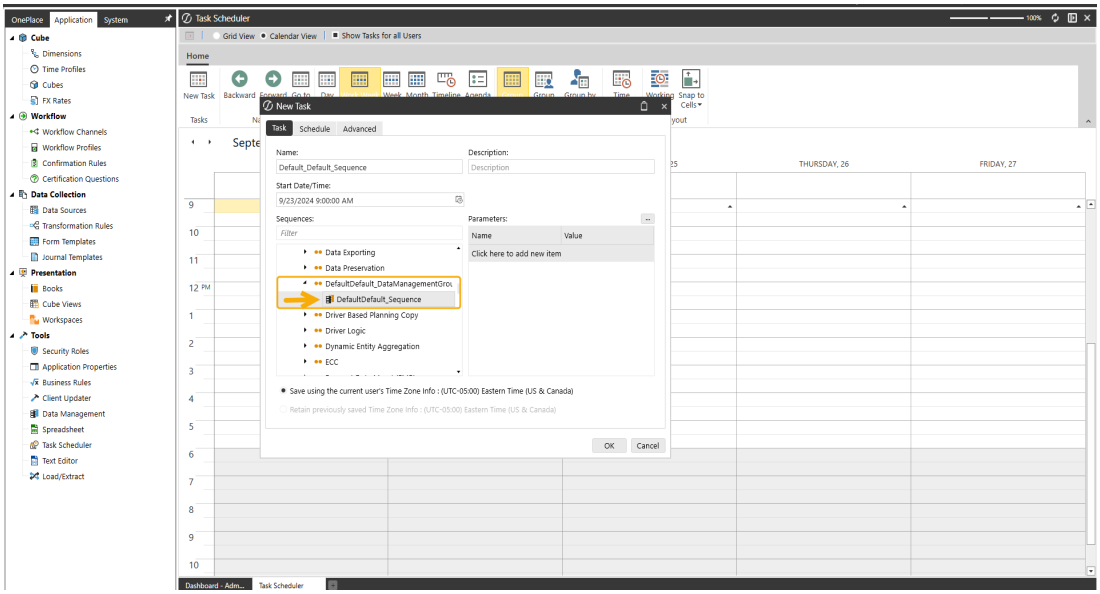
> **NOTE:** The workspace name will be prefixed to the Data Management Sequence Name if you are selecting a sequence outside the Default Workspace. For example, Budget Data Management workspace is prefixed to Export Budget (Version 4) sequence.

Task Scheduler in Workspaces

Task Scheduler allows you to schedule data management sequences that can run one or more data management steps within the application. You can reference the workspace path and sequence within the Task Scheduler to determine the tasks, sequences, or modifications are running correctly.

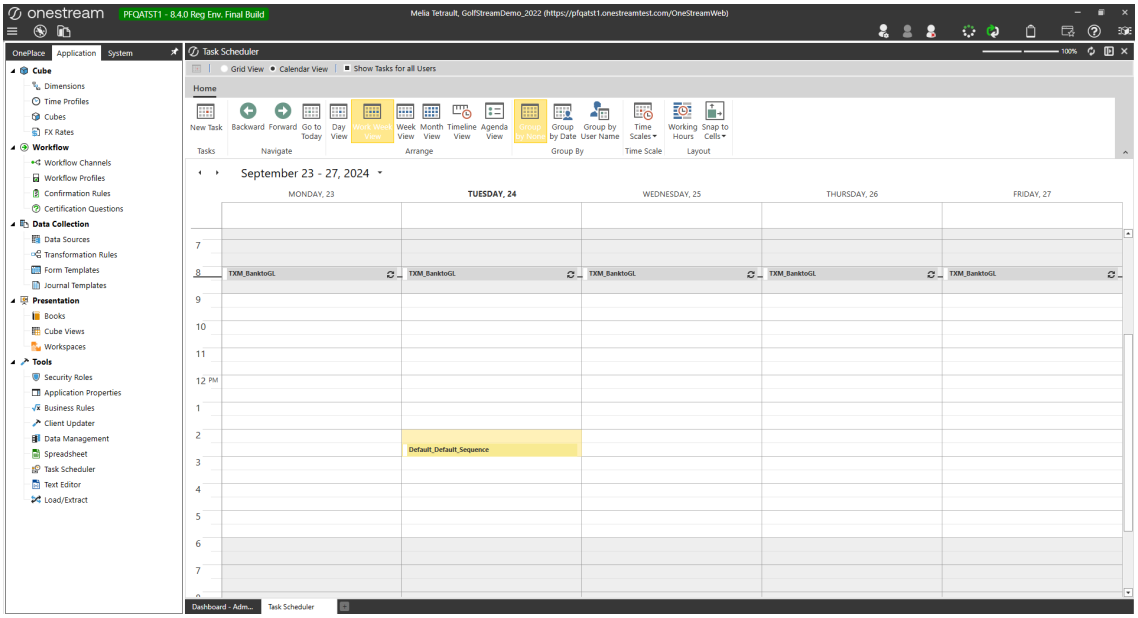
Display Task Scheduler in Workspaces

1. Under **Application Tools** > **Task Scheduler**, click **New Task**.
2. Enter the **Name**, **Description**, and **Start Date/Time** for the task.
3. On the **Schedule** tab, specify details for the task.
4. On the **Advanced** tab, select the number of times to retry a failed task.
5. Under **Sequences**, select the data management sequence under the workspace name and path, and click the **OK** button to add the task to the Grid View and the Calendar View.

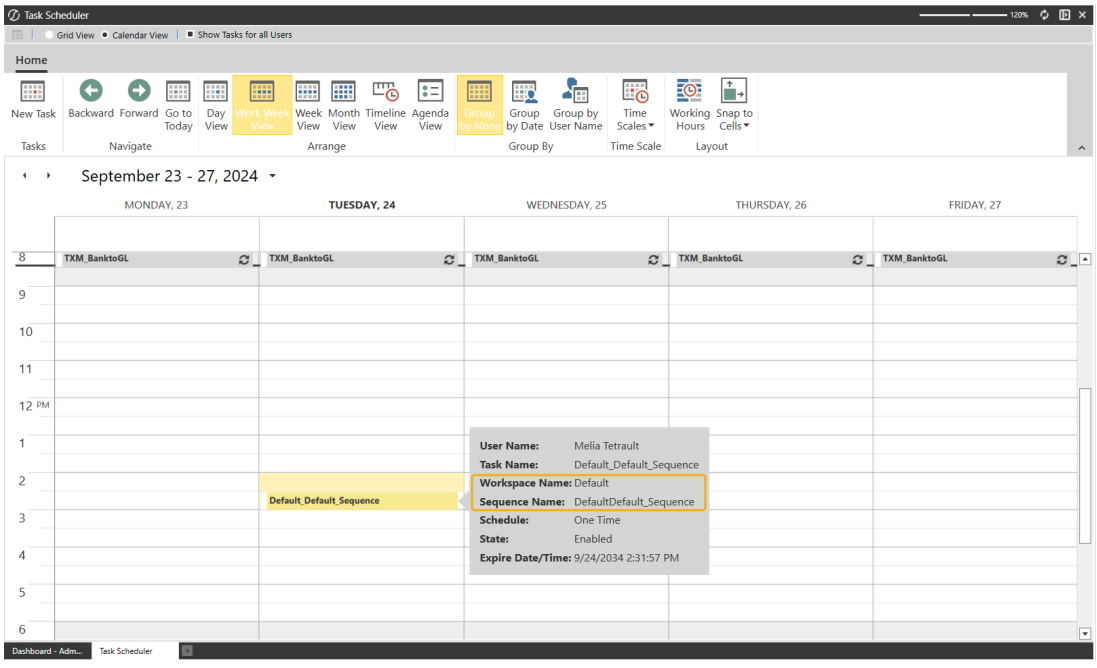


> **NOTE:** When selecting a data management sequence, if a workspace, a data management group, or a maintenance unit has no sequences in it, it will not display as an option to select under **Sequences**. If a workspace is not set to shareable, you cannot select the Workspace. .

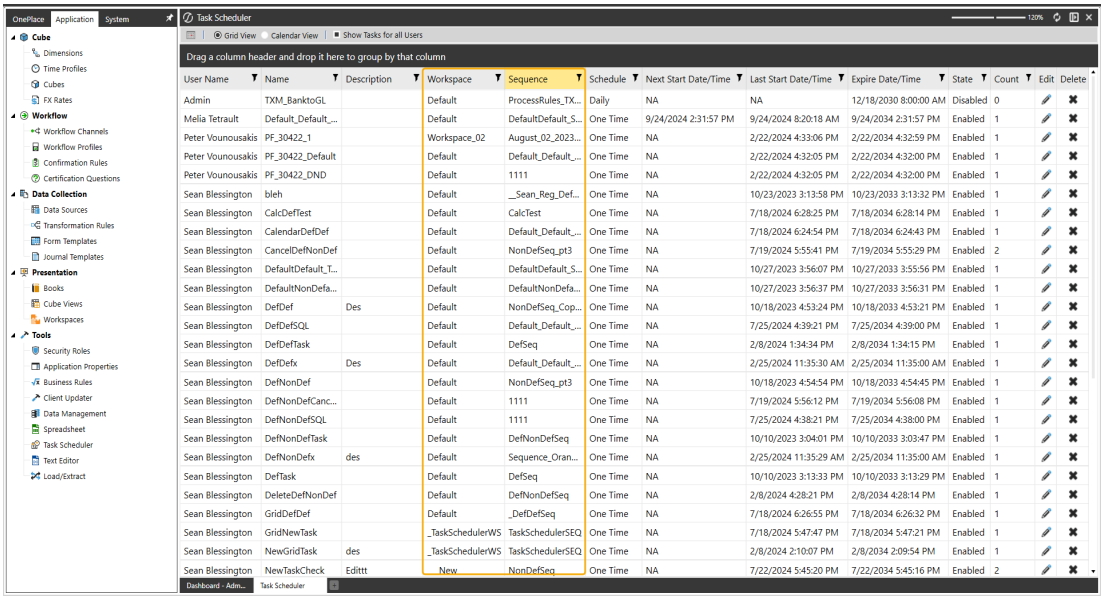
6. Under the **Task Scheduler** > **Calendar View**, the new Task is added and listed.



You can also view the workspace name and sequence name in the calendar pop under calendar view.



7. Under the **Task Scheduler** > **Grid View**, the task displays the Workspace and Sequence name.



> **NOTE:** You will not be able to delete a sequence on the Workspaces or Data Management pages if it is assigned to a Task Scheduler item.

Load and Extract for Data Management Objects

You can load and extract data management objects under Tools in the **Application** tab under **Load/Extract**.

- Load and Extract XML Application Workspaces
- Extract File Type Task Scheduler

Load and Extract XML Application Workspaces

You can import and export parts of the Application Workspaces for data management objects using an XML format.

XML Extract Application Workspaces

To extract Data Management Groups, follow these steps:

1. On the **Application** tab, under **Tools**, click **Load/Extract**.
2. Click the **Extract** tab and select Data Management as the file type.
3. Expand **Groups** and select data management groups to export.
4. Click to extract.

To extract the default workspace and default maintenance unit, follow these steps:

1. On the **Application** tab, under **Tools**, click **Load/Extract**.
2. Click the **Extract** tab and select Application Workspaces as the file type.
3. Click the **All** folder to de-select all items.
4. Expand **Workspaces** in the tree hierarchy and select the default workspace and default maintenance unit.
5. Click to extract.

To extract the default workspace and a non-default maintenance unit, follow these steps:

1. On the **Application** tab, under **Tools**, click **Load/Extract**.
2. Click the **Extract** tab and select Application Workspaces as the file type.
3. Click the **All** folder to de-select all items.

4. Expand **Workspaces** in the tree hierarchy and select the default workspace and the non-default maintenance unit.
5. Click to extract.

To extract a non-default workspace, follow these steps:

1. On the **Application** tab, under **Tools**, click **Load/Extract**.
2. Click the **Extract** tab and select Application Workspaces as the file type.
3. Click the **All** folder to de-select all items.
4. Expand **Workspaces** in the tree hierarchy and select the workspace.
5. Click to extract.

> **NOTE:** You can also use the search bar and click to search for items to extract in the tree hierarchy.

> **NOTE:** You cannot extract data management profiles since they will no longer be displayed in the user interface but still extract specific Data Management Groups in addition to extracting all of them at once. The tree hierarchy will display All and Groups for all data management groups to extract.

XML Load Application Workspaces

To load Data Management Groups, follow these steps:

1. On the **Application** tab, under **Tools**, click **Load/Extract**.
2. Click the **Load** tab and select an existing XML file of data management groups to load by

the ellipsis (...) or editor.

3. Click to load.

To load a default workspace and default maintenance unit, default workspace and non-default maintenance unit, or a non-default workspace, follow these steps:

1. On the **Application** tab, under **Tools**, click **Load/Extract**.
2. Select the **Load** tab and select the Application Workspaces XML file type to load by the ellipsis (...) or editor.
3. Click to load.

Extract File Type Task Scheduler

When you extract Task Scheduler items, the workspace name will be prefixed to the Data Management Sequence Name on the extract.

1. On the **Application** tab, under **Tools**, click **Load/Extract**.
2. Select the **Extract** tab and select Task Scheduler as the file type.
3. Select Task Scheduler items to extract.
4. Click to extract.

When you extract Task Scheduler items sequences in the default workspace/default maintenance unit, it will display on the extract. The Default> Default>Task Scheduler Item will list the Data Management Sequence name on the <DataMgmtSequenceName> line.

```
        <NumericDays />
        <ScheduleMonths />
        <OnWeeksOfMonth />
        <NumAllowedFailures>0</NumAllowedFailures>
        <EnableRunWindow>false</EnableRunWindow>
        <RunWindowStartTime>1900-01-01T00:00:00</RunWindowStartTime>
        <RunWindowEndTime>2200-12-31T00:00:00</RunWindowEndTime>
    </ItemDefinition>
</ScheduleItem>
<ScheduleItem>
    <Name>DefaultDefault_Extract</Name>
    <Description />
    <UserName>Sean Blessington</UserName>
    <DataMgmtSequenceName>___TaskSchedule_DefDef_Sequence</DataMgmtSequenceName>
    <DataMgmtSequenceArgs />
    <StartTime>2023-09-14T16:00:00</StartTime>
    <ExpirationTime>2200-12-31T05:00:00</ExpirationTime>
    <ManagerName>Sean Blessington</ManagerName>
    <IsEnabledByManager>true</IsEnabledByManager>
    <IsEnabledByUser>true</IsEnabledByUser>
    <ItemDefinition>
        <ScheduleFrequency>OneTime</ScheduleFrequency>
        <ScheduleEveryNumMinutes>0</ScheduleEveryNumMinutes>
        <ScheduleOccurEveryXDaysWeeks>0</ScheduleOccurEveryXDaysWeeks>
        <ScheduleOccurEveryXWeeks>0</ScheduleOccurEveryXWeeks>
        <ScheduleDays />
        <NumericDays />
        <ScheduleMonths />
        <OnWeeksOfMonth />
        <NumAllowedFailures>0</NumAllowedFailures>
        <EnableRunWindow>false</EnableRunWindow>
        <RunWindowStartTime>1900-01-01T00:00:00</RunWindowStartTime>
        <RunWindowEndTime>2200-12-31T00:00:00</RunWindowEndTime>
    </ItemDefinition>
</ScheduleItem>
<ScheduleItem>
    <Name>DefaultNonDefault_Extract</Name>
    <Description />
    <UserName>Sean Blessington</UserName>
    <DataMgmtSequenceName>___TaskSchedule_DefNonDef_Sequence</DataMgmtSequenceName>
    <DataMgmtSequenceArgs />
    <StartTime>2023-09-14T18:00:00</StartTime>
    <ExpirationTime>2200-12-31T05:00:00</ExpirationTime>
    <ManagerName>Sean Blessington</ManagerName>
    <IsEnabledByManager>true</IsEnabledByManager>
    <IsEnabledByUser>true</IsEnabledByUser>
    <ItemDefinition>
        <ScheduleFrequency>OneTime</ScheduleFrequency>
        <ScheduleEveryNumMinutes>0</ScheduleEveryNumMinutes>
        <ScheduleOccurEveryXDaysWeeks>0</ScheduleOccurEveryXDaysWeeks>
```

> **NOTE:** Since the Task Scheduler Item did not use a Non-Default Workspace>Sequence, the Data Management Sequence name within the file will not be prefixed by a Workspace Name.

When you extract Task Scheduler items sequences in the default workspace/non-default maintenance unit, it will display on the extract. The Default> Non-Default>Task Scheduler Item will list the Data Management Sequence name on the <DataMgmtSequenceName> line.

```
            <StartTime>2023-09-14T16:00:00</StartTime>
            <ExpirationTime>2200-12-31T05:00:00</ExpirationTime>
            <ManagerName>                    /ManagerName>
            <IsEnabledByManager>true</IsEnabledByManager>
            <IsEnabledByUser>true</IsEnabledByUser>
            <ItemDefinition>
                <ScheduleFrequency>OneTime</ScheduleFrequency>
                <ScheduleEveryNumMinutes>0</ScheduleEveryNumMinutes>
                <ScheduleOccurEveryXDaysWeeks>0</ScheduleOccurEveryXDaysWeeks>
                <ScheduleOccurEveryXWeeks>0</ScheduleOccurEveryXWeeks>
                <ScheduleDays />
                <NumericDays />
                <ScheduleMonths />
                <OnWeeksOfMonth />
                <NumAllowedFailures>0</NumAllowedFailures>
                <EnableRunWindow>false</EnableRunWindow>
                <RunWindowStartTime>1900-01-01T00:00:00</RunWindowStartTime>
                <RunWindowEndTime>2200-12-31T00:00:00</RunWindowEndTime>
            </ItemDefinition>
        </ScheduleItem>
        <ScheduleItem>
            <Name>DefaultNonDefault_Extract</Name>
            <Description />
            <UserName>                    </UserName>
            <DataMgmtSequenceName>___TaskSchedule_DefNonDef_Sequence</DataMgmtSequenceName>
            <DataMgmtSequenceArgs />
            <StartTime>2023-09-14T18:00:00</StartTime>
            <ExpirationTime>2200-12-31T05:00:00</ExpirationTime>
            <ManagerName>                    /ManagerName>
            <IsEnabledByManager>true</IsEnabledByManager>
            <IsEnabledByUser>true</IsEnabledByUser>
            <ItemDefinition>
                <ScheduleFrequency>OneTime</ScheduleFrequency>
                <ScheduleEveryNumMinutes>0</ScheduleEveryNumMinutes>
                <ScheduleOccurEveryXDaysWeeks>0</ScheduleOccurEveryXDaysWeeks>
                <ScheduleOccurEveryXWeeks>0</ScheduleOccurEveryXWeeks>
                <ScheduleDays />
                <NumericDays />
                <ScheduleMonths />
                <OnWeeksOfMonth />
                <NumAllowedFailures>0</NumAllowedFailures>
                <EnableRunWindow>false</EnableRunWindow>
                <RunWindowStartTime>1900-01-01T00:00:00</RunWindowStartTime>
                <RunWindowEndTime>2200-12-31T00:00:00</RunWindowEndTime>
            </ItemDefinition>
        </ScheduleItem>
    </taskScheduler>
</taskScheduler>
</OneStreamXF>
```

> **NOTE:** Since the Task Scheduler Item did not use a Non-Default Workspace>Sequence, the Data Management Sequence name within the file will not be prefixed by a Workspace Name.

When you extract Task Scheduler items sequences in a non default workspace, it will display on the extract. The Non-Default Workspace, Task Scheduler Item will list the correct Data Management Sequence name on the <DataMgmtSequenceName> line.

```
<?xml version="1.0" encoding="utf-8"?>
<OneStreamXF version="8.1.0.15312">
    <taskScheduler>
        <taskScheduler>
            <ScheduleItem>
                <Name>NonDefault_Extract</Name>
                <Description />
                <UserName>                    </UserName>
                <DataMgmtSequenceName>(MST) XYZ_TestSolution.___TaskSchedule_NonDef_Sequence</DataMgmtSequenceName>
                <DataMgmtSequenceArgs />
                <StartTime>2023-09-14T20:00:00</StartTime>
                <ExpirationTime>2200-12-31T05:00:00</ExpirationTime>
                <ManagerName>                    </ManagerName>
                <IsEnabledByManager>true</IsEnabledByManager>
                <IsEnabledByUser>true</IsEnabledByUser>
                <ItemDefinition>
                    <ScheduleFrequency>OneTime</ScheduleFrequency>
                    <ScheduleEveryNumMinutes>0</ScheduleEveryNumMinutes>
                    <ScheduleOccurEveryXDaysWeeks>0</ScheduleOccurEveryXDaysWeeks>
                    <ScheduleOccurEveryXWeeks>0</ScheduleOccurEveryXWeeks>
                    <ScheduleDays />
                    <NumericDays />
                    <ScheduleMonths />
                    <OnWeeksOfMonth />
                    <NumAllowedFailures>0</NumAllowedFailures>
                    <EnableRunWindow>false</EnableRunWindow>
                    <RunWindowStartTime>1900-01-01T00:00:00</RunWindowStartTime>
                    <RunWindowEndTime>2200-12-31T00:00:00</RunWindowEndTime>
                </ItemDefinition>
            </ScheduleItem>
            <ScheduleItem>
                <Name>DefaultDefault_Extract</Name>
                <Description />
                <UserName>                    </UserName>
                <DataMgmtSequenceName>___TaskSchedule_DefDef_Sequence</DataMgmtSequenceName>
                <DataMgmtSequenceArgs />
                <StartTime>2023-09-14T16:00:00</StartTime>
                <ExpirationTime>2200-12-31T05:00:00</ExpirationTime>
                <ManagerName>                    </ManagerName>
                <IsEnabledByManager>true</IsEnabledByManager>
                <IsEnabledByUser>true</IsEnabledByUser>
                <ItemDefinition>
                    <ScheduleFrequency>OneTime</ScheduleFrequency>
                    <ScheduleEveryNumMinutes>0</ScheduleEveryNumMinutes>
                    <ScheduleOccurEveryXDaysWeeks>0</ScheduleOccurEveryXDaysWeeks>
                    <ScheduleOccurEveryXWeeks>0</ScheduleOccurEveryXWeeks>
                    <ScheduleDays />
                    <NumericDays />
                    <ScheduleMonths />
                    <OnWeeksOfMonth />
```

NOTE: Since the Task Scheduler Item used a Non-Default Workspace>Sequence, the Data Management Sequence name within the file will contain the Workspace Name as a prefix.

Example: If a Task Scheduler Item was created using a Sequence located in the Non-Default Workspace and NonDefaultWorkspace_Sequence" Data Management Group, it's <DataMgmtSequenceName> line should read: "**NonDefaultWorkspace.NonDefaultWorkspace___Sequence</DataMgmtSequenceName>**".

```
<?xml version="1.0" encoding="utf-8"?>
<OneStreamXF version="8.1.0.15423">
    <taskScheduler>
        <taskScheduler>
            <ScheduleItem>
                <Name>NonDefaultWorkspace_Task</Name>
                <Description />
                <UserName>[redacted]</UserName>
                <DataMgmtSequenceName>NonDefaultWorkspace.NonDefaultWorkspace_Sequence</DataMgmtSequenceName>
                <DataMgmtSequenceArgs />
                <StartTime>2023-10-27T19:58:36.29</StartTime>
                <ExpirationTime>2200-12-31T05:00:00</ExpirationTime>
                <ManagerName>[redacted]</ManagerName>
                <IsEnabledByManager>true</IsEnabledByManager>
                <IsEnabledByUser>true</IsEnabledByUser>
                <ItemDefinition>
                    <ScheduleFrequency>OneTime</ScheduleFrequency>
                    <ScheduleEveryNumMinutes>0</ScheduleEveryNumMinutes>
                    <ScheduleOccurEveryXDaysWeeks>0</ScheduleOccurEveryXDaysWeeks>
                    <ScheduleOccurEveryXWeeks>0</ScheduleOccurEveryXWeeks>
                    <ScheduleDays />
                    <NumericDays />
                    <ScheduleMonths />
                    <OnWeeksOfMonth />
                    <NumAllowedFailures>0</NumAllowedFailures>
                    <EnableRunWindow>false</EnableRunWindow>
                    <RunWindowStartTime>1900-01-01T00:00:00</RunWindowStartTime>
                    <RunWindowEndTime>2200-12-31T00:00:00</RunWindowEndTime>
                </ItemDefinition>
            </ScheduleItem>
```

Substitution Variable Items

Workspace Substitution Variable Items are a collection of static name and value pair properties that can be retrieved in dashboarding or code. Utilizing a substitution variable permits a copied solution to only need the substitution variable value changed. For example, myschema1 to myschema2. Name/Value pairs are added to a property at the Workspace level and referenced on Dashboards/Components.

> **TIP:** Substitution variables can also be used elsewhere throughout Workspaces such as inside labels, images, or button image URLs.

Substitution variable items can be viewed, edited, or deleted through two general properties:

Substitution Variable Name Suffix
Enter the suffix for a substitution variable prefixed by "WSSV" to use within the Workspace.

Substitution Variable Value
Enter the value to replace the processed substitution variable.

Substitution Variable Syntax

|WSSVSUFFIX|

The syntax for referencing the substitution variable with a suffix of "Suffix".

|WSSV{NameOfSubstitutionVariable}|

The syntax for a Workspace substitution variable.

|WSSV{Suffix}|

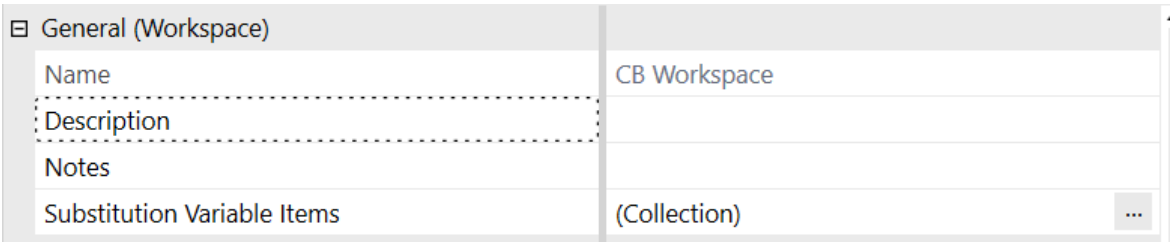
The syntax within a Workspace running a dashboard that references the substitution variable property.

> **NOTE:** The substitution variable can also be retrieved in code through the DashboardWorkspace object provided to each of the factory services.

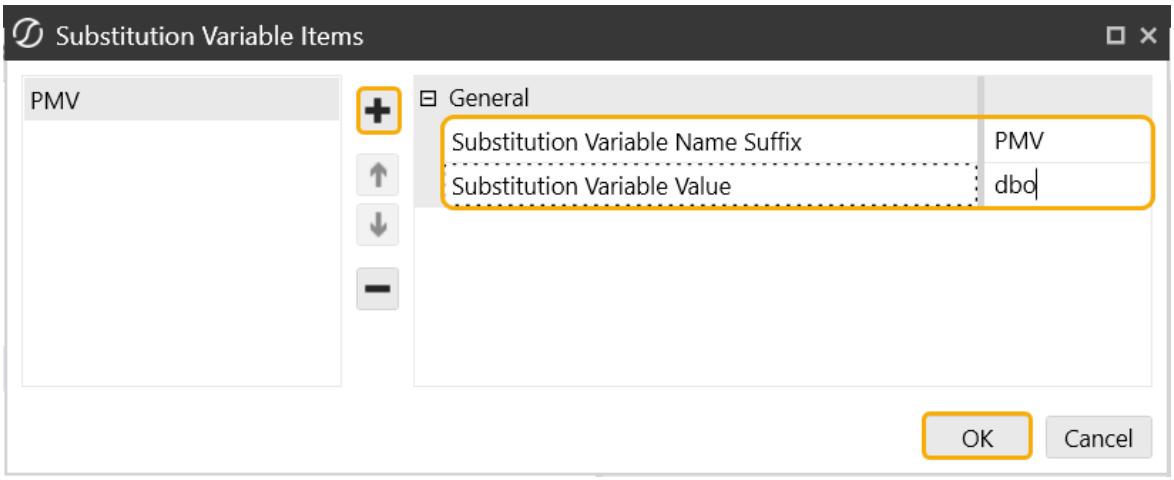
Add a Workspace Substitution Variable

Follow these steps to add a Workspace Substitution Variable:

1. Navigate to **Application** > **Presentation** > **Workspaces**. Select or create a new Workspace.
2. Under the General (Workspace) section, click the ellipses (...) next to **Substitution Variable Items**.



3. The Substitution Variable Items dialog box appears. Click on the **Add Item** button to add a variable.
4. Enter values into the Substitution Variable Name Suffix and Substitution Variable Value.



5. Select the **OK** button, then select the **Save** button on the Workspace.

> **TIP:** You can remove a variable through the **Remove Selected Item** button and move variables through the **Move Up or Move Down** buttons in the Substitution Variable Items dialog box.

Find and Assign Values Using Object Lookup

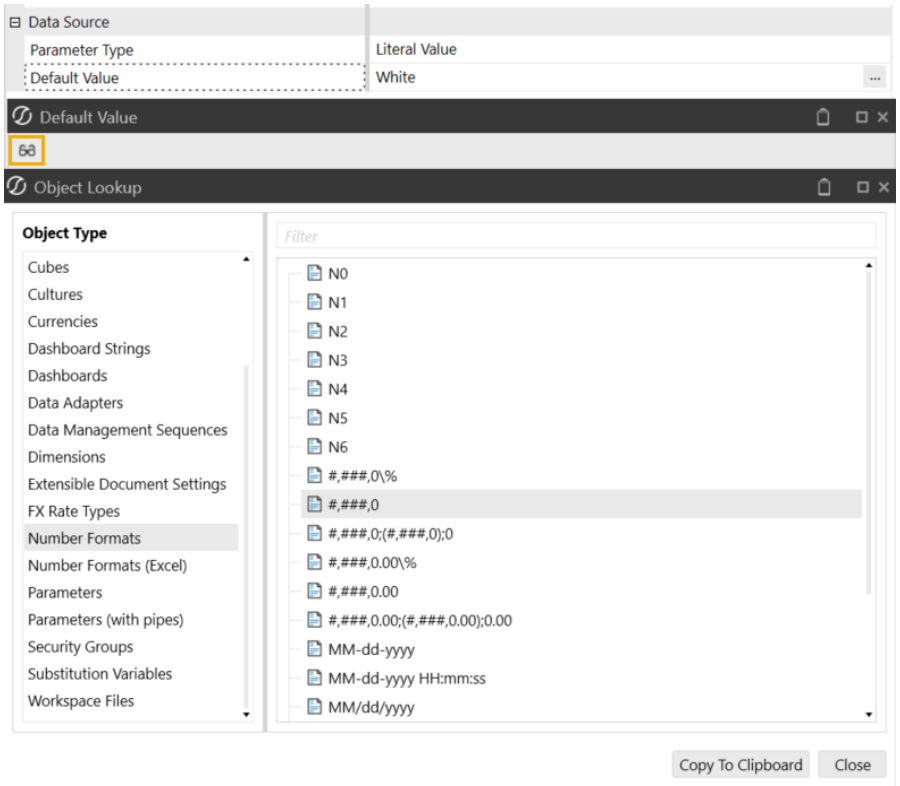
Use the **Object Lookup** dialog box throughout OneStream to search for object to assign rather than having to remember a specific object name or format. The **Object Lookup** dialog box is useful when you do not know the value or object to apply with a parameter or to display as the default value the first time a parameter runs.

In an Input Value parameter, the default value is the object or value the parameter applies when assigned to an object. For the other parameter types, the default value is the object or value to initially display when a parameter first runs. For example, to display a specific member at the top of a list of six members in a member list parameter, specify that member name as the default value.

Object Lookup enables you to browse application values and objects, such as business rules, colors, cube views, dimensions, number formats, and then copy and paste them where needed. Instead of entering a default value, use Object Lookup as follows to find and assign a value or object.

For example, to use Object Lookup to a default value to a parameter, first create the parameter, then do the following:

1. In the parameter's **Default Value**, click **Edit** and then **Object Lookup**.

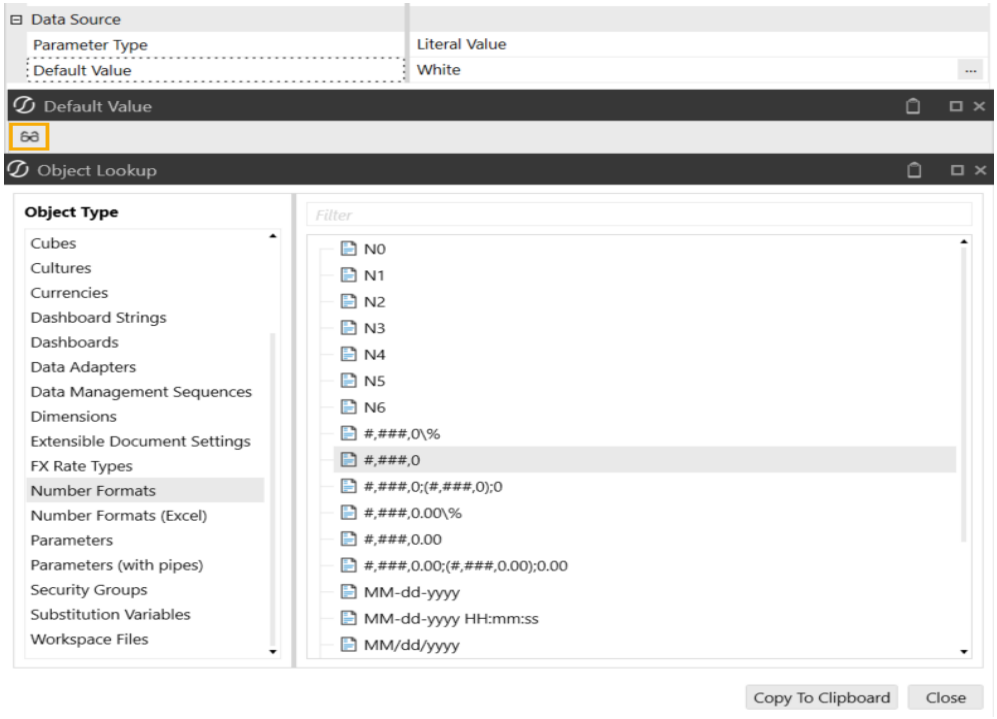


2. In the **Object Lookup** dialog box, select an object type to the left to browse available objects or values. To refine the objects displayed, filter by name.
3. Select the object and click **Copy To Clipboard**.
4. In the **Default Value** dialog box, right-click, select **Paste**, and then click **OK**.

To remove or use another value or object, delete the contents in the **Default Value** dialog box and click **Object Lookup**.

Parameters (with Pipes) Example

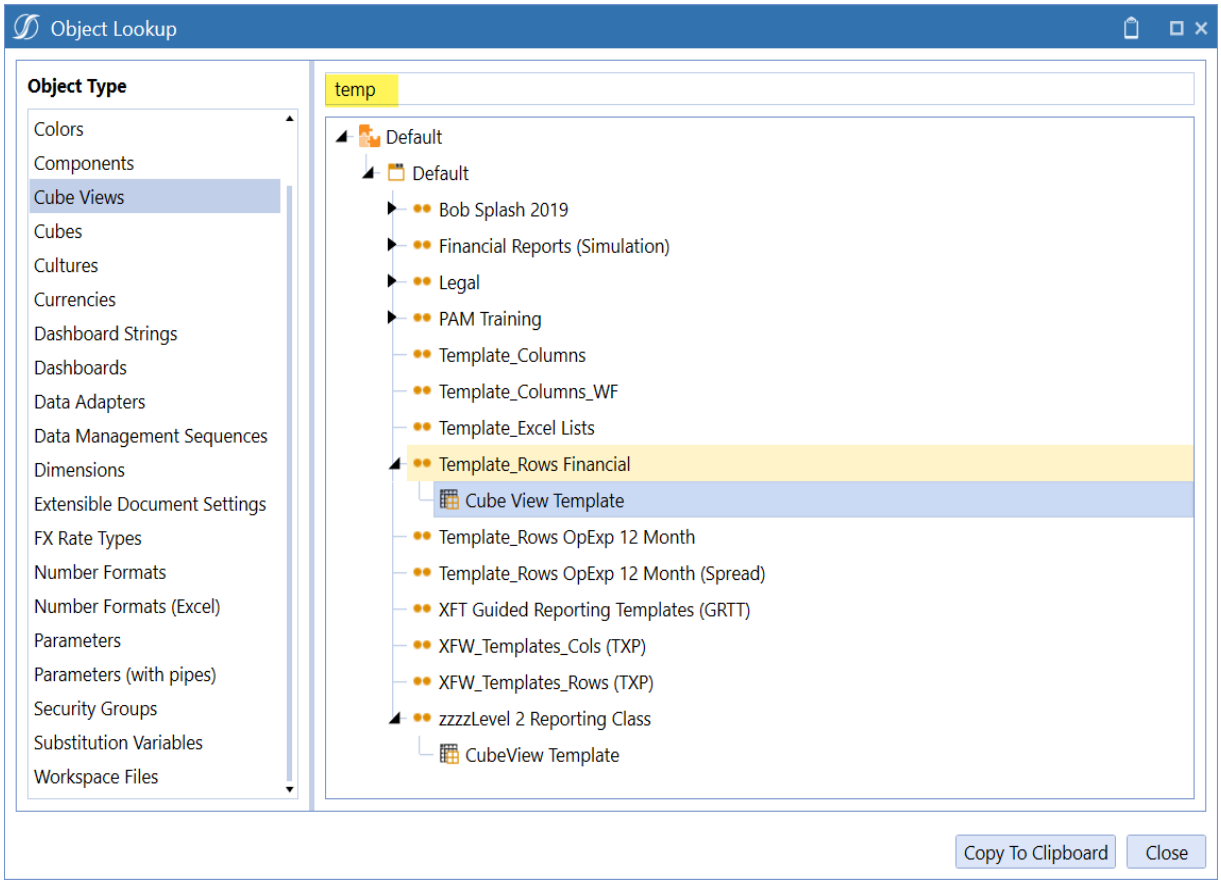
When designing a cube view header, click the **Object Lookup** icon and select **Parameters (with Pipes)** from the Object Type menu. If the name of the specific parameter is known, begin typing it in the filter at the top of the dialog box. Select the desired cube view styles parameter, such as `DefaultHeader`, and click **Copy to Clipboard** or click CTRL/double-click to copy.



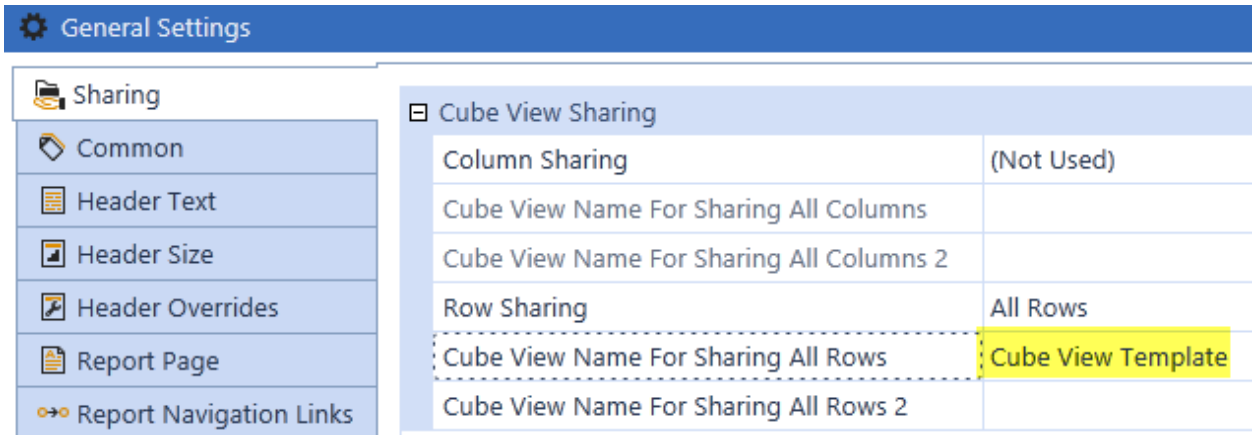
Next, click into the empty field where the copied parameter needs to go, and click CTRL+V. This pastes the parameter into the Cube View field.

Cube Views Example

Another example of how this feature can be used is when using Cube View Templates to create Cube View Rows or Columns. Open the Object Lookup, and begin typing the desired Cube View Template name, select the Cube View Template, and click Copy to Clipboard.



Then paste it into the Cube View Name For Sharing All Rows property.



Extensible Document Settings

You can find each of the strings for Extensible Document Settings in the Object Lookup dialog. There is a variety of item types that can be used in Extensible Documents, however each one needs to be configured a specific way to process correctly at run-time. An example of each item type's configuration is provided below. For more details on how to utilize and configure these item types in an Extensible Document, see Extensible Document Framework in Presenting Data With Extensible Documents.

Insert Content Using Document Variables

Cube View Report

To insert a Cube View Report into an Extensible Document, the following string needs to be updated and pasted when configuring the document variables:

"{XF}{Application}{CubeViewReport}{CubeViewName}"

(Optional formatting): "IncludeReportMargins=False" "IncludeReportHeader=False" "IncludeReportFooter=False" "IncludePageHeader=True" "IncludePageFooter=True"

> **IMPORTANT:** Arguments such as the "CubeViewName" are placeholders for Cube Views, Reports, and Files that must be updated.

Example

"{XF}{Application}{CubeViewReport}{CubeViewStatement}"

Excel Sheet

To insert an Excel Sheet into an Extensible Document, the following string needs to be updated and pasted when configuring the document variables:

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/ExcelFileName.xfDoc.xlsx}"

(Optional formatting): “ExcelSheet=Sheet1"

> **IMPORTANT:** The specific path to the File in the File Explorer must be included and updated.

Example

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/Excel4.xfDoc.xlsx} "

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/Excel4.xfDoc.xlsx}"
"ExcelSheet=Sheet1"

Excel Named Range

To insert an Excel Named Range into an Extensible Document, the following string needs to be updated and pasted when configuring the document variables:

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/ExcelFileName.xfDoc.xlsx}"

(Optional formatting): “ExcelNamedRange=SampleNamedRange"

Example

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/Excel5.xfDoc.xlsx} "

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/Excel4.xfDoc.xlsx}"
"ExcelNamedRange=NamedRange1"

Microsoft Word Document
To insert a Microsoft Word Document into an Extensible Document, the following string needs to be updated and pasted when configuring the document variables:

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/WordFileName.docx}"

Example

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/OneStreamWordDocument.docx}"

Rich Text Document
To insert a Rich Text Document into an Extensible Document, the following string needs to be updated and pasted when configuring the document variables:

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/RTFFileName.rtf}"

Example

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/RichTextFormatDoc.rtf}"

Text Document
To insert a Text Document into an Extensible Document, the following string needs to be updated and pasted when configuring the document variables:

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/TextFileName.txt}"

Example

"{XF}{Application}{File}{Documents/Public/ExtensibleDocs/TextFileData.txt}"

Report
To insert a Report into an Extensible Document, the following string needs to be updated and pasted when configuring the document variables:

"{XF}{Application}{Report}{ReportComponentName}"

(Optional formatting): “IncludeReportMargins=False” “IncludeReportHeader=False” “IncludeReportFooter=False” “IncludePageHeader=True” “IncludePageFooter=True”

Example

"{XF}{Application}{Report}{Report3}"

> **NOTE:** Parameters and Substitution Variables are not listed under Extensible Document Settings.

Insert Content Using Office Image

Chart/Chart Report

To insert a Chart or Chart Report Dashboard Component into an Extensible Document, the following string needs to be updated and pasted in the Content Control Properties dialog:

```
{XF}{ItemLocation}{ItemType}{ChartComponentName}
```

Example

{XF}{Application}{Chart}{Waterfall}

{XF}{Application}{ChartReport}{Waterfall}

> **NOTE:**
> Use this for all Chart (Advanced) Dashboard Components.

Cube View Report

To insert a Cube View Report into an Extensible Document, the following string needs to be updated and pasted in the Content Control Properties dialog when configuring the image:

{XF}{Application}{ItemType}{CubeViewName}

Example

{XF}{Application}{CubeViewReport}{BalanceSheetSummary}

Excel Sheet/Excel Named Range

To insert an Excel Sheet or Excel Named Range into an Extensible Document, the following string needs to be updated and pasted in the Content Control Properties dialo when configuring the image:

{XF}{Application}{ItemType}{FilePath}

Example

{XF}{Application}{ExcelFile}{Documents/Users/jsmith/Favorites/VarianceReport.xfDoc.xlsx}

> **NOTE:** For Excel Named Range Item Types, the Excel Named Range Name is configured in the formatting string. (e.g., ExcelNamedRange=TotalAssets)

PDF

To insert a PDF into an Extensible Document, the following string needs to be updated and pasted in the Content Control Properties dialog when configuring the image:

{XF}{Application}{ItemType}{FilePath}

Example

{XF}{Application}{FileViaPDF}{Documents/Users/jsmith/Favorites/IS.pdf}

Report

To insert a Report Dashboard Component into an Extensible Document, the following string needs to be updated and pasted in the Content Control Properties dialog when configuring the image:

{XF}{ItemLocation}{ItemType}{ReportComponentName}

Example

{XF}{Application}{Report}{UserTaskActivity}

Rich Text

To insert Rich Text into an Extensible Document, the following string needs to be updated and pasted into the Alt Text Description Field for the specific image:

{XF}{Application}{RichText}{RichTextContent}

(Optional formatting): PageNumber=1, CropLeft=0, CropTop=0, CropWidth=0, CropHeight=0, Zoom=100, MaintainAspectRatio=True, FillMode=Width, Anchor=TopLeft, IncludeBorders=False, BackgroundColor=#FFFFFF, MeasurementUnit=Inch, PageWidth=8.5, PageHeight=11

Example

{XF}{Application}{RichText}{|!Narrative2!|} BackgroundColor=#C1E5F5, PageNumber=1, IncludeBorders=False

> **NOTE:** "Narrative2" is a parameter containing a specific Rich Text. You can also paste the specific rich text format in the string.

Options

The following options are available when formatting Extensible Document Image Content. Each Image Type has a formatting string which is located in the Object Lookup Dialog under Extensible Document Settings|Insert Content in Office Image. The user can copy and paste the desired formatting string into the Description field when configuring the image. If changes need to be made to the string, delete the current option and replace it with the correct one. The list below covers all of the formatting options for each Image Type.

Item Location

Application, System

Item Type

Chart, ChartReport, CubeViewReport, ExcelFile, FileViaPDF, Report

FillMode Options

Width, Height, LargestSide, SmallestSide

Anchor Options

BottomCenter/Left/Right, MiddleCenter/Left/Right, TopCenter/Left/Right

Cropping Options

This allows a user to narrow in on a portion of the image before other settings are applied. The default setting is 0 which means cropping is not being used.

CropLeft, CropTop, CropWidth, CropHeight

XFCell

XFCell is a retrieve function used mainly in text documents such as Microsoft Word or PowerPoint. The specific Dimension details provided in the function obtains a single cell of data from OneStream and displays the updated value on an Extensible Document at run-time. Examples of common XFCell formulas are provided below. See Extensible Document Framework in Presenting Data With Extensible Documents for more details on how to configure XFCells in an Extensible Document.

The following example pulls data for a specific Entity and Account:

XFCell(E#US:A#Sales)

Additional Dimension Members can be added for more specific detail:

XFCell(E#US:A#Sales:T#2014)

> **NOTE:** Any Dimensions not specified in the formula will come from the user's POV.

Number formatting and scaling can also be added to an XFCell.

The example below is using the number format N3 with a scaling of 6

XFCell(E#US:A#Sales, NumberFormat=N3, Scale=6)

Culture Invariant is a default culture not associated with a specific country. It is typically used when users need to convert a number to a string, but do not want the result to be different if one user is running the string on a French PC and another user is using an English PC.
XFCell(E#US:A#Sales, Culture=Invariant)

Include Member Scripts in XFCells in order to retrieve data for specific Dimension Members. Culture User is based on the computer's Windows settings and the User settings.
XFCell(Memberscript, Culture=User, NumberFormat=N3, DisplayNoDataAsZero=True, Scale=3, FlipSign=True, ShowPercentageSign=False)

Use Business Rules in Extensible Documents

Business Rules can also be referenced in an XFCell function. The following details need to be updated for this to work correctly.

BR#[BRName, FunctionName=yourFunctionName, Name1=Value1, AnotherName=[AnotherValue]]

Example
BR#[BRName=FMK_Helper, FunctionName=GetUserSetting, SolutionID=MBL, FieldName=[Period]]

Application Tools

Use application tools to manage how you work with the OneStream application, configuring applications to best suit your needs. Application tools include setting application security roles, setting application properties, and scheduling data management sequences. In this section you will learn how to use these tools and others to manage the application environment.

Application Security Roles

Below are the specific Application-level security roles and what they control:

Administer Application
This role allows a user to administer the application and load zip files. This is useful when multiple applications exist in one environment and different groups of administrators/users need to administer separate applications.

Administrator Database
This application-level role is intended for a few people that are allowed to mass delete metadata and data, primarily using the database page.

This roleType is unlike most other roleTypes because Administrators are not automatically given access to operations that require this role.

Open Application
This allows the user to see and open the application.

Modify Data
This allows the user to modify data. The user is basically a read-only user throughout this application if he/she does not have this role.

View All Data

This allows a group of users to view all data in the application.

Create Audit Attachments

This allows the user to create data attachments for supporting documentation.

Create Footnote Attachments

This allows the user to create a footnote attachment for supporting documentation.

Certify and Lock Descendants

This allows the user to certify and lock descendants from the Workflow. This is typically an administrator function.

Unlock and UnCertify Ancestors

This allows the user to uncertify and unlock ancestors from the Workflow. This is typically an administrator function.

Preserve Import Data

The administrator will lock the Workflows and then preserve imported data when changes need to be made. The Workflow can then be unlocked so changes can be made.

Restore Import Data

This allows the administrator to restore imported data to the original state from the Preserve Import process.

Unlock Workflow Unit

This allows a user to unlock a Workflow Unit, however, the user must also have Workflow Execution access in order to lock a Workflow Unit.

View Source Data Audit

This allows a user to view the Source Data Audit Report within the Import Workflow.

Encrypt Business Rules
This allows a user to Encrypt and Decrypt a rule from the Business Rule screen in the Application tab, if the user is in the role.

Manage Application Properties
This allows a user to update this application's properties.

Manage Metadata
This allows a user to edit metadata under the Dimension Library for this application.

Manage FX Rates
This allows a user to update FX Rates.

Manage Data
This allows users to manage data in all aspects included, but not limited to exporting data and clearing data completed through Data Management. This is typically an administrator function.

Manage Cube Views
This allows a user to create new Cube Views and manage Cube View Groups and Profiles.

Manage Data Sources
This allows a user to create new Data Sources.

Manage Transformation Rules
This allows a user to create new Transformation Rules and manage Transformation Rules Groups and Profiles.

Manage Confirmation Rules
This allows a user to create new Confirmation Rules and manage Confirmation Rules Groups and Profiles.

Manage Certification Questions
This allows a user to create new Certification Questions and manage Certification Question Groups and Profiles.

Manage Workflow Channels
This allows a user to create new Workflow Channels.

Manage Workflow Profiles
This allows a user to create new Workflow Profiles.

Manage Journal Templates
This allows a user to create new Journal Templates and manage Journal Groups and Profiles.

Manage Form Templates
This allows a user to create new Form Templates and manage Forms Groups and Profiles.

Manage Application Dashboards
This allows a user to create new Application Dashboards and manage Dashboard Groups and Profiles.

Manage Application Database Files
Two file systems which are stored in the Framework database (i.e., the System database) and each Application database. Users in the security roles for ManageSystemDatabaseFiles and ManageApplicationDatabaseFiles have full read and write access to his/her user folders in those two database file systems, respectively. These folders are private to the user and access is intentionally restricted to just the user and managers. Security cannot be edited for a user folder. Users can be given read and/or write access to specific folders in the database file systems using the individual folders' security settings, however this excludes access to User folders and sub-folders.

Below are the specific Application-Level User Interface Roles and what they control:

Application Load Extract Page
This gives access to the Load/Extract screen located in |Application |Tools|. This is typically restricted to administrators.

Application Properties Page

This gives access to the Application Properties screen located in |Application |Tools|. This is typically restricted to administrators.

Application Security Roles Page

This gives access to the Application Security screen located in |Application |Tools|. This is typically restricted to administrators.

BookAdminPage

This gives access to the Book Designer screen located in |Application|Presentation|. This is typically restricted to administrators, or any users who create Report Books.

Business Rules Page

This gives access to the Business Rules screen located in |Application |Tools|. This is typically restricted to administrators.

Certification Questions Page

This gives access to the Certification Questions screen located in |Application| Workflow|. This is typically restricted to administrators.

Confirmation Rules Page

This gives access to the Confirmation Rules screen located in |Application |Workflow|. This is typically restricted to administrators.

Cube Admin Page

This gives access to the Cube Admin screen located in |Application |Cube|. This is typically restricted to administrators.

Cube Views Page

This gives access to the Cube Views screen located in |Application |Presentation|. This is typically restricted to administrators, or any users who create Cube Views.

Dashboard Admin Page

This gives access to the Dashboard Admin screen located in |Application |Presentation|. This is typically restricted to administrators.

Data Management Admin Page

This gives access to the Data Management Admin screen located in |Application |Tools|. This is typically restricted to administrators.

Data Sources Page

This gives access to the Data Sources screen located in |Application |Data Collection|. This is typically restricted to administrators.

Dimension Library Page

This gives access to the Dimension Library screen located in |Application |Cube|. This is typically restricted to administrators.

FX Rates Page

This gives access to the FX Rates screen located in |Application |Cube|. This is typically restricted to administrators.

Form Templates Page

This gives access to the Form Templates screen located in |Application |Data Collection|. This is typically restricted to administrators.

Journal Templates Page

This gives access to the Journal Templates screen located in |Application |Data Collection|. This is typically restricted to administrators.

Transformation Rules Page

This gives access to the Transformation Rules screen located in |Application |Data Collection|. This is typically restricted to administrators.

Workflow Channels Page
This gives access to the Workflow Channels screen located in |Application |Workflow|. This is typically restricted to administrators.

Workflow Profiles Page
This gives access to the Workflow Profiles screen located in |Application |Workflow|. This is typically restricted to administrators.

> **NOTE:** Click 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, click CTRL and Double Click. This will enter the correct name into the appropriate field.

AnalyticsApi
This role allows a user to access OneStream connectors to pull data into external systems and to control who can use the connectors. The role becomes available when access to the Power BI Connector is granted. The role may be available for use with additional connectors in the future.

Application Properties

This is where default properties are set for the application and for properties that differ by Scenario Type.

General Properties

Global Point of View

These are enabled when forcing Global Scenario and Global Time through Transformation settings. An initial value should be configured even if the Transformation setting is not being used.

Global Scenario
This is the default Scenario users will see when looking at Workflow.

Global Time
This is the default Time users will see when looking at Workflow.

Company Information

Company Name
Place company name here for it to appear on automatically generated reports from Cube Views.

Logo File (png, height ~ 50 pixels)
Attach a logo file in order for it to appear on Cube Views and Reports. An image format of PNG is required approximately 50 pixels high.

Workflow Channels

UD Dimension Type for Workflow Channels
The Origin Dimension controls data load, but in some cases other User Defined Dimensions require their own layer of locking. For example, a company plans by Entity by Product. One Entity can have five products done by different people. Each channel can be locked separately to protect that layer of data instead of locking the entire Entity.

Formatting

Number Format
This shows the format for numeric values displayed throughout the application. Configure to show additional degrees of precision to the right of the decimal point. This setting can be overridden through Cube View formatting.

N0
This setting will not show any decimals or zeroes

N1-N6

These settings will show X amount of decimals. If N2 is chosen, two decimals are displayed, N5 will display five decimals, etc.

> **NOTE:** The N in the above settings indicates that these settings are international.

The following formats use 10000.001 as an example.

#,###,0\%

This returns 10,000% and -10,000%

#,###,0

This returns 10,000 and -10,000

The three sections in a number format, separated by semi-colons, represent the format for positive numbers, negative numbers and zeros.

#,###,0;(#,###,0);0

This returns 10,000 and (10,000)

#,###,0.00%

This returns 10,000.00% and -10,000.00%

#,###,0.00

This returns 10,000.00 and -10,000.00

#,###,0.00;(#,###,0.00);0.00

This returns 10,000.00 and (10,000.00)

> **NOTE:** In order to vertically align positive and negative numbers in reports where parenthesis are used for negative values, include trailing spaces in the positive number format. This will account for the trailing parenthesis used by negative numbers. Example: #,###,0.00 ;(#,###,0.00);0.00

> **NOTE:** Click , select the desired format, and click CTRL and Double Click. This will enter the correct format into the appropriate field.

Currencies

All currencies used in the application must be listed here in order to be used on the Entity, to do any translation of currency, or enter any rates. The list of currencies will include any available currencies that are pre-Euro, or phased out currencies for historical data loading purposes. If a now defunct or new currency is not listed and is needed for the application, call OneStream Support.

Transformation

Enforce Global POV

If set to True, this will enforce the current Global Scenario and Time setting for all users, so they cannot change their Workflow View. If the Global POV is enforced, will display on the Import task during the Workflow.

Allow Loads Before Workflow View Year

If set to True, this will enforce the current Workflow View setting to allow data loading to time periods prior to the current Workflow year. If this is set to False, data cannot be imported to time periods prior to the current Workflow year and will display on the Import task during the Workflow.

Allows Loads After Workflow View Year

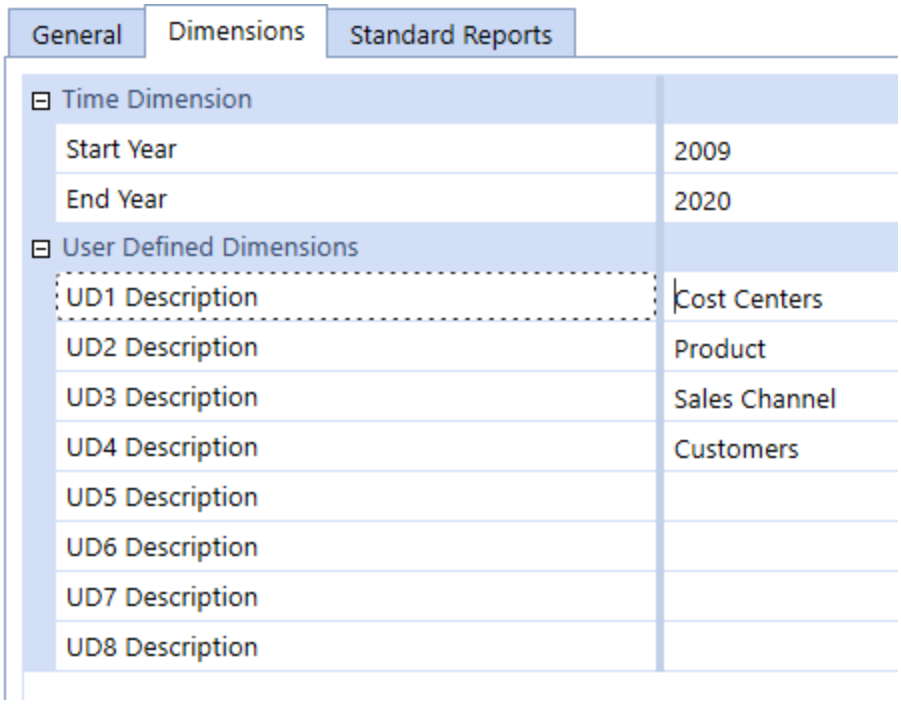
If set to True, this will enforce the current Workflow setting to allow data loading to time periods after the current Workflow year. If this is set to False, data cannot be imported to time periods after the current Workflow year and will display on the Import task during the Workflow.

Certification

Lock after Certify

If set to True, this will auto lock after certification in the Workflow.

Dimension Properties



Time Dimension

Start Year

The starting year of the application.

End Year

The ending year of the application.

User Defined Dimensions (Descriptions)

Applies a custom description to the eight User Defined dimensions. The description applies to the dimension type, not to each dimension and will be visible to the user in various interfaces in the OneStream Application. The descriptions display in the hover/tool-tips, pop-up windows and other dimension interfaces. These are viewable in:

- Point of View
- Dimension Library
- Cube View Member Filters
- Drill Down Dimension headers
- Excel Add-in / Spreadsheet
- Journals

UD1-8 Description
Enter a generic description that best describes the purpose of each User Defined dimension. See example above.

Standard Reports

These settings will be applied with auto-generating a report from a Cube View.

Logo

Height
Enter a numerical value to determine the Height of the report. (e.g. 105 pixels)

Bottom Margin
Enter a numerical value to determine the Bottom Margin size.

Title

Top Margin

Enter a numerical value to determine the Top Margin size.

Font Family

The font displayed in the Title of the report.

Font Size

Enter a numerical value to determine the size of the font.

Bold

If set to True, the Title will be bold in the report.

Italic

If set to True, the Title will be in italics in the report.

Text Color

Use the ellipsis icon to choose a text color for the report Title.

Header Labels

This is where the default Header Label properties are defined for all the reports in the application.

Top/Bottom Margin

Enter a numerical value to determine the Top/Bottom Margin size.

Font Family

The font displayed in the Header Labels of the report.

Font Size

Enter a numerical value to determine the size of the font.

Bold

If set to True, the Header Labels will be bold in the report.

Italic
if set to True, the Header Labels will be in italics in the Report.

Text Color
Use the ellipsis icon to choose a text color for the Header Labels.

Header Bar

Background Color
Use the ellipsis icon to choose a Header Background color.

Line Color
Use the ellipsis icon to choose a Header Line color.

Footer

Text
An open field to enter footer text.

Font Family
The font displayed in the Footer.

Font Size
Enter a numerical value to determine the size of the font in the Footer.

Show Line
If set to True, the report will show a line in the Footer.

Show Date
If set to True, the report will show the date in the Footer.

Show Page Numbers
If set to True, the report will show page numbers in the Footer.

Line Color
Use the ellipsis icon to choose the color of the line in the Footer.

Text Color
Use the ellipsis icon to choose a text color in the Footer.

Business Rules

Business Rules contain calculation logic configured to run against different parts of an application. These rules are compiled with VB.NET or C# code and are created within a rich integrated development environment, or IDE.

See the *API Overview Guide*.

There are several areas in the product using the exact same rule syntax and applying it to how data is imported, how the Cubes are calculated, and other operations. Once this syntax is understood, logic can be written.

Note:

There are three ways to write this calculation logic:

Business Rules
Business Rules are found under the Application Tab|Tools. There are nine types of Business Rules as shown below. These can be stored, secured, and then assigned to multiple areas of the product with the ability to re-use them.

Complex Expression
This logic can be created as a Business Rule or as a Complex Expression from within an application artifact such as a Data Source. The syntax is the same with the only difference being that a Business Rule can be shared across multiple application artifacts where a Complex Expression is contained within the artifact.

Member Formula

The same Business Rule syntax can be applied to Member Formulas as well. This logic stays with the Member and cannot be shared.

There are also three utility groups available when writing Business Rules:

BRAPI

BRAPI provides application programming interface to commonly used functions involving all areas of the product where a Business Rule can be used.

API

The more specific API provided as a Parameter to a Business Rule provides functions specific to the type of Business Rule being written. For example, when implementing a Business Rule for a finance-oriented task, API refers to the functions used for processing calculation logic and other capabilities related to processing a Cube's data and metadata.

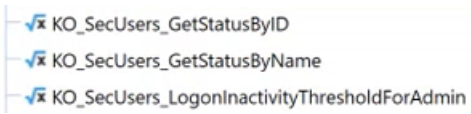
ARGS

An argument represents the value passed to a Business Rule when the procedure is called and the calling code supplies the arguments. For example, if a Parser Business Rule is assigned to the Account Dimension, args will supply the Account Dimension data as well as a set of functions available to use against that data. Different args will be provided depending on the type of Business Rule used.

Business Rule Encryption (Decryption)

There is also functionality to Encrypt and Decrypt Business Rules with password protection when writing and saving Business Rules.

The authorized user can Encrypt Business Rule by clicking the encrypt button.

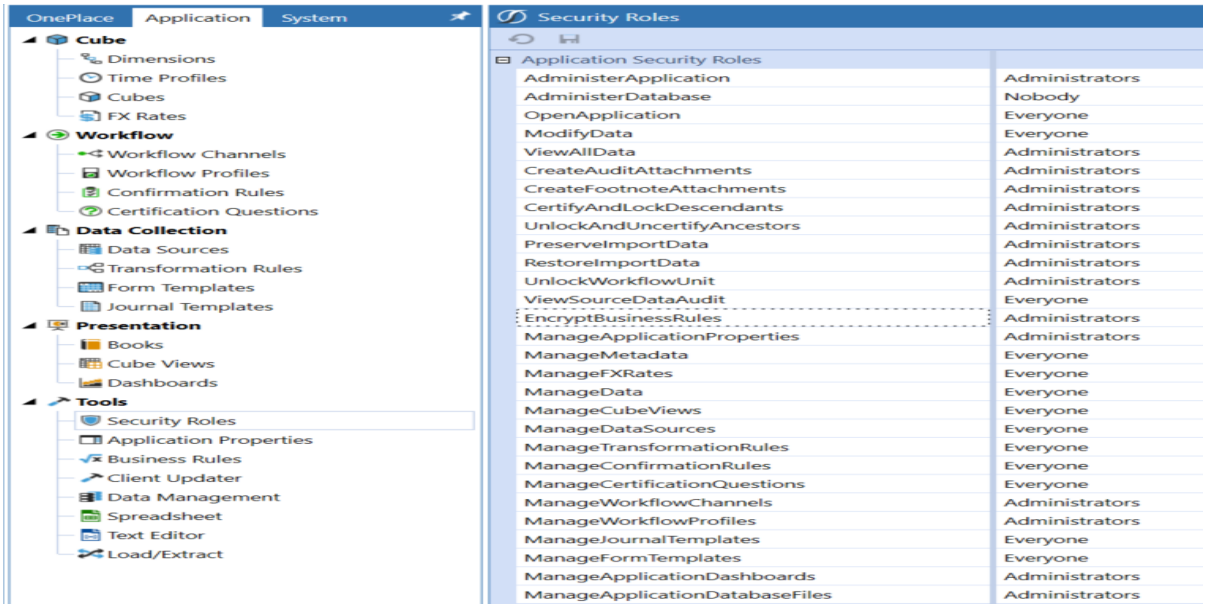


Once clicked, an Encrypt Business Rule dialog will display prompting the user for a password. The user will create and enter the password and then reenter the password in the box below and click OK. The system encrypts the Business Rule, displays “Business Rule Is Encrypted” message text in the editor and the editor is in read-only mode.

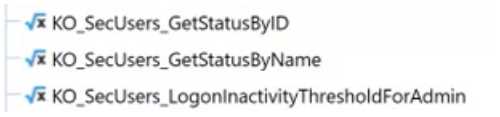
> **NOTE:** It is important to remember and record the password being used for each Business Rule being encrypted or the Business Rule will not be able to be decrypted for further changes without the assistance of OneStream Support.

To Decrypt a Business Rule, an authorized user can Decrypt Business Rule by clicking the Decrypt button . At this time, the Decrypt Business Rule dialog box will display prompting the user for a password. User will enter the password and click OK. The system will decrypt the Business Rule, display the Business Rule text editor and then return to read/write mode again.

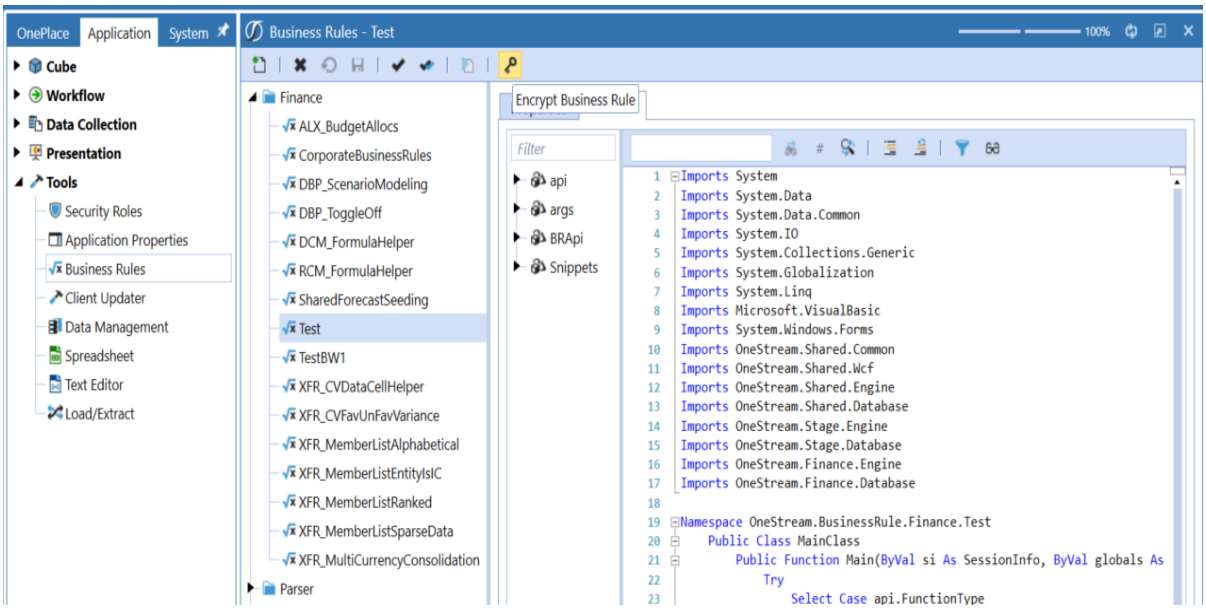
Set the proper access to Encrypt Business Rules to allow an authorized user to Encrypt and Decrypt a rule from the Business Rule screen in the Application tab, if the user is in the role.



Next advance to the Business Rules section (Application Tab>>Tools>>Business Rules) and Refresh the screen and the Encrypt Business Rule button will appear in the menu.

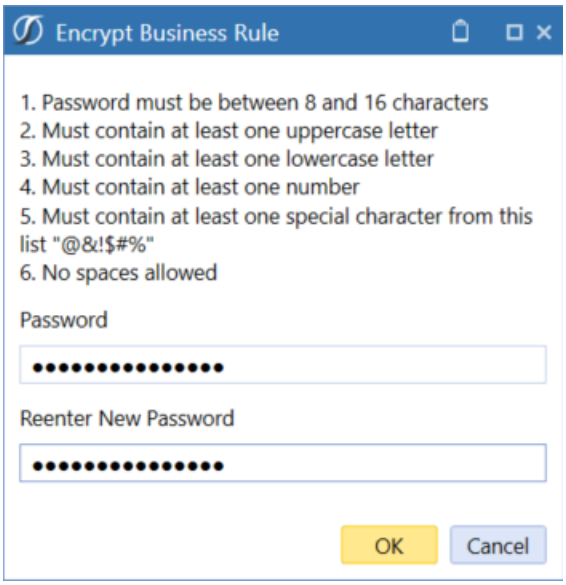


Select the Business Rule to be encrypted, then select the Encrypt Business Rule button.

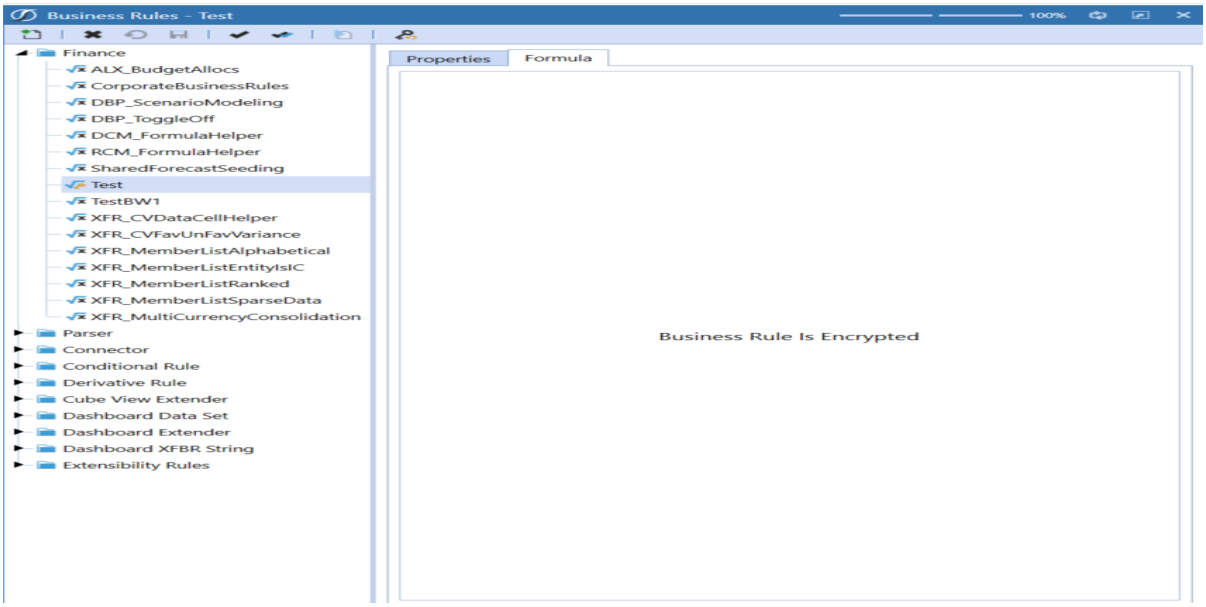


At this time, an Encrypt Business Rule dialog will display prompting the user for a password.

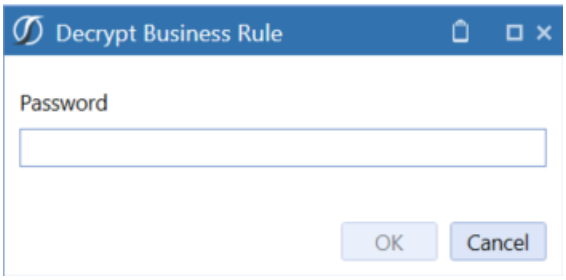
The user will create and enter the password, then Re-Enter the password in the box below and click OK.



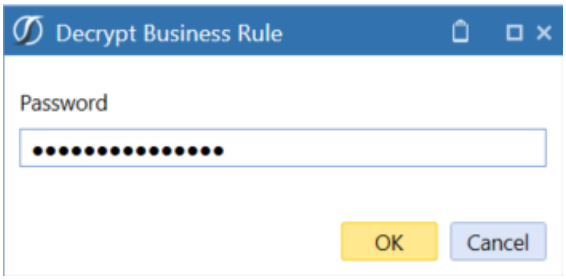
The system encrypts the Business Rule, displays “Business Rule Is Encrypted” message text in the editor and the editor is in read-only mode and the Decrypt Business Rule button is now displayed in the menu bar.



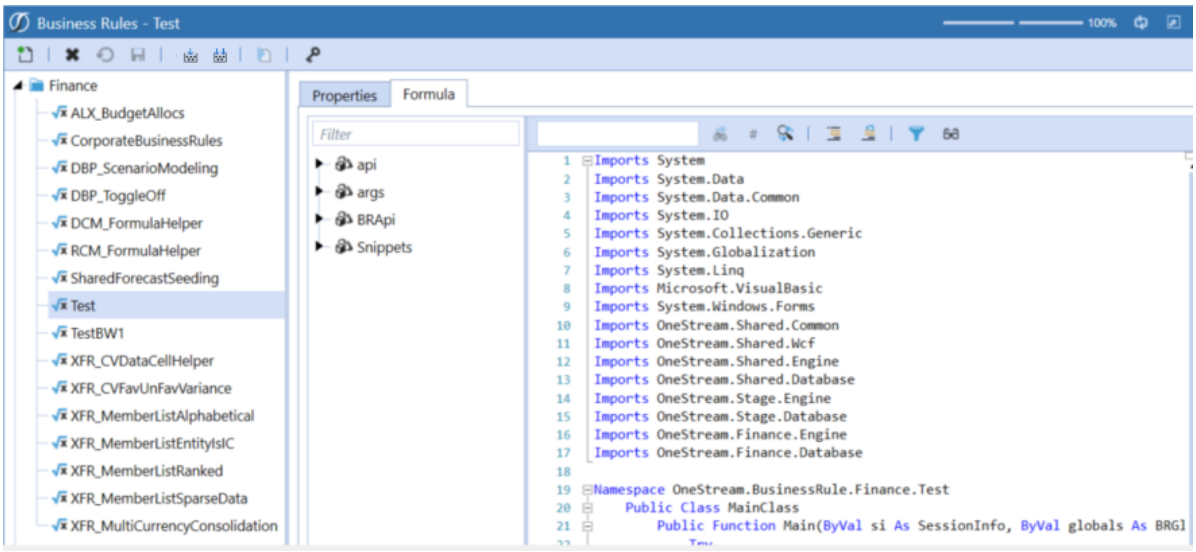
To Decrypt a Business Rule, an authorized user can Decrypt Business Rule by clicking the Decrypt button . At this time, the Decrypt Business Rule dialog box will display prompting the user for a password.



User will enter the password and click OK.



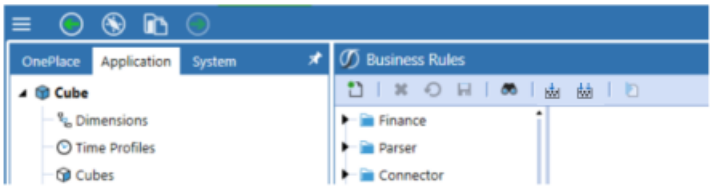
The system Decrypts the Business Rule displays the Business Rule text editor and the Business Rule is in read/write mode again.



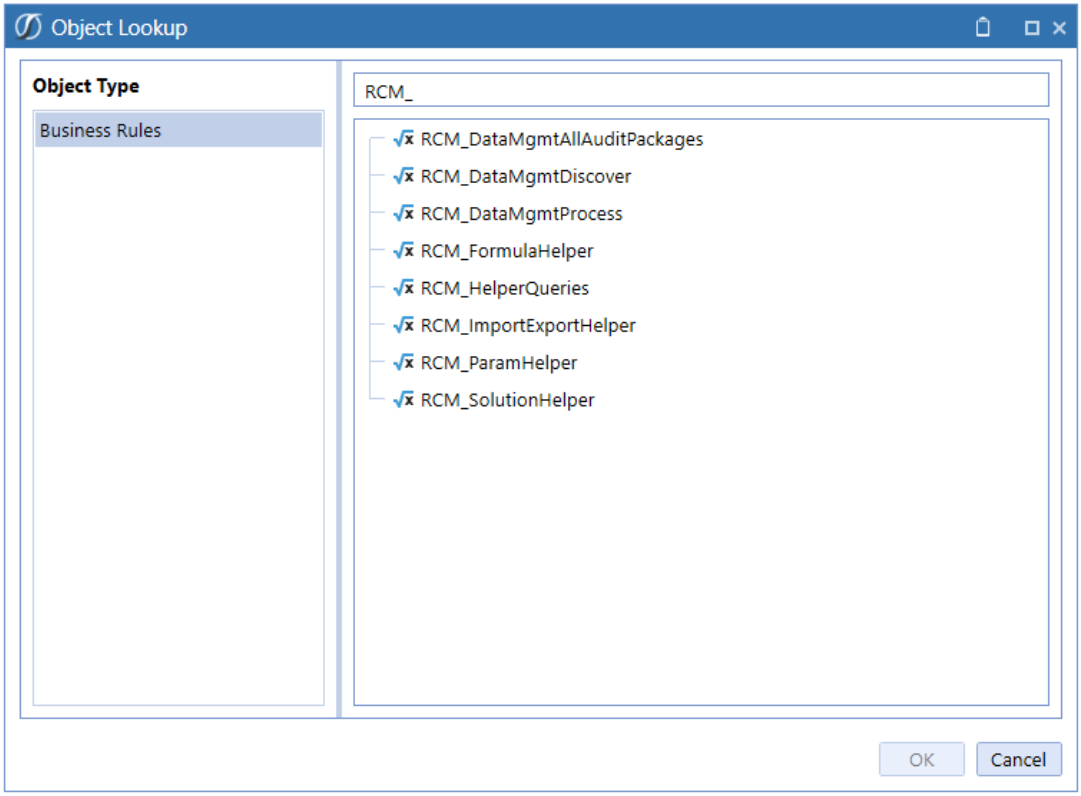
Business Rule Search

Find a business rule quickly by performing a search instead of scrolling through business rules.

1. Select the Application tab.
2. From Tools, select Business Rules.



3. Click Business Rule Search. The Object Lookup dialog box opens.
4. Start typing the start of the business rule name. All business rules that match your text display in the list.



5. Select the business rule from the list and click **OK**.

Business Rule Toolbar and Hotkeys

Compile Business Rule to Check Syntax

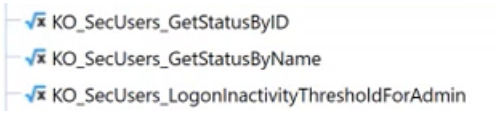
Use this to compile a selected Business Rule in order to check its syntax.

Compile All Business Rules and Formulas to Check Syntax

Use this to compile all Business Rules and Formulas in order to check syntax. This is most commonly used after installing a software upgrade where some Rule or Formula syntax may have changed. This feature is only available for application administrators.

Execute Extender

Use this to run the selected Extender Business Rule



Encrypt Business Rule

Use this to Encrypt the Business Rule formula. Clicking this button will be prompt the user to enter a password in the Encrypt Business Rule dialog box to complete the encryption process.

Decrypt Business Rule

Use this to Decrypt the Business Rule formula. Clicking this button will be prompt the user to enter a password in the Decrypt Business Rule dialog box to complete the encryption process.

Ctrl+M

Expand /Collapse all regions and methods. Click in the script after selecting the Business Rule in order to use the hotkey.

Business Rule Client Image Types

Client images are specified via a XFClientImageType Class in a Business Rule. These images can display on certain Dashboard Parameter Components such as Buttons, Maps, SQL Table Editors, Grid Views, etc.

The Image Types are as follows:

- StatusGrayBall
- StatusWhiteBall
- StatusOrangeBall
- StatusBlueBall
- StatusRedBall
- StatusLightGreenBall
- StatusGreenBall
- StatusGrayCheckMark
- StatusGreenCheckMark
- StatusLockedWithCheckMark
- StatusLockedWithFolder

Business Rule Properties

General

Name
The name of the Business Rule

Type

The type of Business Rule (see below for a detailed description of each Business Rule Type)

Contains Global Functions for Formulas

Set this to **True** to write a function in a **Business** Rule and then call the function from Member Formulas or other Business Rules. This is helpful when the same code must be copied to multiple Member Formulas and instead of using the same complicated code, a Public function with two lines of code written in a Business Rule can be called. Only use this setting for Business Rules referenced from Member Formulas because it does incur some overhead when the system needs to compile Member Formulas.

For more details on this feature, see About the Financial Model.

Referenced Assemblies

Enter a list of referenced assembly names separated by semi-colons.

Reference Another Business Rule

To reference another Business Rule, enter BR\ followed by the other Business Rule name.

> **NOTE:** A rule may reference either a VB.NET or C# rule.

BR\SharedFunctionsBR

Example:

```
Dim sharedFinanceBR As New
OneStream.BusinessRule.Finance.SharedFinanceFunctions.MainClass
Dim myResult As String = sharedFinanceBR.Test(si, api, args)
```

For more details on this feature, see Referencing a Business Rule from a Member Formula or Business Rule in About the Financial Model.

Reference a DLL File

DLL files can be stored in the File Share's Business Rule Assembly Folder, an Application Server Configuration File, or in a common Network Share Folder to which numerous Application Servers can reference. To reference a DLL File, enter XF\ followed by the DLL file name.

XF\ThirdPartyFunctions.dll

Otherwise, use no prefix and enter the full path and file name of any DLL file stored on the application server(s) file system.

Is Encrypted

This will be set to True if the Business Rule has been encrypted or False if it has not been encrypted.

Security

Access

Members of this group will have access to the Business Rule

Maintenance

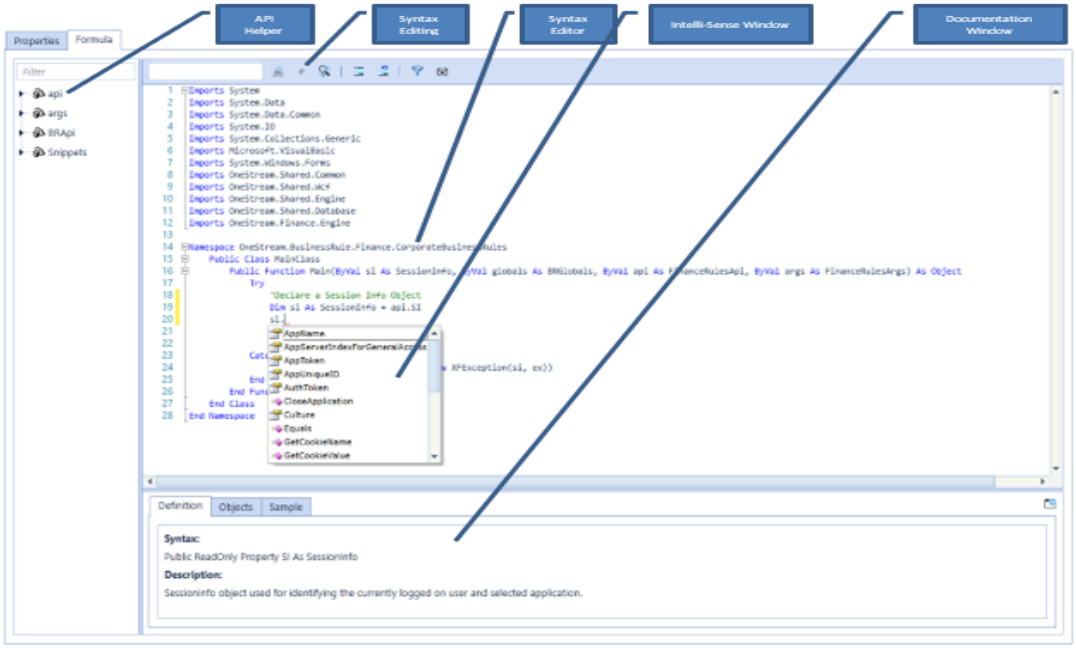
Members of this group have the authority to maintain the Business Rule

> **NOTE:** Click 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, click CTRL and Double Click. This will enter the correct name into the appropriate field.

Business Rules Editor Overview

The OneStream Business Rule editor is a powerful in-solution screen that provides integrated API context help, syntax editing with intelli-sense, and full outlining capabilities. The actual syntax content and Business Rule structure will be discussed at length in subsequent sections of this document.

The image below explains the major regions and elements of the Business Rule editor.



Business Rule Types

Finance Rule

Finance Business Rules are used to generate multi-Dimensional calculations. These Business Rules are written as Shared Business Rules and applied to a Cube or Member Formulas. Eight of these can be assigned to run at the Cube level during consolidations.

Example APIs

api.FunctionType
The expression used when special logic needs to be run and a certain process needs to be isolated.

Finance Function Type

Calculate
Additional logic during calculation of Entity, Consolidation Scenario and Time. This sets the value of one or more values (left side of Formula) equal to another (right side). It then executes a calculation for a specifically qualified Point of View. This is the most common function used.

Translate
Additional logic that uses custom translation.

FXRate
Custom logic used to determine Foreign Exchange rates for any intersection.

Consolidate Share
Additional logic used during the custom calculation of the Share Member.

Consolidate Elimination
Additional logic used during the custom calculation of the Elimination Member.

Calculation Drill Down Member Formula
Provides custom drill down results.

Conditional Input Rule

Conditional Input Rules make data cells read-only. While the settings for this can be done directly on the Cube, using a Conditional Input Business Rule offers more flexibility and still allows the use of the Cube settings. This rule can return the following: ConditionalInputResultType.Default, ConditionalInputResultType.NoInput, ConditionalInputResultType.NoInputAllowCellDetail, and ConditionalInputResultType.NoCellDetailAllowInput.

The following Business Rule example will make all cells for the Account 6000 read-only. This should be added to a Business Rule attached to a Cube.

```
Case Is = FinanceFunctionType.ConditionalInput
    If api.Pov.Account.Name.XFEqualsIgnoreCase("6000") Then
        Return ConditionalInputResultType.NoInput
    End If
    Return ConditionalInputResultType.Default
```

Confirmation Rule

Special logic that runs with Confirmation Rules.

Data Cell

Named GetDataCell calculations that can be reused such as a Better/Worse calculation in Cube Views.

Dynamic Calc Account

Special logic to use in Dynamic Calc members.

Member List and Member List Headers

A custom list of members for use in Cube Views and other areas. See Commonly Used Member Filter Functions in Cubes for more details on using custom lists in a Cube View.

Finance Business Rule Example

```
Namespace OneStream.BusinessRule.Finance.CorporateBusinessRules
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

				'Populate Headcount Flow
					If api.pov.Scenario.name = "Actual" then

				api.data.calculate("A#Headcount:F#BegBalHC=A#Headcount:F#None:T#POVPrior1")

				api.data.calculate("A#Headcount:F#EndBalHC=A#Headcount:F#None")

				api.data.calculate("A#Headcount:F#CheckHC=A#Headcount:F#BegBalHC+A#Headcount:
				F#TotalMovementHC-A#Headcount:F#EndBalHC")
					End If
					If api.pov.Scenario.name = "Variance" then
						api.data.Calculate("F#BudgetEBITDA:A#[EBITDAVar] =
				A#EBITDA:S#Budget:F#None")
						api.data.Calculate("F#ActualEBITDA:A#[EBITDAVar] =
				A#EBITDA:S#Actual:F#None")
						api.data.Calculate("F#[Bad Debt]:A#[EBITDAVar] =
				(A#54100:S#Budget:F#None-A#54100:S#Actual:F#None)")

				api.data.Calculate("F#VarianceCheckSum:A#[EBITDAVar]=F#BudgetEBITDA:
				A#[EBITDAVar]+F#VarianceEBITDA:A#[EBITDAVar]-F#ActualEBITDA:A#[EBITDAVar]")
					End If
			End Select

			Return Nothing
		Catch ex As Exception
			Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
		End Try
	End Function
	End Class
End Namespace
```

Parser Rule

Parser Business Rules are used to evaluate and/or modify field values being processed by the Stage Parser Engine as it reads source data. These Business Rules are written as Shared Business Rules or Logical Expressions and applied to a Data Source Dimension.

Example API

args.Line

This will return the entire record being processed from the Data Source.

args.Value

This will return the Dimension the Business Rule is assigned to in the Data Source.

Common Usage

- Custom parsing logic
- Field value concatenation
- Field value bypassing
- Evaluate field other than current field being parsed

Parser Business Rule Example

```
Namespace OneStream.BusinessRule.Parser.XFR_ParseDelimitedTextAndGetField
	Public Class MainClass
		'------------------------------------------------------------------------------------------------------------
		'Reference Code:	XFR_ParseDelimitedTextAndGetField
		'Description:		Parse a delimited string and return a selected field from the delimited list.
		'Usage:		Parser Business Rule intended to be used on any field within a Data Source.
		'------------------------------------------------------------------------------------------------------------
		Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As ParserDimension, ByVal args As
		ParserArgs) As Object
			Try
				'Parse the line
				Dim fields as List(Of String) = StringHelper.SplitString(args.Line, ",",
				StageConstants.ParserDefaults.DefaultQuoteCharacter)
				If fields.Count >= 2 Then
					'Return the 2nd field (Zero based index)
					Return fields(1)
				Else
					'Line invalid, just return incoming value and let parser engine handle the evaluation
					Return args.Value
				End If
			Catch ex As Exception
				Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
			End Try
		End Function
	End Class
End Namespace
```

Connector Rule

Connector Business Rules are used to communicate with, collect data from, and drill back to external systems. These Business Rules are written as Shared Business Rules and applied to a Data Source.

See Connectors in Data Collection Guides for more information on using Connectors.

Example API

args.ActionType

This will return one of the four available Connector action types.

ConnectorActionTypes.GetFieldList
This will run the SQL query to return the available field list to the Data Source for Dimension assignment.

ConnectorActionTypes.GetData
This will run the SQL query to retrieve the source data and return it to the Stage based on the Data Source Dimension assignment.

ConnectorActionTypes.GetDrillBackTypes
This will return a list of the specified drill back types. (e.g., File or Data Grid)

ConnectorActionTypes.GetDrillBack
This will run the required SQL query against the source system and will provide greater detail than what was originally imported.

Connector Business Rule Example

Namespace OneStream.BusinessRule.Connector.RevenueMgmtHouston

```
Public Class MainClass
    Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As
    ConnectorArgs) As Object
        Try
            'Get the query information
            Dim connectionString As String = GetConnectionString(si, globals, api)

            'Get the Field name list or load the data
            Select Case args.ActionType
                Case Is = ConnectorActionTypes.GetFieldList
                    'Return Field Name List
                    Dim fieldListSQL As String = GetFieldListSQL(si, globals, api)
                    Return api.Parser.GetFieldNameListForSQLQuery(si, DbProviderType.OLEDB,
                    connectionString, true, fieldListSQL, false)

                Case Is = ConnectorActionTypes.GetData
                    'Process Data
                    Dim sourceDataSQL As String = GetSourceDataSQL(si, globals, api)
                    api.Parser.ProcessSQLQuery(si, DbProviderType.OLEDB, connectionString,
                    true, sourceDataSQL, false, api.ProcessInfo)
                    Return Nothing

                Case Is = ConnectorActionTypes.GetDrillBackTypes
                    'Return the list of Drill Types (Options) to present to the end user
                    Return  Me.GetDrillBackTypeList(si, globals, api, args)

                Case Is = ConnectorActionTypes.GetDrillBack
                    'Process the specific Drill-Back type
                    Return Me.GetDrillBack(si, globals, api, args, args.DrillBackType.DisplayType,
                    connectionString)
            End Select

        Catch ex As Exception
            Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
        End Try
    End Function

    'Create a Connection string to the External Database
    Private Function GetConnectionString(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer) As
    String
        Try

            'Named External Connection In App Server Config
            '-----------------------------------------------------------------
            Return "RevenueMgmtSystem"

        Catch ex As Exception
            Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
        End Try
    End Function

    'Create the field list SQL Statement
    Private Function GetFieldListSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer) As String
        Try
            'Create the SQL Statement
            Dim sql As New Text.StringBuilder

            sql.Append("SELECT Top(1)")
            sql.Append("TransID, PlantCode, CustId, CustName, InvNo, InvYear, InvMonth, InvDesc, GLAccount,
            WorkDay, ProdModel, BomCode, UnitPrice, Units, Amount, DestinationCode ")

            sql.Append("FROM InvoiceDocumentDetail ")
```

```
                Return sql.ToString

            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
End Function

'Create the data load SQL Statement
Private Function GetSourceDataSQL(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer) As
String
            Try
                'Create the SQL Statement
                Dim statement As New Text.StringBuilder
                Dim selectClause As New Text.StringBuilder
                Dim fromClause as New Text.StringBuilder
                Dim whereClause as New Text.StringBuilder
                Dim orderByClause as New Text.StringBuilder

                selectClause.Append("SELECT ")
                selectClause.Append("TransID, PlantCode, CustId, CustName, InvNo, InvYear, InvMonth, InvDesc,
                GLAccount, WorkDay, ProdModel, BomCode, UnitPrice, Units, Amount, DestinationCode ")

                fromClause.Append("FROM InvoiceDocumentDetail ")

                whereClause.Append("WHERE ")
                'Get the YEAR from the current XF Workflow Unit TimeKey
                whereClause.Append("(")
                whereClause.Append("invYear = " &
                TimeDimHelper.GetYearFromId(api.WorkflowUnitPk.TimeKey).ToString)
                whereClause.Append(")")

                'Get the MONTH from the current XF Workflow Unit TimeKey
                whereClause.Append(" And ")
                whereClause.Append("(")
                whereClause.Append("invMonth = 'M" &
                TimeDimHelper.GetSubComponentsFromId(api.WorkflowUnitPk.TimeKey).Month.ToString & "'")
                whereClause.Append(") ")

                'Select Houston Plant Codes
                whereClause.Append(" And ")
                whereClause.Append("(")
                whereClause.Append("PlantCode IN('H200','H210')")
                whereClause.Append(") ")

                orderByClause.Append("ORDER BY ")
                orderByClause.Append("PlantCode, CustId, WorkDay, ProdModel, DestinationCode")


                'Create the full SQL Statement
                statement.Append(selectClause.ToString)
                statement.Append(fromClause.ToString)
                statement.Append(whereClause.ToString)
                statement.Append(orderByClause.ToString)

                Return statement.ToString

            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
End Function

'Create the drill back options list
Private Function GetDrillBackTypeList(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal
args As ConnectorArgs) As List(Of DrillBackTypeInfo)
            Try
                'Create the SQL Statement
```

Conditional Rule

Conditional Rules (mapping) are used to conditionally evaluate mapping criteria during the data transformation process. These Business Rules are written as Shared Business Rules or Logical Expressions and applied to a Transformation Rule definition.

They are only applicable to Transformation Rules with the type of Composite, Range, List, or Mask either as a Business Rule or Complex Expression.

Example API's

args.GetSource
This will return the source value for the specified Dimension.

args.GetTarget
This will return the mapped or transformed value for the specified Dimension.

args.OutputValue
This will return the originally mapped target value from the Transformation Rules.

Conditional Business Rule Example 1

Conditional Rule for Dimension Member mapping based on source values. Note that if the Business Rule does not call args.OutputValue, the target field can be empty and will not be considered.

```
Namespace OneStream.BusinessRule.ConditionalRule.XFR_Conditional_Source
	Public Class MainClass
		'------------------------------------------------------------------------------------------------------------
		'Reference Code:	XFR_Conditional_Source
		'Description:		Demonstrates how to map a flow Dimension based on the source account.
		'Usage:			Used for conditional flow Member mapping based on source account.
		'------------------------------------------------------------------------------------------------------------
		Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As
		ConditionalRuleArgs) As Object
			Try
				'Grab the source account value
				Dim account As String = args.GetSource("A#")

				'Assign the flow Member based on the source account
				If account.StartsWith("1190") Then
					Return "Additions"
				Else
					'Just return value assigned to the Transformation Rule
					Return args.OutputValue
				End If

				Return Nothing

			Catch ex As Exception
				Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
			End Try
		End Function
	End Class
End Namespace
```

Conditional Business Rule Example 2

Conditional Rule for Dimension Member mapping based on target values.

```
Namespace OneStream.BusinessRule.ConditionalRule.XFR_Conditional_Target
	Public Class MainClass
		'------------------------------------------------------------------------------------------------------------
		'Reference Code:	XFR_Conditional_Target
		'Description:		Demonstrates how to map a flow Dimension based on the target account.
		'Usage:			Used for conditional flow Member mapping based on target account.
		'------------------------------------------------------------------------------------------------------------
		Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As
		ConditionalRuleArgs) As Object
			Try

				'Grab the target account value
				Dim account As String = args.GetTarget("A#")

				'Assign the flow Member based on the first character of the target account
				Select Case Left(account,1)
					Case "1","2","3"
						Return "End_Inp"
					Case Else
						Return "None"
				End Select

			Catch ex As Exception
				Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
			End Try
		End Function
	End Class
End Namespace
```

Derivative Rule

Derivative Rules (derive data prior to mapping) are used to evaluate and/or calculate values during the data derivation process. These Business Rules are written as Shared Business Rules or Logical Expressions and applied to a Derivative Rule definition.

They are only applicable to Transformation Rules with the type of Derivative either as a Business Rule or Complex Expression.

Common Usage

- Calculate mathematical expressions
- Lookup value from transformation cache for use in calculations

- Lookup value from Cube for use in calculations
- Source system check rule logic (validation rules on source data)

Derivative Business Rule Example

```
Namespace OneStream.BusinessRule.DerivativeRule.XFR_ExecuteStageCacheQuery
	Public Class MainClass
		'------------------------------------------------------------------------------------------------------------
		'Reference Code:	XFR_ExecuteStageCacheQuery
		'
		'Description:		Retrieves a value from the transformation cache calculated by another derivative rule
		'			and uses the value as an input to calculate the ratio of one account to another.
		'
		'Usage:			Derivative Business Rule intended to demonstrate how to retrieve a value from the
		'			transformation cache and use it as input to a calculation.
		'------------------------------------------------------------------------------------------------------------
		Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Transformer, ByVal args As
		DerivativeRuleArgs) As Object
			Try
				'Get Cash Value by querying the cache
				Dim valueCash as Decimal = Decimal.Zero
				Using dtCash as DataTable = api.QueryDataCache(si, "Ac = 'Cash'", False, True)
					If Not dtCash Is Nothing Then
						For Each row As DataRow in dtCash.Rows
							valueCash = row("Am")
							Exit For
						Next
					End If
				End Using

				'The current value is AR
				Dim valueAR as Decimal = args.ColumnValue

				'Return the ratio of AR/Cash
				If (valueAR <> Decimal.Zero) and (valueCash <> Decimal.Zero) Then
					Return (valueAR / valueCash)
				Else
					'Just return the value from the Stage
					Return args.ColumnValue

				End If
				Return Nothing

			Catch ex As Exception
				Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
			End Try
		End Function
	End Class
End Namespace
```

Cube View Extender

Cube View Extender Rules are used to apply advanced Cube View formatting to any Cube View while in Report view.

> **NOTE:** These rules do not apply to how a Cube View looks like in the Data Explorer Grid view.

The Extender Rule is used in conjunction with the Execute Cube View Extender Business Rule setting on the Cube View. See Cube View Extender: Advanced Cube View Formatting in Implementing Security for examples on how to use this rule. See the *API Overview Guide*.

Common Usage

The following are key uses for Cube View Extender Business Rules in formatting Reports.

- Alter Headers and Footers
    - Display different logos on select reports based on conditional logic or security and manage their placement and size
    - Customize the page number in the header or footer
      Page numbers can be on the top or bottom row of a report and the horizontal position can be specified for rows. This only applies to the top or bottom rows.
    - Format individual header and footer fields
    - Control the Left, Right, Center Subtitle widths
    - Control the font size of Title and Subtitles
    - Customize the date display
- Alter Column and Row Headers

    - Customize bottom text alignment
    - Control Word Wrap
- Apply Conditional Formatting to Data Cells
    - Format cells based on their contents
    - Change the text color of a value in order to effectively hide the result

Dashboard DataSet

DashboardDataSet Rules are used to create programmatic query results. This rule type combines multiple types of data into a single result set using the full syntax capability of VB.Net. These Business Rules are written as Shared Business Rules and applied to Dashboard Data Adapters or Dashboard Parameters.

Common Usage

- Combine different types of data for a report
- Build programmatic data queries (e.g., analytic plus sql)
- Conditionally build data query reports
- Conditionally build data query for Parameters
- Create geographical data to display via a Map Parameter Component in a Dashboard
- Create a data series to display via a Chart (Advanced) Parameter Component in a Dashboard

Dashboard Extender

DashboardExtender Rules are used to perform a variety of tasks associated with custom Dashboards and OneStream Solutions. These Business Rules can be thought of as multi-purpose rules and make up the majority of the code written in a OneStream Solution. In addition, they are written as Shared Business Rules and applied to Application Dashboard Parameter Components (Buttons, Combo Boxes, etc.).

Common Usage

- Execute task when the user clicks a button
- Perform a task and show a message to the user
- Perform a custom calculation
- Upload a file from the end user's machine
- Automate a Workflow
- Build a custom Workflow
- Create custom data tables
- Include Page State to store parameters and values about a specific Dashboard page instance

Dashboard XFBRString

Dashboard XFBRString Rules are used to process conditional Dashboard Parameters. These rules inspect and alter a Dashboard Parameter value using the full syntax capabilities of VB.Net or C#. Dashboard String Functions are written as Shared Business Rules and called by using a XFBR (BusinessRuleName, FunctionName, UserParam=[UserValue]) specification anywhere a standard Dashboard Parameter is used. After a Dashboard BRString rule is created, create a Dashboard Component to call the BRString using the following script.

```
BRString(brRuleName, funcNameWithinBRRule, optionalName1 = var1, optionalName2 = var2)
```

Enter the Business Rule as a Parameter using |!BRString!|

The return value from the Business Rule will be used in the Dashboard Component.

> **NOTE:** This Business Rule can be applied to any Dashboard or Cube View property where a Parameter is used.

Extensibility Rule

Extensibility Rules have these two types: Extender and Event Handlers. Extender Rules are the most generalized type of Business Rule in the OneStream platform. Use these to write a simple utility function or a specific helper function called as part of a Data Management Job. Event Handlers are exclusively called before or after a certain operation occurs within the system.

> **IMPORTANT:** Full Administrator rights are required to edit Extensibility Rules.

Extensibility Business Rule Example

Event Handler that sends an email notification after a ProcessCube event.

```
Namespace OneStream.BusinessRule.DataQualityEventHandler.DataQualityEventHandler
	Public Class MainClass
		'----------------------------------------------------------------------------------------------------
		'Reference Code:	DataQualityEventHandler
		'
		'Description:		Event handler method that provides an opportunity to supplement a normal data
		'			quality action with your own custom functionality.
		'				(Example: email after ProcessCube or publish report to SharePoint after
		'				failed Confirmation).
		'
		'Usage:			Executes when a Data Quality action is run and fires this Business Rule.  If you have
		'			written code in that handles the specified event operation the code will be executed.
		'----------------------------------------------------------------------------------------------------
		Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Object, ByVal args As
		DataQualityEventHandlerArgs) As Object
			Try
				'Define a switch to control event processing, since many of these are reference examples we do not
				want them to run all the time
				Dim processEvents as Boolean = False

				'Set the default return values
				Dim returnValue As Object = args.DefaultReturnValue
				args.UseReturnValueFromBusinessRule = False
				args.Cancel = False

				'Evaluate the operation type in order to determine which subroutine to process
				Select Case args.OperationName
					Case Is = BREventOperationType.DataQuality.ProcessCube.NoCalculate
						'Execute a Data Management job after process Cube runs
						Me.XFR_HandleProcessCubeNoCalculate(si, globals, api, args)

					Case Is = BREventOperationType.DataQuality.Certify.FinalizeSetCertifyState
						'Send an email after a Workflow Profile executes its certification
						Me.XFR_HandleFinalizeSetCertifyState(si, globals, api, args)

				End Select

				Return returnValue

			Catch ex As Exception
				Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
			End Try
		End Function

#Region "ProcessCube.NoCalculate Helpers"

		Private Sub XFR_HandleProcessCubeNoCalculate(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Object,
		ByVal args As DataQualityEventHandlerArgs)
			'----------------------------------------------------------------------------------------------------
			'Reference Code:	XFR_HandleProcessCubeNoCalculate
			'
			'Description:		Run a Data Management Sequence after the Workflow process Cube task is
			'			run.
			'				Note: the Data Management Sequence name is assigned to a
			'				Workflow Profile CalcDef filter field
			'				so this event does not have to be modified, the user can simply
			'				edit the CalcDef grid
			'				for a Workflow Profile and this Business Rule will execute the
			'				specified sequence.
			'
			'Usage:			Used to supplement the standard "ProcessCube" functionality associated
```

```
                                        with a Workflow Profile by allowing a Data Management Sequence to be
                                        executed for the Workflow Profile as well.
                '-----------------------------------------------------------------------------------------------------
                Try
                        'Get the DataUnitInfo from the Event arguments so that we can get the name of the
                        DataManagement sequence to process.
                        Dim calcInfo As DataUnitInfo = DirectCast(args.Inputs(2), DataUnitInfo)
                        If Not calcInfo Is Nothing Then
                                'Make sure that a Sequence name as assigned to the filter value of the Calc Definition of
                                the executing Workflow Profile
                                If calcInfo.FilterValue <> String.Empty Then

                                        'Now, execute the DataMgmt Sequence that was specified in the FilterValue
                                        (In a background thread)
                                        BRApi.Utilities.StartDataMgmtSequence(si, calcInfo.FilterValue, Nothing)

                                End If
                        End If

                Catch ex As Exception
                        Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
                End Try
        End Sub

#End Region

#Region "FinalizeSetCertifyState Helpers"
        'Apply tag this to Text1 of users that should be notified when a child Workflow certifies
        Public const NotifyCertifyCompleted as String = "#NCC"

        Private Sub XFR_HandleFinalizeSetCertifyState(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Object,
        ByVal args As DataQualityEventHandlerArgs)
                '-----------------------------------------------------------------------------------------------------
                'Reference Code:        XFR_HandleFinalizeSetCertifyState
                '
                'Description:                   Send an email to users assigned the Workflow execution group of the Parent
                                                Workflow Profile of the Workflow Profile that just certified.
                '
                'Usage:                         Used to provide email notification when a dependent Workflow Profile
                                                certifies.
                '-----------------------------------------------------------------------------------------------------
                Try

                        'Check the before / after flag, we want to handle the AFTER event
                        If (args.IsBeforeEvent = False)  Then

                                'Check to see if the profile was certified, add send an email all Parents of the certifying
                                profile
                                Dim certifyInstance as CertifyInstanceInfo = DirectCast(args.Inputs(2),
                                CertifyInstanceInfo)
                                If Not certifyInstance is Nothing Then
                                        If certifyInstance.IsCertified(si) Then

                                                'Get the list of users assigned to the Parent of the Workflow
                                                profile so that we know who should get an email
                                                Dim notificationUsers as List(Of UserInfo) =
                                                BRApi.Workflow.General.GetUsersInWorkflowGroupForParents
                                                (si, certifyInstance.Workflow.WfUnitPk.CreateWorkflowUnit
                                                ClusterPk(), SharedConstants.WorkflowProfileAttributeIndexes.
                                                WorkflowExecutionGroup, True, True, NotifyCertifyCompleted)
                                                For Each userSummary As UserInfo in notificationUsers
                                                        'Send the email
                                                        me.CreateMessageAndSendMail(si, certifyInstance,
                                                        userSummary)
                                                Next
                                        End If
```

```
                End if

            End if

        Catch ex As Exception
            Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
        End Try
End Sub

Private Sub CreateMessageAndSendMail(ByVal si As SessionInfo, certInstance as CertifyInstanceInfo, userSummary as
UserInfo)
        '---------------------------------------------------------------------------------------------------
        'Reference Code:    CreateMessageAndSendMail
        '
        'Description:               Create the email message body and execute the "Send Mail" function.
        '---------------------------------------------------------------------------------------------------
        Try
            Using dbConnApp as DbConnInfo = BRApi.Database.CreateApplicationDbConnInfo(si)

                Dim emailInfo as new Text.StringBuilder
                emailInfo.AppendLine("To: " & userSummary.User.Email)

                'Create the message title
                Dim messageTitle as new Text.StringBuilder
                messageTitle.Append("OneStream Workflow ['')
                messageTitle.Append(certInstance.WorkflowProfileName)
                messageTitle.Append(", ")
                messageTitle.Append(ScenarioDimHelper.GetNameFromId(dbConnApp,
                certInstance.Workflow.WfUnitPk.ScenarioKey))
                messageTitle.Append(", ")

                messageTitle.Append(TimeDimHelper.GetNameFromId
                (certInstance.Workflow.WfUnitPk.TimeKey))
                messageTitle.Append("] is ready for you to review.")
                messageTitle.AppendLine("")

                'Create the message body
                Dim messageBody as new Text.StringBuilder
                messageBody.AppendLine("Certification Information:")
                messageBody.AppendLine("-----------------------------------------------------------")

                messageBody.AppendLine("Profile Name..... " & certInstance.WorkflowProfileName)

                Dim signOff as CertifySignOffInstanceInfo = Nothing
                If certInstance.SignOffsInDescendingTimestampOrder.Count > 0 Then
                    signOff = certInstance.SignOffsInDescendingTimestampOrder(0)
                    messageBody.AppendLine("User Name..... " & signOff.UserName)

                    messageBody.AppendLine("Time Completed..... " &
                    signOff.TimeStamp.ToLocalTime & " Local Time")

                    messageBody.AppendLine("Comments..... " & signOff.Comments)

                End If
                messageBody.AppendLine("")

                messageBody.AppendLine("Dependent Information:")
                messageBody.AppendLine("-----------------------------------------------------------")

                For Each relativeInfo as CertifyRelativeInfo in certInstance.Dependants
                    messageBody.AppendLine(relativeInfo.WorkflowProfile.Name & "  (" &
                    relativeInfo.Workflow.GetOverallStatusText("Not Started", "Completed",
                    "Locked", "Running", "") & ")")
                Next
                messageBody.AppendLine("")
```

```
                                    'Test the email by writing a log message
                                    ErrorHandler.LogMessage(si, emailInfo.ToString & vbcrlf &  messageTitle.ToString &
                                    vbcrlf &  messageBody.ToString)

                                    'Send an email message
                                    'Me.SendMail(si, userSummary.User.Email, messageTitle.ToString,
                                    messageBody.ToString)

                            End Using

                    Catch ex As Exception
                            Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
                    End Try
            End Sub

            Private Sub SendMail(ByVal si As SessionInfo, ByVal userEmailAddress as String, ByVal subject as String, ByVal
            messageBody as String)
                    '---------------------------------------------------------------------------------------------------
                    'Reference Code:        SendMail
                    '
                    'Description:                    Create the mail connection and send.
                    '---------------------------------------------------------------------------------------------------

                    Try
                    'Prepare the message
                    Dim emailConnectionName As String = "Office365Mail"
                    Dim toEmail As New List (Of String)
                      toEmail.Add("userEmailAddress")
                    Dim subject As String = "Subject"
                    Dim body As String = "Body"
                    Dim attachments As New List (Of String)
                      attachments.Add ("FileLocationString")

                    'Send the Message
                    BRApi.Utilities.SendMail(si, emailConnectionName, toEmail, subject, body, attachments)

                    Catch ex As Exception
                            Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
                    End Try
            End Sub

#End Region

        End Class
End Namespace
```

Extender

This can be used to automate custom tasks like running external scripts, backups and report publishing.

WCF Event Handler

This allows direct interaction with the Microsoft Windows Communication Foundation which means it listens to communication between the client and the web server. The rule will intercept the communication, analyze it, and if certain criteria is met, it will run its logic. This is quite flexible and has a variety of uses such as creating, reading, deleting, and updating different types of objects in the system for users in a group or Transformation Rule changes. For example, a rule can be created to e-mail an auditor about every metadata change as it happens.

Available operations

Transformation Event Handler

This can be run at various points from Import through Load. Available operations.

- StartParseAndTransForm
- InitializeTransFormer
- ParseSourceData
- LoadDataCacheFromDB
- ProcessDerivativeRules
- ProcessTransformationRules
- DeleteData
- DeleteRuleHistory
- WriteTransFormedData
- SummarizeTransFormedData
- CreateRuleHistory
- EndParseAndTransForm
- FinalizeParseAndTransForm

- StartRetransForm
- EndRetransForm
- FinalizeRetransForm
- StartClearData
- EndClearData
- FinalizeClearData
- StartValidateTransForm
- ValidateDimension
- EndValidateTransForm
- FinalizeValidateTransForm
- StartValidateIntersect
- EndValidateIntersect
- FinalizeValidateIntersect
- LoadIntersect
- StartLoadIntersect
- EndLoadIntersect
- FinalizeLoadIntersect

Journals Event Handler

This can be run before, during, or after a Journal operation such as Submission, Approval, or Post. Available operations:

- SubmitJournal
- ApproveJournal
- RejectJournal
- PostJournal
- UnpostJournal
- StartUpdateJournalWorkflow
- EndUpdateJournalWorkflow
- FinalizeUpdateJournalWorkflow

Save Data Event Handler

This is run in order to track all save events in an application.

Forms Event Handler

This can be run before, during, or after an operation such as Form Save. Available operations:

- SaveForm
- CompleteForm
- RevertForm
- StartUpdateFormWorkflow
- EndUpdateFormWorkflow
- FinalizeUpdateFormWorkflow

Data Quality Event Handler

This can be run before, during, or after data quality events like Confirmation and Certification.

Available operations:

- StartProcessCube
- Calculate
- Translate
- Consolidate
- EndProcessCube
- FinalizeProcessCube
- PrepareICMatch
- StartICMatch
- PrepareICMatchData
- EndICMatch
- StartConfirm
- EndConfirm
- FinalizeConfirm
- SaveQuestionResponse
- StartSetQuestionairreState
- SaveQuestionairreState

- EndSetQuestionairreState
- StartSetCertifyState
- SaveCertifyState
- EndSetCertifyState
- FinalizeSetCertifyState

Data Management Event Handler

This can be run before or after a Data Management Sequence or Step runs. Available operations:

- StartSequence
- ExecuteStep
- EndSequence

Workflow Event Handler

This can be run before or after a Workflow execution step. Available operations:

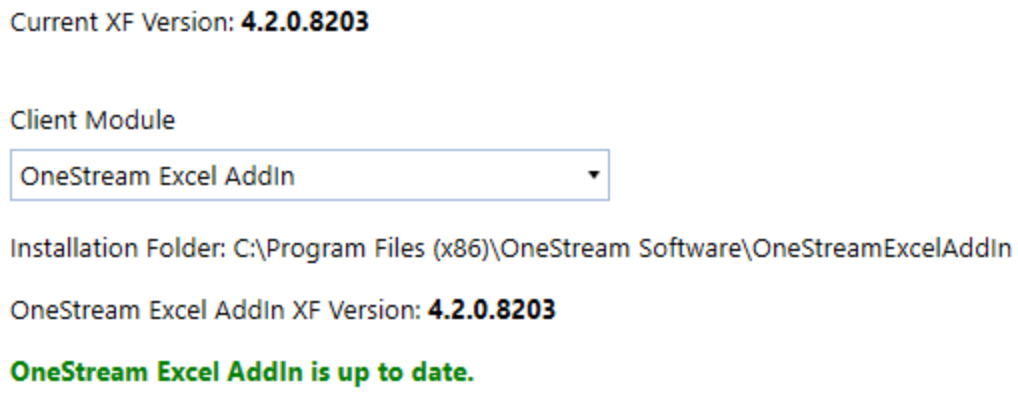
- UpdateWorkflowStatus
  WorkflowLock
  WorkflowUnlock

Client Updater

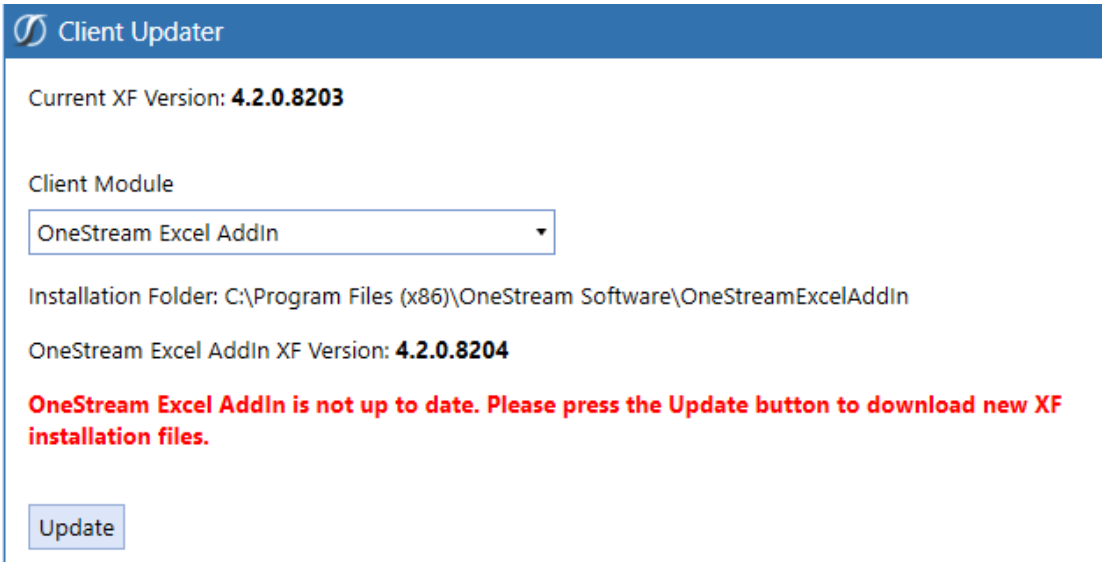
The Client Updater retrieves updated software from the OneStream server for the Excel Add-In client program when the version being used does not match the version found on the server being connected. To update, the user needs to be able to write to the installation folder. From this page, the user can first save work and then restart using elevated Windows privileges using the Restart OneStream as Windows Administrator button.

Information about the current version displays at the top of the window, as well as the module being reviewed, the location of the installation folder and the version status all appear within this window.

Example of versions that match:



Example of versions that do not match:



Client Module

Use this field to select which application (OneStream Excel Add-In) to compare to the current version of each that are currently installed on the user's desktop. Click the selection arrow and select the appropriate application. If the versions do not match, click **Update** and then **OK**.

> **NOTE:** Close any open versions of Excel before clicking OK to proceed.

> **NOTE:** A backup folder with files for the outdated version is automatically created and saved as part of the update process. It can be found in the same location as the newly updated version folder.

If this functionality has been disabled, the following message appears when trying to update those applications. "The Client Updater has been disabled by your System Administrator. Please use OneStream's full client installation program, or see your System Administrator."

Data Management

The Data Management module allows you to copy data or clear data for a Cube, Scenario, Entity, and Time. In order to do this, each Data Management Group must contain a Sequence and a Step. The Steps are created and specifically defined by Cube, Scenario, Entity, and Time. Once the Step is defined and saved, it can be assigned to a Sequence. Once these Groups are created, they are organized into Data Management Profiles.

Search

1. Click the binoculars to open the Select Data Management Object dialog box.
2. Select the Object type from the drop-down: All Items, Sequence, or Step.
3. Enter as much information as you have and click Search.
4. The search results show.
5. If you want to view the results by hierarchy, click View in Hierarchy.

Data Management

A Data Management Group allows you to create different groups each containing Sequences and Steps. A Data Management Group can be assigned to multiple Data Management Profiles.

Click to create a new Data Management Group.

General

Name
The name of the new data group.

Description
A description of the new data group.

Rename, Copy, and Paste

Existing Sequences and Steps can be renamed, copied, and pasted using the toolbar. Data Management Groups and Profiles cannot be copy or pasted.

Rename a Step or Sequence by selecting it from the Data Management Group then clicking on the tool bar. Rename the Step or Sequence then click **OK**.

> **NOTE:** A confirmation dialogue appears when changing the name of a Sequence. Click **OK** to rename or **Cancel** to revert changes.

Copy a Sequence or Step by right-clicking it then selecting **Copy** or clicking on the toolbar.

Paste a Sequence or Step by right-clicking the area of the Data Management Group pane in which to paste then click **Paste** or clicking on the toolbar. The steps of a sequence

> **NOTE:** Pasting a Sequence or Step in the same hierarchy results in "Name_copy".

Sequences

A Sequence is an ordered series of one or more Data Management Steps which will execute in the order in which the Steps are organized. Click to create a new Data Management Sequence.

General

Name

The name given to the Data Management Sequence.

Description (Optional)

A description of the Data Management Sequence.

Data Management Group

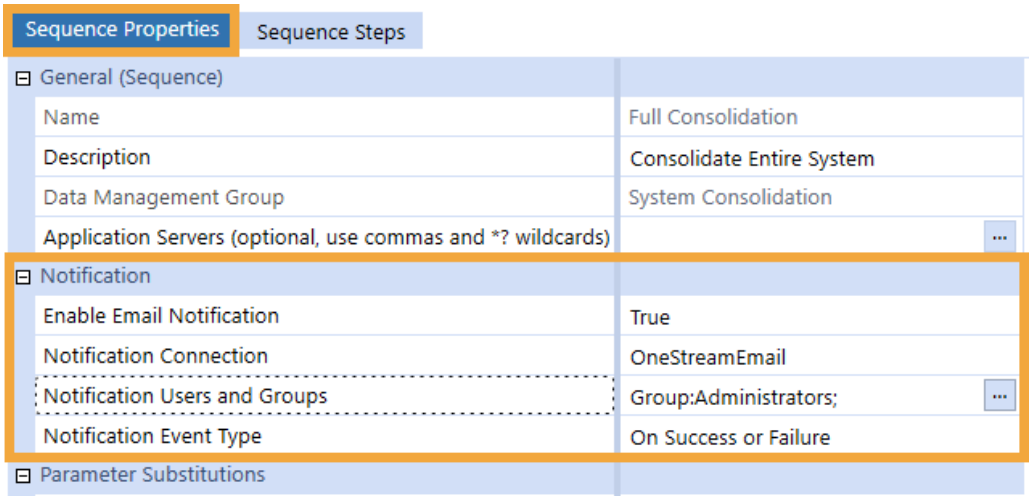
This indicates the Data Management Group under which the particular Sequence was created.

Application Server (Optional)

This allows the specification of a particular application server in the event the Sequence has an abnormally long run-time or one application server is preferred over another.

Notification

Enable email notifications upon the completion, failure, or both, of a Data Management Sequence.



Enable Email Notification

The default value is **False**, meaning email notifications will not be sent. When set to **True**, notifications can be customized by User and Group and are sent based on Event Type.

> **NOTE:** If there is no email server available, this feature is disabled.

Notification Connection

Email server connections are automatically identified. Select the appropriate server.

Notification User and Groups

Select the Users and Groups that need to be notified when the Data Management Sequence runs. Highlight the User or Group, click the left arrow, the click OK when all the Users and Groups have been added.

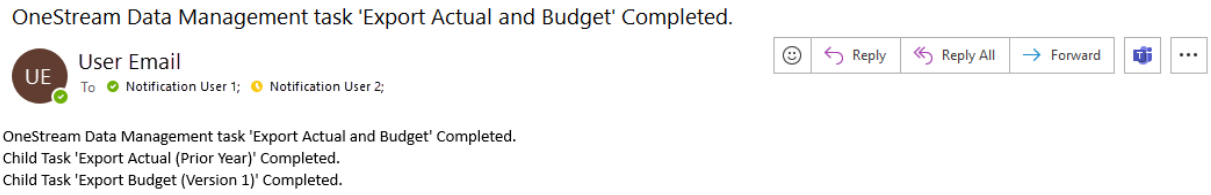
> **NOTE:** The Error Log captures information about Users with no, invalid, or deactivated email accounts.

Notification Event Type

Select the notification event to trigger the email.

- **Not Used** disables the feature when Enable Email Notification is set to **True**.
- **On Success** sends the email when the Data Management Sequence successfully runs.
- **On Failure** sends the email when the Data Management Sequence fails.
- **On Success or Fauilure** sends the email when the Data Management Sequence finishes regardless of status.

This is an example of the notification email:



Task Activity

The Data Management sequence monitors the server's CPU and evaluates other queued tasks in order to make sure that they are processed in order as resources become available. If the server resource utilization is greater than the limit, the job status will be set to Queued and the task progress bar will stay on the queued task step until enough CPU resources are available to start the task. This can be monitored in Task Activity.

Use Queueing

Set this to True to use queueing for this task in order to have better control of the application server's CPU utilization. This task will not start running until CPU utilization is below the specified value, or until all previously queued tasks have been completed. The default is set to True.

Maximum % CPU Utilization To Run

Enter the maximum % CPU utilization allowed on the application server before this task is allowed to transition from queued to running. Enter a number between 1 and 100 or leave this setting blank to use the default. The default is 70%, however this can be modified in the Application Server Configuration.

> **NOTE:** Do not set a value less than 10 or the task may never start.

Maximum Queued Time Before Canceling (minutes)

Enter the number of minutes that this task is allowed to be queued before canceling it automatically. Enter a number, or leave this setting blank to use the default. The default is 20 minutes; however this can be modified in the Application Server Configuration.

When a task gets queued, the task progress dialog will stay on Task Queued. Open Task Activity to monitor the queued task and the current server CPU utilization.

> **NOTE:** Batch processing queue overrides other Workflow batch queue settings. The batch processing queue does not apply to batch script, only the Batch screen in the Workflow.

Parameter Substitutions 1-8

Parameter Name (Optional)
The name of a Parameter being used as a variable to be passed into the Sequence.

Value
The value passed into the related Parameter variable.

Sequence Steps Tab

This is where Steps are assigned and ordered to the Sequence.

Steps

There are six built in Data Management Step types.

Calculate

A Step can be created to use one of the built-in consolidation/calculation options available.

Name
The name of the Data Management Step.

Description (Optional)
The description of the Data Management Step.

Data Management Group
The Data Management Group to which the Step belongs.

Step Type
The type of Data Management Step chosen.

Calculation Type

Calculate
This executes a calculation when Parameter Components are changed. It runs calculations at the Entity level within the Local Member of the Consolidation Dimension without translating or consolidating.

Translate
This executes Translate when Parameter Components are changed. It runs the Calculate step above at the Entity level and then translates data within the Translated Member of the Consolidation Dimension for each applicable relationship.

Consolidate
This executes a Consolidate when Parameter Components are changed. This runs the Calculate and Translate steps and then completes the calculations required all the way up the Consolidation Dimension.

Force Calculate, Translate, or Consolidate
The Force menu (e.g. Force Consolidation) items will run as if every cell included is marked as needing to be calculated, translated or consolidated.

Calculate, Translate, or Consolidate with Logging
The Logging items (e.g. 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.

Force Calculate, Translate, or Consolidate with Logging
This executes a Force Calculation, Translation, or Consolidation with Logging.

Cube
Specify the Cube where the consolidation/calculation will run.

Entity Filter
Specify the Entity, list of Entities or any combination of Entity hierarchies to be included in the consolidation/calculation.

Parent Filter
If alternate hierarchies are used, a Parent may be specified in order to be included in the consolidation/calculation.

Consolidated Filter
Specify the Consolidation Member or Members to be included in the consolidation/calculation.

Scenario Filter
Specify the Scenario Member or Members to be included in the consolidation/calculation.

Time Filter
Specify the Time Member or Members to be included in the consolidation/calculation.

Execute Scenario Hybrid Source Data Copy

Copies base-level Cube data from a specified source Scenario (using the Hybrid Source Data settings) to a specified target Scenario. Data copy occurs when a standard calculation is run on the target Scenario and, by default, follows the standard OneStream calculation sequence.

Clear Data

Name
The name of the Data Management Step.

Description (Optional)
A description of the Data Management Step.

Data Management Group
The Data Management Group to which the Step belongs.

Step Type
The type of Data Management Step chosen.

Use Detailed Logging
If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

Cube
Select the Cube where the consolidation/calculation will run.

Entity Filter
Select the Entity, list of Entities or any combination of Entity hierarchies to be included in the consolidation/calculation.

Scenario Filter
Select the Scenario Member or Members to be included in the consolidation/calculation.

Time Filter
Select the Time Member or Members to be included in the consolidation/calculation.

Clear Imported Data
This indicates whether the Import Member of the Origin Dimension should be included in the Clear Step. Settings are True or False.

Clear Forms Data
This indicates whether the Forms Member of the Origin Dimension should be included in the Clear Step. Settings are True or False.

Clear Adjustment Data and Delete Journals
This indicates whether the Adjustment Members of the Origin Dimension should be included in the Clear Step. Settings are True or False.

Copy Data

Name
The name of the Data Management Step.

Description (Optional)
A description of the Data Management Step.

Data Management Group
The Data Management Group to which the Step belongs.

Step Type
The type of Data Management Step chosen.

Use Detailed Logging
If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

Source Cube
Select the source Cube for the data copy.

Source Entity Filter
Select the source Entity, list of Entities or any combination of Entity hierarchies to be included in the data copy.

Source Scenario
Select the source Scenario Member or Members to be included in the data copy.

Source Time Filter
Select the source Time Member or Members to be included in the data copy.

Source View
Select the View Member to be included in the data copy. This selection allows users to copy YTD data into a Periodic Member and vice versa.

Destination Cube

Select the destination Cube for the data copy.

Destination Entity Filter

Select the destination Entity, list of Entities or any combination of Entity hierarchies to be included in the data copy.

Destination Scenario

Select the destination Scenario Member or Members to be included in the data copy.

Destination Time Filter

Select the destination Time Member or Members to be included in the data copy.

Destination View

Select the destination View Member to be included in the data copy.

Copy Imported Data

Choose True or False to copy O#Import data

Copy Forms Data

Choose True or False to copy O#Forms data.

Copy Adjustment Data

Choose True or False to copy O#Adjustments data.

> **NOTE:** If the Source and Destination View fields are left blank, the data will copy to the same View Member. (e.g., YTD will copy to YTD or Periodic will copy to Periodic)

Custom Calculate

The typical use of the Custom Calculate Step is for speed of calculations during data entry and flexibility. For instance, a user could make on-the-spot changes in a Form, run this Custom Calculate and quickly experience What-if analysis based on the limited amount of data on the Form. Instead of running a full Calculate or Consolidation on a Data Unit, the Custom Calculate Data Management Step can be used to run a calculation on a slice of data within one or many Data Units. The calculation could be executed from within Data Management, by clicking Save on a Form in Workflow (through a Forms Event Handler) or related to a button on a Dashboard being used to enter Budget data, to state a few examples.

This type of calculation does not create audit information for each data cell affected, therefore will run faster than using the Copy Data Data Management Step type.

The user executing must be a member of the Scenario's Manage Data Group or the Step will fail. This helps prevent unauthorized users from launching Steps like this, which could alter or clear data unintentionally.

Name
The name of the Data Management Step.

Description (Optional)
A description of the Data Management Step.

Data Management Group
The Data Management Group to which the Step belongs.

Use Detailed Logging
If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

Data Units

These are the Data Unit settings for what will be affected by calculation being applied. A single Cube and Scenario can be affected by this Step. A single or multiple Entities, Entity Parents, Consolidation members and Time Periods can be affected through Filter settings for Entity, Parent, Consolidation and Time in this section using the Member Filter Builder dialog. The related Business Rule below will run once for each of the number of Data Units specified here. These Data Unit members (e.g. Entity, Scenario, Time, etc.) can be referenced in the related Business Rule with functions such as API.Pov.Entity.Name or similar, but otherwise will not need to be mentioned within the context of the Business Rule. Upon running, this will reset Calculation Status of the affected Data Units to needing calculation.

Point of View

These are the single member entries of dimensions not in the Data Unit to be affected by the calculation. Settings are for View, Account, Flow, Origin, IC and UD1-UD8. These values are provided in the Step as a convenience and to be referenced from within the Business Rule. For instance, the UD1 member listed here in the POV setting could be referenced from within the Business Rule as api.Pov.UD1.Name. This can make the Business Rule flexible since the same rule could be used against multiple Data Management Custom Calculate Steps, but run differently based on the POV setting.

```
Business Rule / Function Name
The name of the Finance-Type Business Rule and contained Function to run when this Step runs.
This allows the user to specify settings such as Durable Storage within the rule. A simple
example of a Business Rule which calculates data with Durable Storage is displayed below:

Select Case api.FunctionType
Case Is = FinanceFunctionType.CustomCalculate
If args.CustomCalculateArgs.FunctionName.XFEqualsIgnoreCase("myFunctionName") Then
    api.Data.Calculate("A#Profit = A#Sales - A#Costs", True)
End If
End Select
```

> **NOTE:** When a Calculation or Consolidation runs on this same Data Unit after this Data Management Step is run, the data saved as calculated by this Step will be cleared unless it is saved with a Storage Type of Durable. ClearCalculatedData is first step in the standard Calculation Sequence that runs during a Calculation or Consolidation on Cube data. In this case, Durable data will be ignored during a calculate or even a Force Calculate or Consolidate unless a ClearCalculated function is used within the Business Rule or Member Formula to purposely clear the Durable data. However, if a calculation recalculates even data marked as Durable, it will then be replaced by the newly calculated data.

It is suggested that if this calculation within a Custom Calculate Data Management Step is to be replicated within a separate Member Formula or Business Rule, both can refer to the same saved Finance Business Rule function by name. To refer to this logic, see Defining a Reference to a Shared Business Rule in Implementing Security.

See Examples of Key Functions in About the Financial Model for examples of using API.Data.Calculate and API.Data.ClearCalculated to store and clear Durable data as desired.

Parameters

Refer to any parameters inside the Business Rule by listing them here in name-value pairs in this fashion:

```
Name1=Frankfurt, Name2=[Houston Heights]
```

Custom Parameters can be used by using the correct syntax. This will result in a prompt to the user at run time.

```
Name3=|!myParam!|
```

Execute Business Rule

Name

The name of the Data Management Step.

Description (Optional)
A description of the Data Management Step.

Data Management Group
The Data Management Group to which the Step belongs.

Step Type
The type of Data Management Step chosen.

Use Detailed Logging
If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

Business Rule
Select from the available application Business Rules to run custom scripts or procedures.

Parameters (Optional)
This field is provided to pass Parameters or variables into the selected Business Rule.

Export Data

Name
The name of the Data Management Step.

Description (Optional)
A description of the Data Management Step.

Data Management Group
The Data Management Group to which the Step belongs.

Step Type
The type of Data Management Step chosen.

Use Detailed Logging

If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

File Share File Name

The name given to the file being exported to the application file share.

Include Cube in File Name

If set to True, the Cube name will be included in the file name when exporting.

Include Entity in File Name

If set to True, the Entity name will be included in the file name when exporting.

Include Parent in File Name

If set to True, the Parent name will be included in the file name when exporting.

Include Cons in File Name

If set to True, the Consolidation Member name will be included in the file name when exporting.

Include Scenario in File Name

If set to True, the Scenario name will be included in the file name when exporting.

Include Time in File Name

If set to True, the Time name will be included in the file name when exporting.

Overwrite Existing Files

Set this to True to overwrite a previously exported file when exporting.

Include Zeroes

If set to True, zero amount records will be included when exporting.

Include Member Descriptions

If set to True, Member descriptions will be included when exporting.

Include Cell Annotations

If set to True, cell annotations will be included when exporting.

Include Input Data
If set to True, input type data will be included when exporting.

Include Calculated Data
If set to True, calculated data will be included when exporting.

Cube
Specify the Cube where consolidation/calculation will run.

Entity Filter
Specify the Entity, list of Entities or any combination of Entity hierarchies to be included in the consolidation/calculation.

Parent Filter
If alternate hierarchies are used, a Parent may be specified in order to be included in the consolidation/calculation.

Consolidated Filter
Specify the Consolidation Member or Members to be included in the consolidation/calculation.

Scenario Filter
Specify the Scenario Member or Members to be included in the consolidation/calculation.

Time Filter
Specify the Time Member or Members to be included in the consolidation/calculation.

Combinations of Data Filters (use #All for all stored base-level data)
Use standard Member Filter functionality to select specific data required for all Dimensions listed below:

Account Filter (e.g., “A#IS.Descendants, A#All”)
Flow Filter
Origin Filter
IC Filter
UD1-8 Filter

Specific Data Filters

Use standard Member Filter functionality in order to have fine grain control over the data exported. Use this filter to choose more specific intersections of data.

Export File

Use the Export File Data Management Step to export an Extensible Document or Report Book to OneStream's File Share.

Name

The name of the Data Management Step.

Description (Optional)

A description of the Data Management Step.

Data Management Group

The Data Management Group to which the Step belongs.

Step Type

The type of Data Management Step chosen.

Use Detailed Logging

If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

File Share Folder

Enter the name of the File Share Folder where the desired file is saved.

File Name Suffix

Enter a suffix to include in the name of the file.

Overwrite Existing Files

Set to True to overwrite a previously exported file when exporting.

File Source Type

URL

Display a file from an internal or external web page.

Application/System Dashboard File

Display a file stored in a Dashboard Maintenance Unit File section.

Application/System Database File

Display a file stored in the Application or System Database Share.

File Share File

Display a file from the File Share.

URL or Full File Name

The URL or name of the file being used. Enter the full URL name, or click the ellipsis and browse to the desired file.

Process Extensible Document

If set to True, the Data Management Step will run and process the attached Extensible Document file, if set to False the unprocessed file will display which is mainly used for testing purposes.

> **NOTE:** An Extensible Document is a Text, Word, PowerPoint, or Excel file that uses Parameters in its content. The file name must contain .xfDoc before the extension.

```
Example: StatusReport.xfDoc.docx
```

See Extensible Document Framework in Presenting Data With Extensible Documents for more details on this feature.

Read Parameter Names from Extensible Doc

If set to True, the Parameters used in the Extensible Document will prompt the user when the Data Management Sequence is run, if this is set to False, the Parameters can be specified ahead of time in the Parameters property below.

Parameters

Enter a comma separated list of name value pairs.

```
Example: ParameterName1=ValueName1, ParameterName2=[Value Name2]
```

Export Report

Name

The name of the Data Management Step.

Description (Optional)

A description of the Data Management Step.

Data Management Group

The Data Management Group to which the Step belongs.

Step Type

The type of Data Management Step chosen.

Use Detailed Logging

If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

File Share File Name

The name given to file being exported to the application file share.

Include Object Name in File

If set to True, the Dashboard Object will be included in the file name when exporting.

Overwrite Existing Files

Set to True to overwrite a previously exported file when exporting.

Report File Type

PDFs in Zip File
This will export all contained PDF's individually in a ZIP file.

Combined PDF File
This will combine and then export all report PDF's into one PDF file.

Object Type
Indicate whether a Dashboard or Dashboard Profile object is to be exported.

Object Name
The name of the Dashboard or Dashboard Profile being used.

Object Parameters (Optional)
This field is provided to pass Parameters or variables used when running the report normally.

Reset Scenario

Similar to Clear Data Step, except it clears additional related application data, yet does not create audit information for each data cell affected, therefore will run faster. It clears data within a range (including parent Entity data), audit information, Workflow Status and Calculation Status as if it never existed. This is intended for Administrator use. User executing must be a member of the Scenario's Manage Data Group or the Step will fail. This helps prevent unauthorized users from launching Steps like this, which could alter or clear data unintentionally. It is recommended to only changes Manage Data Group from Nobody to a select exclusive User Group before running and then change back to Nobody afterwards. Also, ensure that your application database is backed up before performing a Reset Scenario. Note that Reset Data will clear even data marked as Durable.

Name
The name of the Data Management Step.

Description (Optional)
A description of the Data Management Step.

Data Management Group
The Data Management Group to which the Step belongs.

Use Detailed Logging
If set to True, a detailed log found in the Task Activity Log will provide additional detail about the process.

Scenario
Choose one Scenario to be reset.

Reset All Time Periods
Default is True, which will remove data and reset Workflow Status for all time periods. If False, the next three properties will be enabled.

Start Year / End Year
The first and last years' worth of time to clear from the Application and Workflow Status.

Start Time Period in First Year
Optional. Time Period referenced in First Year to begin clearing data.

Run a Sequence or Step

While highlighting a Sequence or Step, click this button: .

When running a Sequence in Data Management, go to **System Tab > Tools > File Explorer** to find any file export.

Go to **File Share > Applications > Choose an application > DataManagement > Export > Username > Choose the latest folder**.

Spreadsheet

The OneStream Windows App Spreadsheet feature enables users to incorporate spreadsheet functionality experienced by users of the OneStream Excel Add-in without needing to have Excel loaded on their computer. It can be used for ad hoc querying/reporting, analysis, data entry, and formatted reports. The Spreadsheet feature enables users to stay within the OneStream Windows App while being able to use functionality similar to what they know and love in Excel. Similar to the Excel Add-in, the Spreadsheet feature leverages OneStream's re-usable Cube Views for fast and easy analysis. See Navigating the Excel Add-In for more details on this feature.

- Here are some other functional highlights and benefits:
- Change point-of-view, interact with Forms, assign Cube Views, drill through to source data and update Workflow status
- Utilize familiar retrieve functions for ad-hoc reporting and analysis
- Eliminate risk of errors and duplication of efforts with standardized and centralized spreadsheet controls
- Safely edit, update and analyze data as spreadsheet forms respect Application and Workflow and security
- Eliminate spreadsheet maintenance when metadata changes because Cube Views are read dynamically
- Sheet-based calculations remain intact even when rows or columns are added
- Include multiple Cube View results in the same sheet

The OneStream Windows App Spreadsheet feature can perform most of the tasks accomplished by the OneStream Excel Add-in, with some limitations. These limitations include, but are not limited to:

- Macros
- Solver
- Document properties
- No option to insert copied/cut cells in spreadsheet
- Preview format does not update for format types Effects (Superscript, Subscript, Strikethrough)
- Ability to manage conditional formatting – color scales are reversed
- Shift+End
- Spreadsheet border button does not store last selection
- Text Direction option from Alignment formatting
- Underline formatting does not appear same as in Excel
- Not all Chart options are available in Spreadsheet
- Ability to change chart colors in spreadsheet
- References to other sheets in Spreadsheet is a different process
- Ctrl+PageUp / Ctrl+PageDn correctly changes the tab but does not reset tab focus in spreadsheet
- CTRL+N creates a new spreadsheet, however if an existing spreadsheet was opened with unsaved changes, it closes the existing spreadsheet without saving edits made since last save

Some functionality is currently not supported in the Spreadsheet Tool. Functionality that is known not to be supported in the Spreadsheet feature at this time is listed below:

- Fill number formatting
- The ability to format part of a cell with different formatting
- Justify alignment option
- Managing Rules under spreadsheet conditional formatting
- Spreadsheet alignment cell format "Center Across Selection"
- Undo/Redo

The following Excel Column Charts are incompatible and unavailable with the Spreadsheet feature:

- Stacked 3-D
- 100% Stacked 3-D
- Stacked Cylinder
- 100% Stacked Cylinder
- Stacked Cone
- 100% Stacked Cone
- Stacked Pyramid
- 100% Stacked Pyramid

The following Excel Line chart is incompatible and unavailable with the Spreadsheet feature:

- 3-D Line

The following Excel Bar charts are incompatible and unavailable with the Spreadsheet feature:

- Stacked 3-D Bar
- 100% Stacked 3-D Bar
- Stacked Horizontal Cylinder
- 100% Stacked Horizontal Cylinder
- Stacked Horizontal Cone
- 100% Stacked Horizontal Cone
- Stacked Horizontal Pyramid
- 100% Stacked Horizontal Pyramid

Microsoft Excel 2016 Chart Types

There are seven new chart types added to the Spreadsheet component in the Windows application. These charts were first introduced to Microsoft Excel 2016, and are now available in our spreadsheet component as an alternative to using Excel.

The OneStream Windows application spreadsheet feature does not require that Excel be loaded on your computer. It can be used for querying and reporting, analysis, data entry, and formatted reports. The Spreadsheet feature allows you to remain in the Windows application using Excel-like functionality.

This is an enhancement to the existing spreadsheet component in the Windows application. It is available to administrators or general users with security rights to access it.

Chart Types

Microsoft Excel 2016 introduced several chart types that help you visualize financial, statistical, and hierarchical data. They include:

- Box and Whisker
- Waterfall
- Histogram
- Pareto
- Funnel
- Sunburst
- Treemap

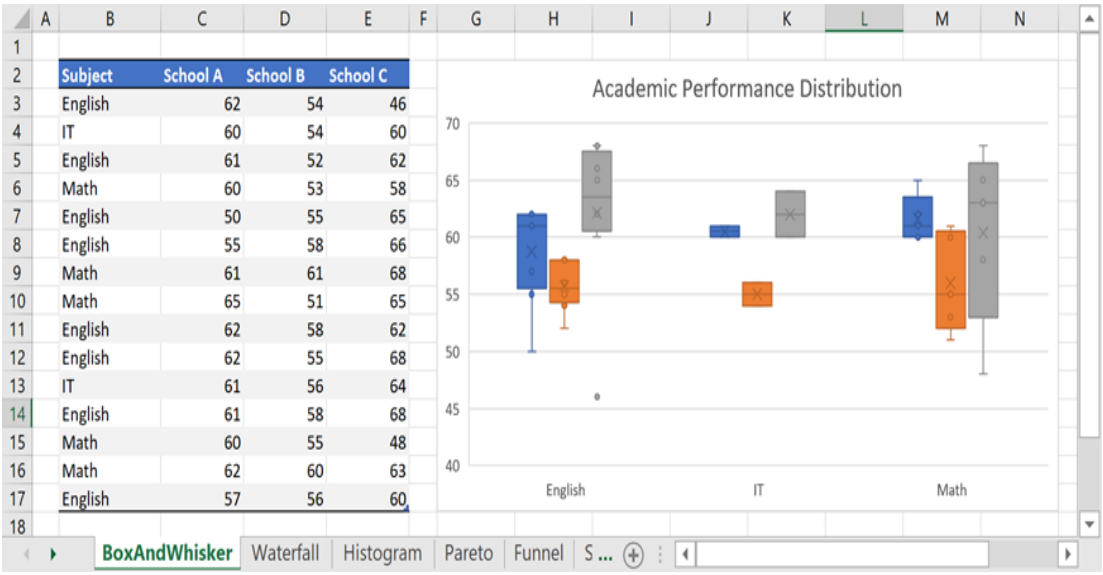
You can add charts to a OneStream spreadsheet worksheet in the same manner as any other chart type. Chart styles allow you to quickly change chart appearance. Styles change the background fill, specify the color of the data series, and apply different shape effects and outlines to the chart.

To apply Excel 2016 chart types to a predefined chart style, right-click on the chart and select one of the available styles.

Box and Whisker

A box and whisker chart shows distribution of data in quartiles, highlighting the mean and outliers. Boxes may have lines extending vertically called “whiskers.” These lines indicate variability outside the upper and lower quartiles, and any point outside those lines or whiskers is considered an outlier.

Box and whisker charts are commonly used in statistical analysis. For example, you could use a box and whisker chart to compare medical trial results or teachers' test scores.



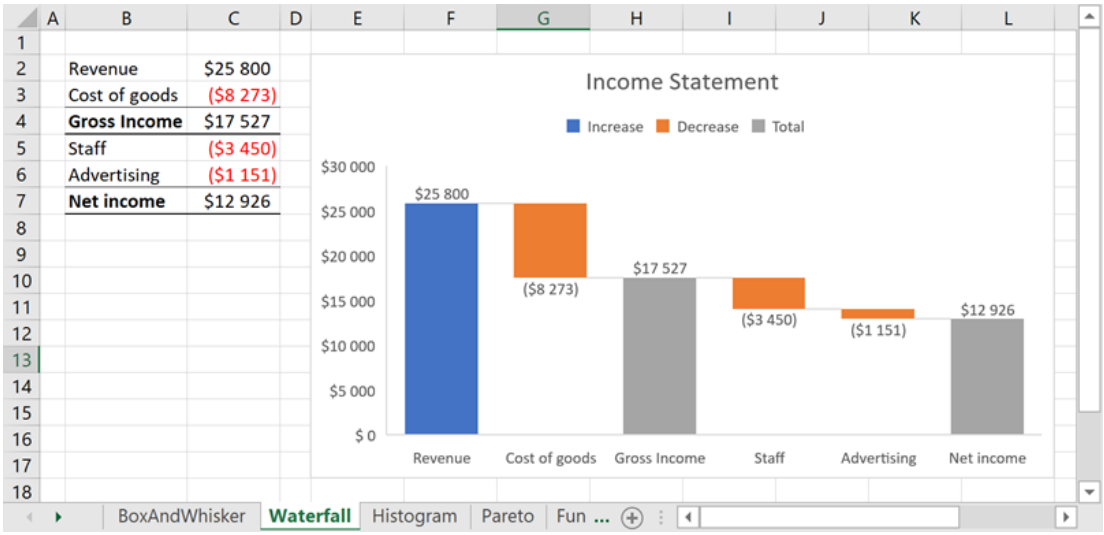
Subject	School A	School B	School C
English	62	54	46
IT	60	54	60
English	61	52	62
Math	60	53	58
English	50	55	65
English	55	58	66
Math	61	61	68
Math	65	51	65
English	62	58	62
English	62	55	68
IT	61	56	64
English	61	58	68
Math	60	55	48
Math	62	60	63
English	57	56	60

Waterfall

A waterfall chart shows a running total as values are added or subtracted. It is useful for understanding how an initial value (for example, net income) is affected by a series of positive and negative values. The columns are color coded so you can quickly tell positive from negative numbers. The initial and the final value columns often start on the horizontal axis, while the intermediate values are floating columns.

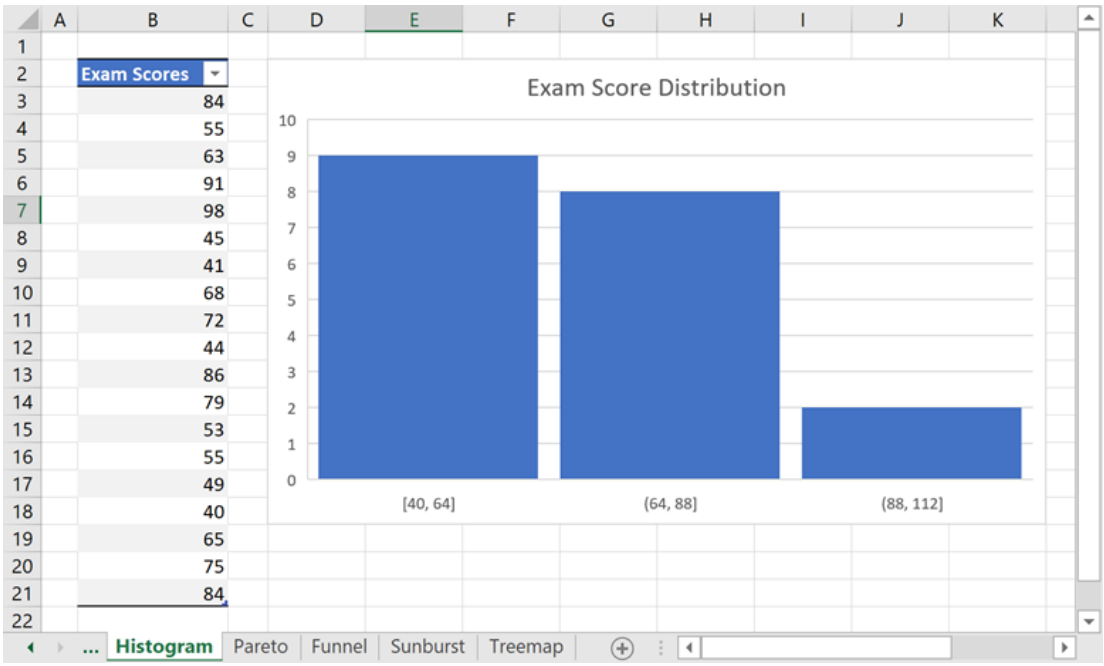
If your data includes values that are considered Subtotals or Totals, such as Net Income, you can set those values so they start on the horizontal axis at zero and don't float. To do this:

1. Right-click on the chart and select **Set as Total**.
2. In the dialog box, select the checkbox for value(s) in the dataset to be set as Total.
3. To make the column float again, clear the **Set as total** checkbox.



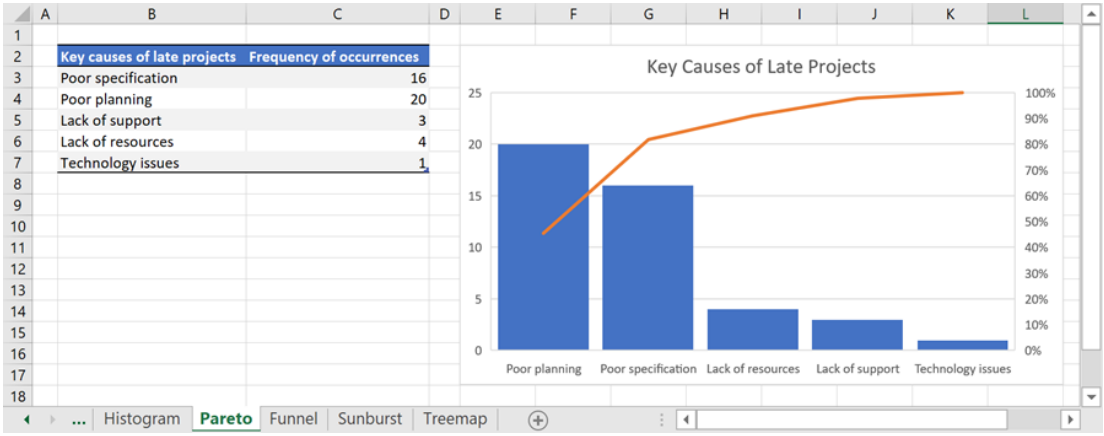
Histogram

Data plotted in a histogram chart shows the frequencies within a distribution. Each column of the chart is called a frequency bin.



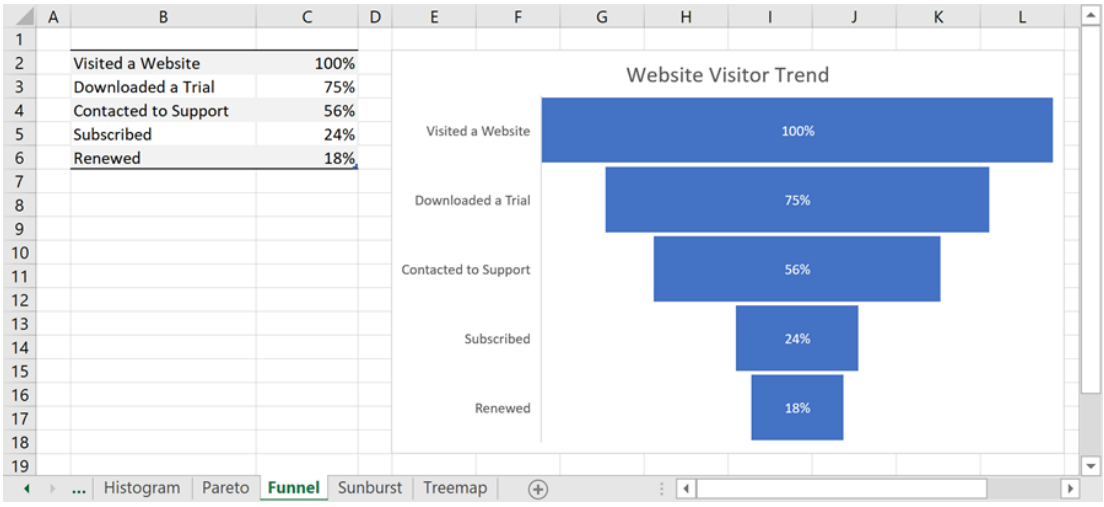
Pareto

A pareto chart is a sorted histogram that contains columns sorted in descending order along with a line representing the cumulative total percentage.



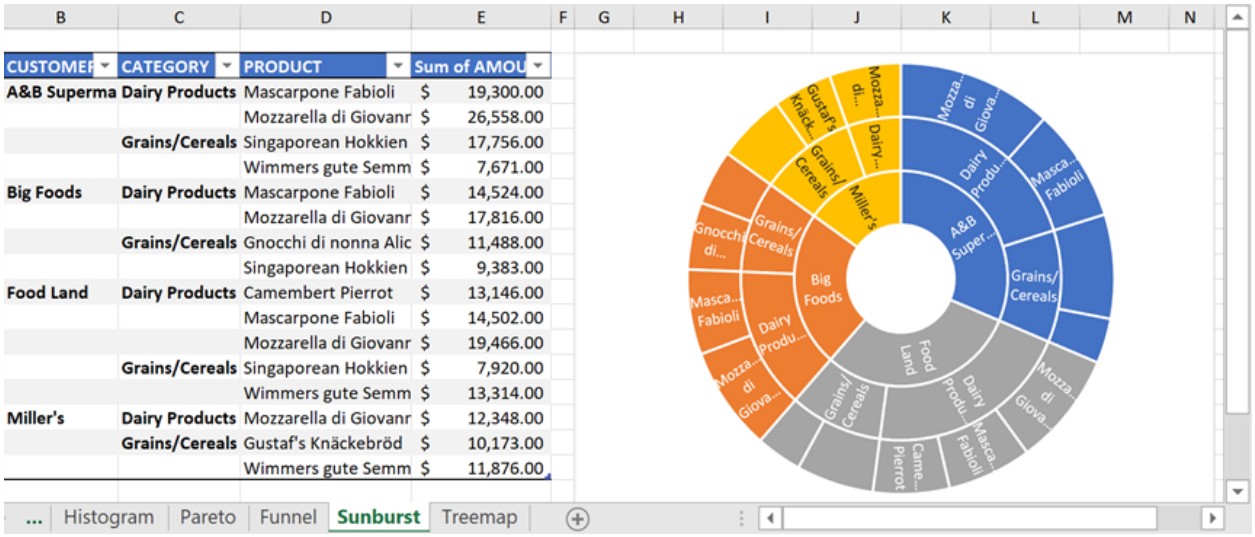
Funnel

Funnel charts show values across multiple stages in a process. Typically, the values decrease gradually, causing the bars to resemble a funnel. For example, you could use a funnel chart to show the number of sales prospects at each stage in a sales pipeline.



Sunburst

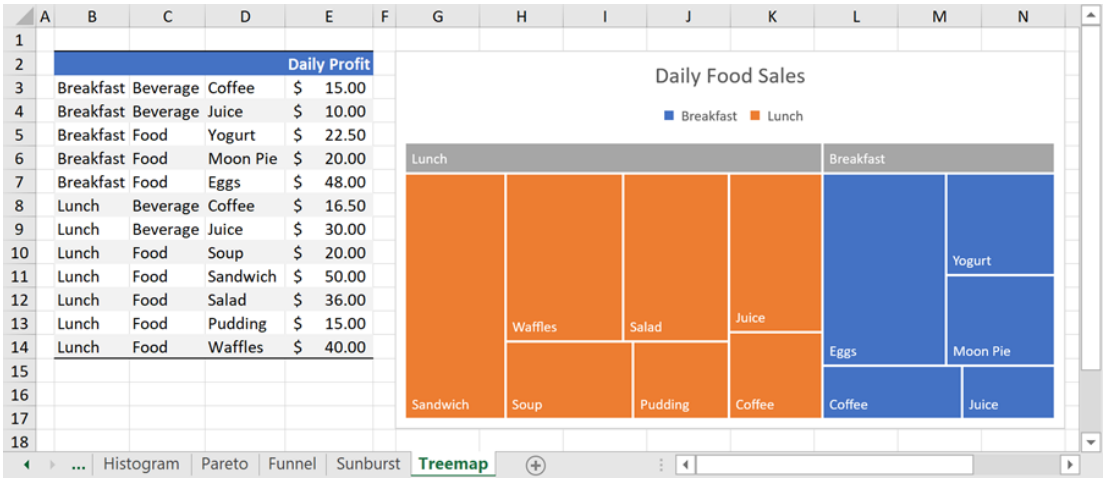
The sunburst chart displays hierarchical data and can be plotted when empty (blank) cells exist within the hierarchical structure. Each level of the hierarchy is represented by one ring or circle with the innermost circle as the top of the hierarchy. A sunburst chart without hierarchical data (one level of categories), looks similar to a doughnut chart. However, a sunburst chart with multiple levels of categories shows how the outer rings relate to the inner rings. The sunburst chart is most effective at showing how one ring is broken into its contributing pieces.



CUSTOMER	CATEGORY	PRODUCT	Sum of AMOU
A&B Superma	Dairy Products	Mascarpone Fabioli	$ 19,300.00
		Mozzarella di Giovanr	$ 26,558.00
	Grains/Cereals	Singaporean Hokkien	$ 17,756.00
		Wimmers gute Semm	$ 7,671.00
Big Foods	Dairy Products	Mascarpone Fabioli	$ 14,524.00
		Mozzarella di Giovanr	$ 17,816.00
	Grains/Cereals	Gnocchi di nonna Alic	$ 11,488.00
		Singaporean Hokkien	$ 9,383.00
Food Land	Dairy Products	Camembert Pierrot	$ 13,146.00
		Mascarpone Fabioli	$ 14,502.00
		Mozzarella di Giovanr	$ 19,466.00
	Grains/Cereals	Singaporean Hokkien	$ 7,920.00
		Wimmers gute Semm	$ 13,314.00
Miller's	Dairy Products	Mozzarella di Giovanr	$ 12,348.00
	Grains/Cereals	Gustaf's Knäckebröd	$ 10,173.00
		Wimmers gute Semm	$ 11,876.00

Treemap

The treemap chart provides a hierarchical view of your data and an easy way to compare different levels of categorization. The treemap chart displays categories by color and proximity and can show data which might be difficult to display with other chart types. The treemap chart is plotted when empty (blank) cells exist within the hierarchical structure. Treemap charts are good for comparing proportions within the hierarchy.



			Daily Profit
Breakfast	Beverage	Coffee	$ 15.00
Breakfast	Beverage	Juice	$ 10.00
Breakfast	Food	Yogurt	$ 22.50
Breakfast	Food	Moon Pie	$ 20.00
Breakfast	Food	Eggs	$ 48.00
Lunch	Beverage	Coffee	$ 16.50
Lunch	Beverage	Juice	$ 30.00
Lunch	Food	Soup	$ 20.00
Lunch	Food	Sandwich	$ 50.00
Lunch	Food	Salad	$ 36.00
Lunch	Food	Pudding	$ 15.00
Lunch	Food	Waffles	$ 40.00

Conclusion

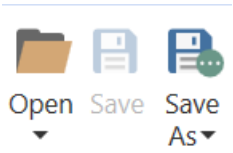
These chart types help you visualize financial, statistical, and hierarchical data inside the Windows application spreadsheet component. They are ideal for those who prefer working inside the Windows application rather than Microsoft Excel.

Retrieve Functions

Retrieving and changing data can be done by using functions. To see the functions and their Parameters, open the Spreadsheet feature and select the Formulas tab. Within the Spreadsheet feature, select Insert Function and select User Defined where it says to Select a category. See Retrieve Functions in Navigating the Excel Add-In for more details on this feature.

File, Open and File, Save As Functionality

Additional options for opening and saving files exist within the Spreadsheet feature.



Open

File type to open

Local Folder

Select a file to be opened from a location on the local computer/network.

OneStream File System

Select a file to be opened from a location within the File Explorer.

> **NOTE:** Displayed files can also be opened from here, or from OnePlace |Documents by right clicking and selecting one of the three options – “Open in Spreadsheet Page”; “Open” (opens file directly in Excel if the application is found on the local computer) or “Open With…” (user specifies program).

Application Workspace File

Select a file to be opened from an Application Workspace.

System Workspace File

Select a file to be opened from a System Workspace.

Save As

File type to be saved as.

Local Folder

Select location on local computer/network to save a file to.

OneStream File System

Select location within File Explorer to save a file to.

Application Workspace File
Select Application Workspace location to save a file to.

System Dashboard File
Select System Workspace location to save a file to.

Reduce Cell Text Size

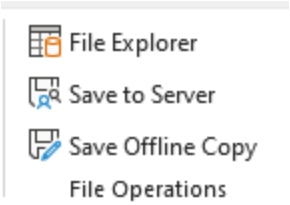
You can automatically reduce text size to fit inside a spreadsheet cell. This eliminates the need to resize the column width and prevents text from being cut off when it exceeds the cell width.

1. Inside the spreadsheet, select a single cell, multiple cells, or all cells.
2. Right-click on the selected cells and select **Format Cells**.
3. Click the **Alignment** tab.
4. Select the **Shrink to fit** checkbox.
5. Click **OK**.

OneStream Ribbon

When using the Spreadsheet feature, there will be a OneStream menu item and a ribbon. See OneStreamRibbon Navigating the Excel Add-In for an explanation of the items within Spreadsheet that function the same as those in the Excel Add-In. Items documented below function differently than they do in the Excel Add-In, or do not exist in the Spreadsheet Tool.

File Operations

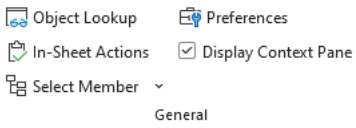


The File Operations grouping in the Excel Add-In does not exist in Spreadsheet. The Save to Server and File Explorer ribbon options are not available because they are integrated with the OneStream application within Spreadsheet. Save an Offline Copy is under the General Grouping in Spreadsheet.

Explore

Create Copy of Selected Quick View is available in Spreadsheet and copied Quick Views can only be pasted within the same workbook. See Create Quick Views in Navigating the Excel Add-In for more options.

General

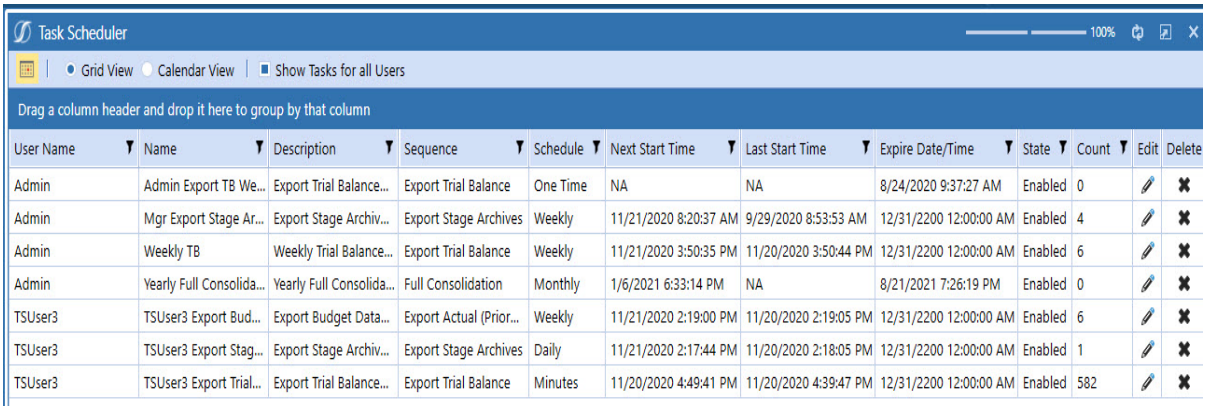


The Display Context Pane option does not exist in the Spreadsheet feature because it is not needed. The OneStream task pane automatically appears when using the Spreadsheet feature.

Under Preferences, the General options are not needed in the Spreadsheet for Windows Only tool. These options manage how Microsoft Sign In is handled, how Macros are handled, and how data is refreshed in an open workbook. The Spreadsheet feature does not utilize this functionality.

Task Scheduler

Task Scheduler provides the ability to schedule data management sequences that execute a data management step within the application. If the sequence doesn't have a step, the job will fail.



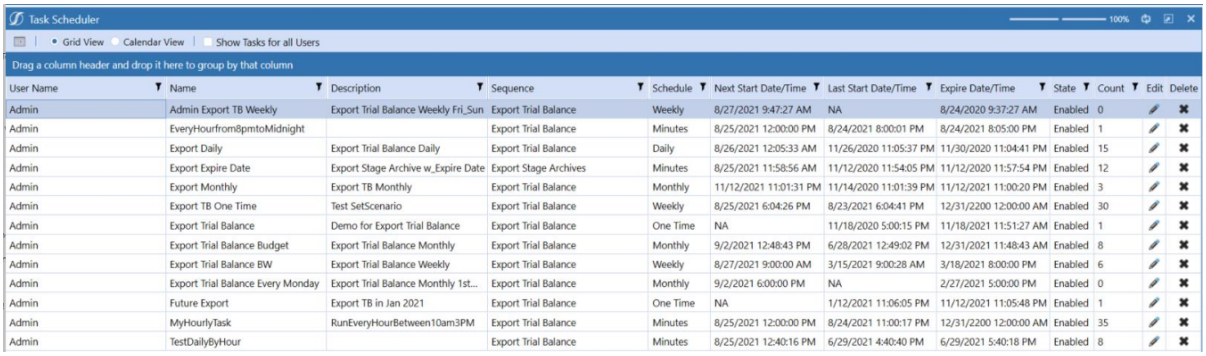
User Name	Name	Description	Sequence	Schedule	Next Start Time	Last Start Time	Expire Date/Time	State	Count	Edit	Delete
Admin	Admin Export TB We...	Export Trial Balance...	Export Trial Balance	One Time	NA	NA	8/24/2020 9:37:27 AM	Enabled	0		
Admin	Mgr Export Stage Ar...	Export Stage Archiv...	Export Stage Archives	Weekly	11/21/2020 8:20:37 AM	9/29/2020 8:53:53 AM	12/31/2200 12:00:00 AM	Enabled	4		
Admin	Weekly TB	Weekly Trial Balance...	Export Trial Balance	Weekly	11/21/2020 3:50:35 PM	11/20/2020 3:50:44 PM	12/31/2200 12:00:00 AM	Enabled	6		
Admin	Yearly Full Consolida...	Yearly Full Consolida...	Full Consolidation	Monthly	1/6/2021 6:33:14 PM	NA	8/21/2021 7:26:19 PM	Enabled	0		
TSUser3	TSUser3 Export Bud...	Export Budget Data...	Export Actual (Prior...	Weekly	11/21/2020 2:19:00 PM	11/20/2020 2:19:05 PM	12/31/2200 12:00:00 AM	Enabled	6		
TSUser3	TSUser3 Export Stag...	Export Stage Archiv...	Export Stage Archives	Daily	11/21/2020 2:17:44 PM	11/20/2020 2:18:05 PM	12/31/2200 12:00:00 AM	Enabled	1		
TSUser3	TSUser3 Export Trial...	Export Trial Balance...	Export Trial Balance	Minutes	11/20/2020 4:49:41 PM	11/20/2020 4:39:47 PM	12/31/2200 12:00:00 AM	Enabled	582		

Grid View

Go to **Application > Tools > Task Scheduler**.

The default view is the calendar view. You can change the view to a grid view which is initially blank until you've scheduled sequences.

The default is to Show Tasks for all Users, but you can click the option to turn it off. Once you've created tasks, they will show in the grid.



User Name	Name	Description	Sequence	Schedule	Next Start Date/Time	Last Start Date/Time	Expire Date/Time	State	Count	Edit	Delete
Admin	Admin Export TB Weekly	Export Trial Balance Weekly Fri_Sun	Export Trial Balance	Weekly	8/27/2021 9:47:27 AM	NA	8/24/2020 9:37:27 AM	Enabled	0		
Admin	EveryHourfrom8pmtoMidnight		Export Trial Balance	Minutes	8/25/2021 12:00:00 PM	8/24/2021 8:00:01 PM	8/24/2021 8:05:00 PM	Enabled	1		
Admin	Export Daily	Export Trial Balance Daily	Export Trial Balance	Daily	8/26/2021 12:05:33 AM	11/26/2020 11:05:37 PM	11/30/2020 11:04:41 PM	Enabled	15		
Admin	Export Expire Date	Export Stage Archive w_Expire Date	Export Stage Archives	Minutes	8/25/2021 11:58:56 AM	11/12/2020 11:54:05 PM	11/12/2020 11:57:54 PM	Enabled	12		
Admin	Export Monthly	Export TB Monthly	Export Trial Balance	Monthly	11/12/2021 11:01:31 PM	11/14/2020 11:01:39 PM	11/12/2021 11:00:20 PM	Enabled	3		
Admin	Export TB One Time	Test SetScenario	Export Trial Balance	Weekly	8/25/2021 6:04:26 PM	8/23/2021 6:04:41 PM	12/31/2200 12:00:00 AM	Enabled	30		
Admin	Export Trial Balance	Demo for Export Trial Balance	Export Trial Balance	One Time	NA	11/18/2020 5:00:15 PM	11/18/2021 11:51:27 AM	Enabled	1		
Admin	Export Trial Balance Budget	Export Trial Balance Monthly	Export Trial Balance	Monthly	9/2/2021 12:48:43 PM	6/28/2021 12:49:02 PM	12/31/2021 11:48:43 AM	Enabled	8		
Admin	Export Trial Balance BW	Export Trial Balance Weekly	Export Trial Balance	Weekly	8/27/2021 9:00:00 AM	3/15/2021 9:00:28 AM	3/18/2021 8:00:00 PM	Enabled	6		
Admin	Export Trial Balance Every Monday	Export Trial Balance Monthly 1st...	Export Trial Balance	Monthly	9/2/2021 6:00:00 PM	NA	2/27/2021 5:00:00 PM	Enabled	0		
Admin	Future Export	Export TB in Jan 2021	Export Trial Balance	One Time	NA	1/12/2021 11:06:05 PM	11/12/2021 11:05:48 PM	Enabled	1		
Admin	MyHourlyTask	RunEveryHourBetween10am3PM	Export Trial Balance	Minutes	8/25/2021 12:00:00 PM	8/24/2021 11:00:17 PM	12/31/2200 12:00:00 AM	Enabled	35		
Admin	TestDailyByHour		Export Trial Balance	Minutes	8/25/2021 12:40:16 PM	6/29/2021 4:40:40 PM	6/29/2021 5:40:18 PM	Enabled	8		

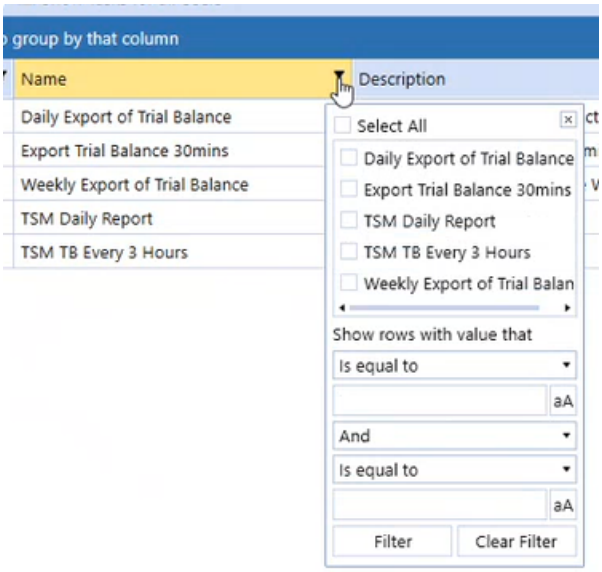
Whatever view you are currently on is the view you will come back to the next time you go to the page.

Task Scheduler Details

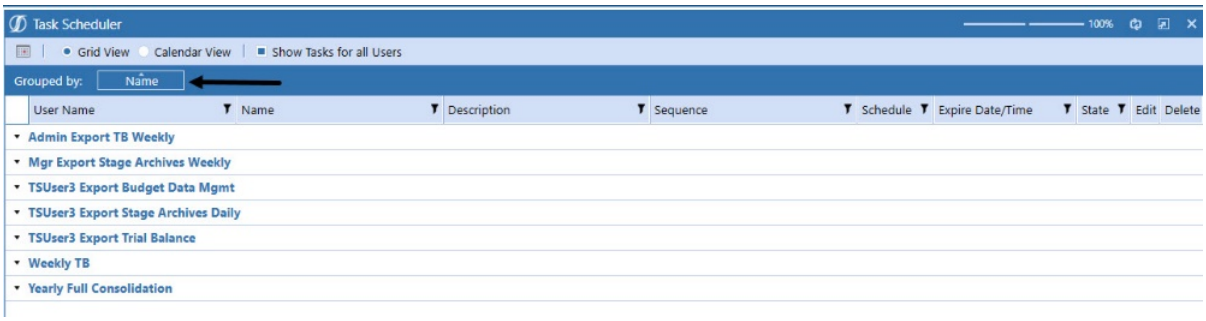
Field	Definition
User Name	The user that created the task.
Name	Name of the task.
Description	Description of the task.
Sequence	Data management sequence that is run by the task scheduler.
Schedule	The time frame implemented for the specific task.
Next Start Time	The next time the task is scheduled to run.
Last Start Time	The last time the task ran.

Field	Definition
Expire Date/Time	The time and date the task will expire and no longer run.
State	Enabled or disabled.
Count	The amount of time the task has ran.
Edit	Click to edit.
Delete	Click to delete.
If enabled by an administrator:	
Invalidate Date/Time	The time and date the task will be suspended and not run until validated.
Validate Task	Validate task to keep active

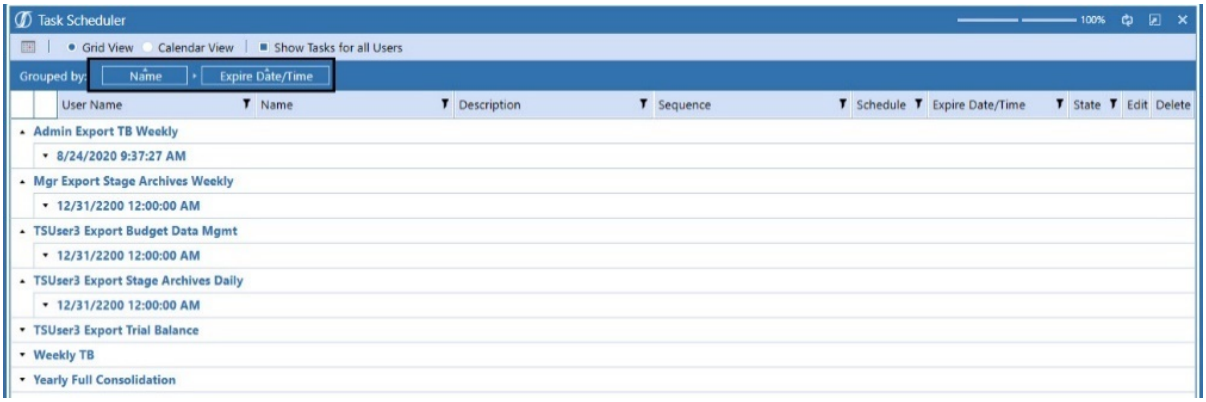
You can filter on any of the fields in the grid that have the filter icon and you can filter on multiple selections.



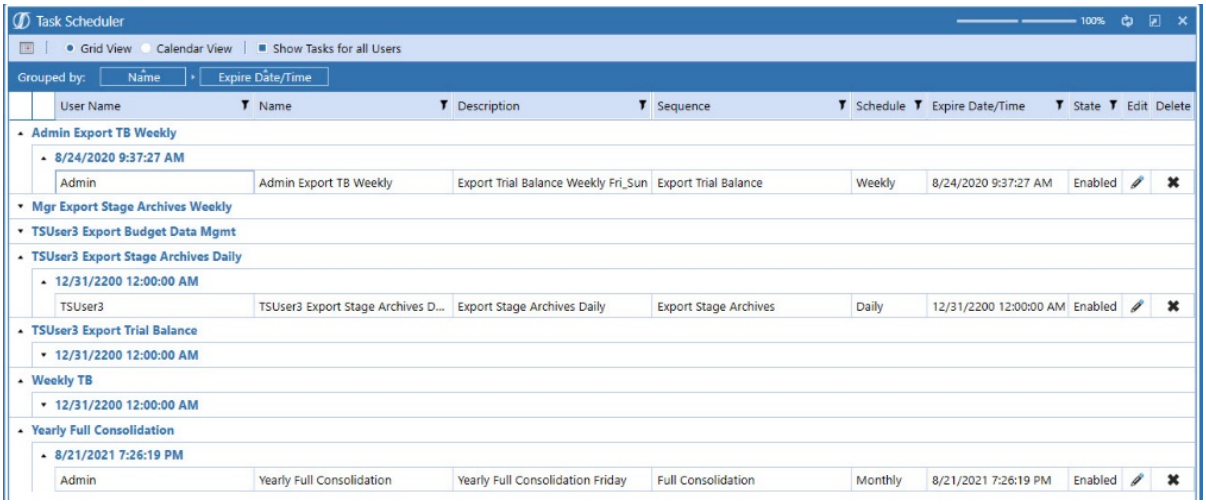
You can group by column name by dragging and dropping the group into the header bar.



You can group by more than one column.

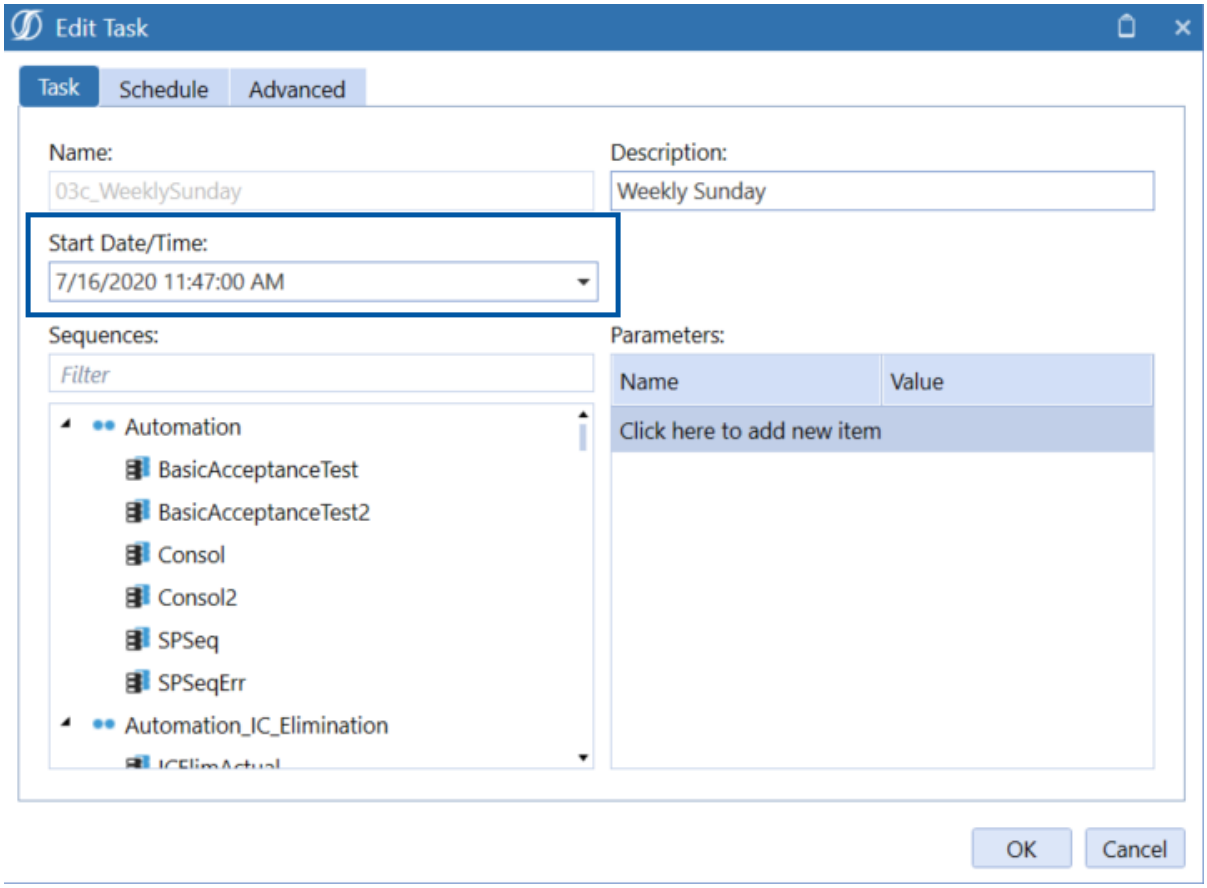


You can expand or collapse by the task.

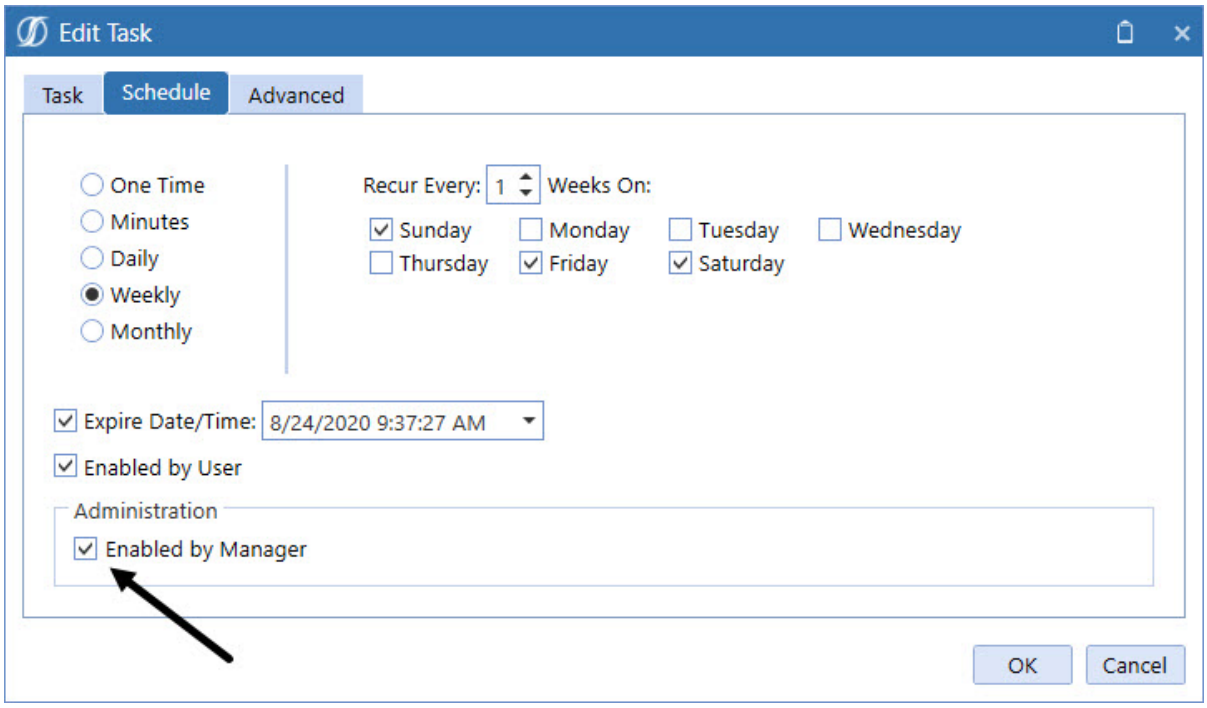


Double-click on a task or click Edit to open Edit Task. The only option that you can not change is the **Name** in the Task tab.

The **Start Date/Time** are your local time.

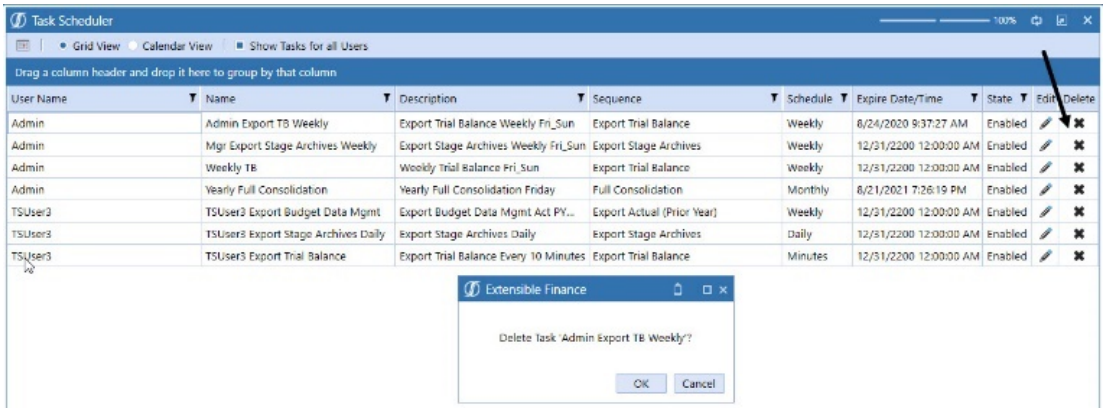


If you are not the Administrator, you will not have the rights to change the **Enabled by Manager** check box.



You can do any of the following Delete options:

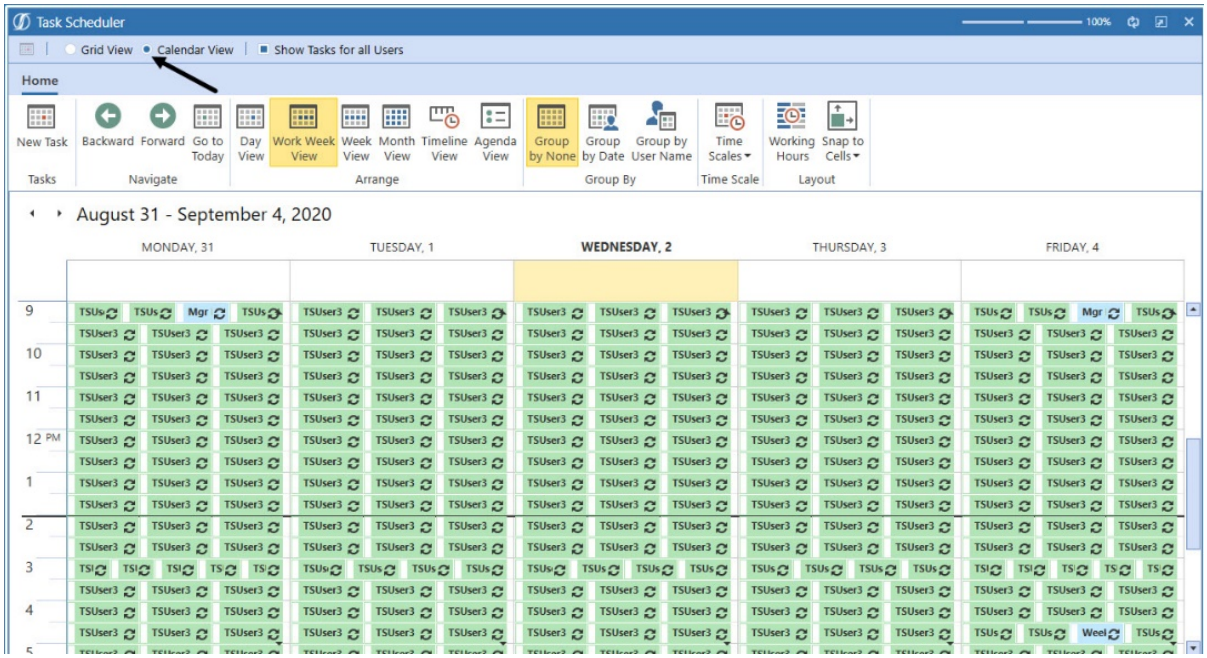
- Click to **Delete**.



- Select the task and right-click to delete.

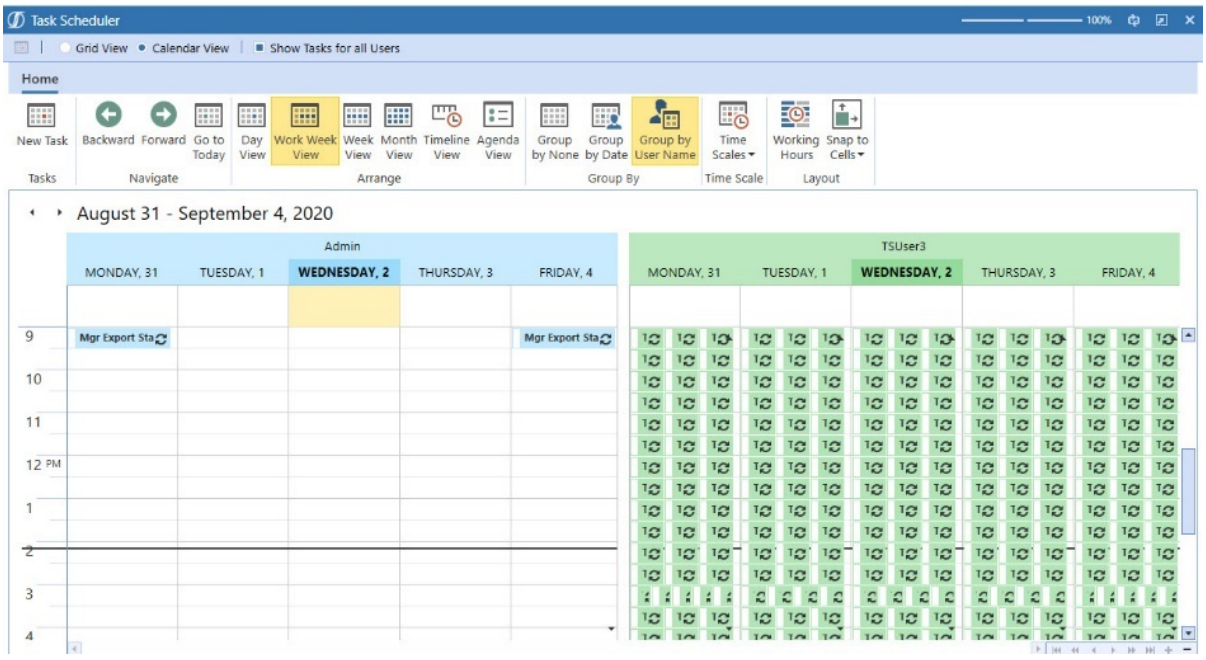
Calendar View

Provides you a view of the task in the calendar.

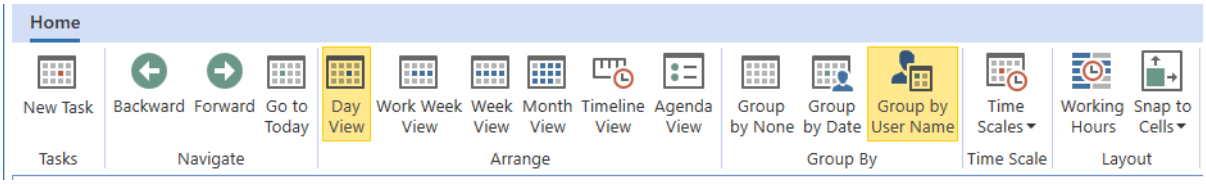


If there is more than one user showing, the tasks are color coded when Show Tasks for All Users is selected.

You can view your own tasks and all user's tasks if checked.

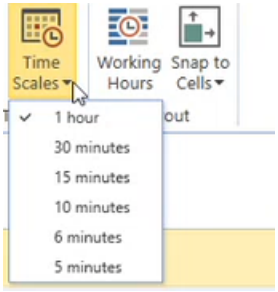


You can go backward and forward within the different views and you can view by today, work week, work, month, timeline, and agenda.

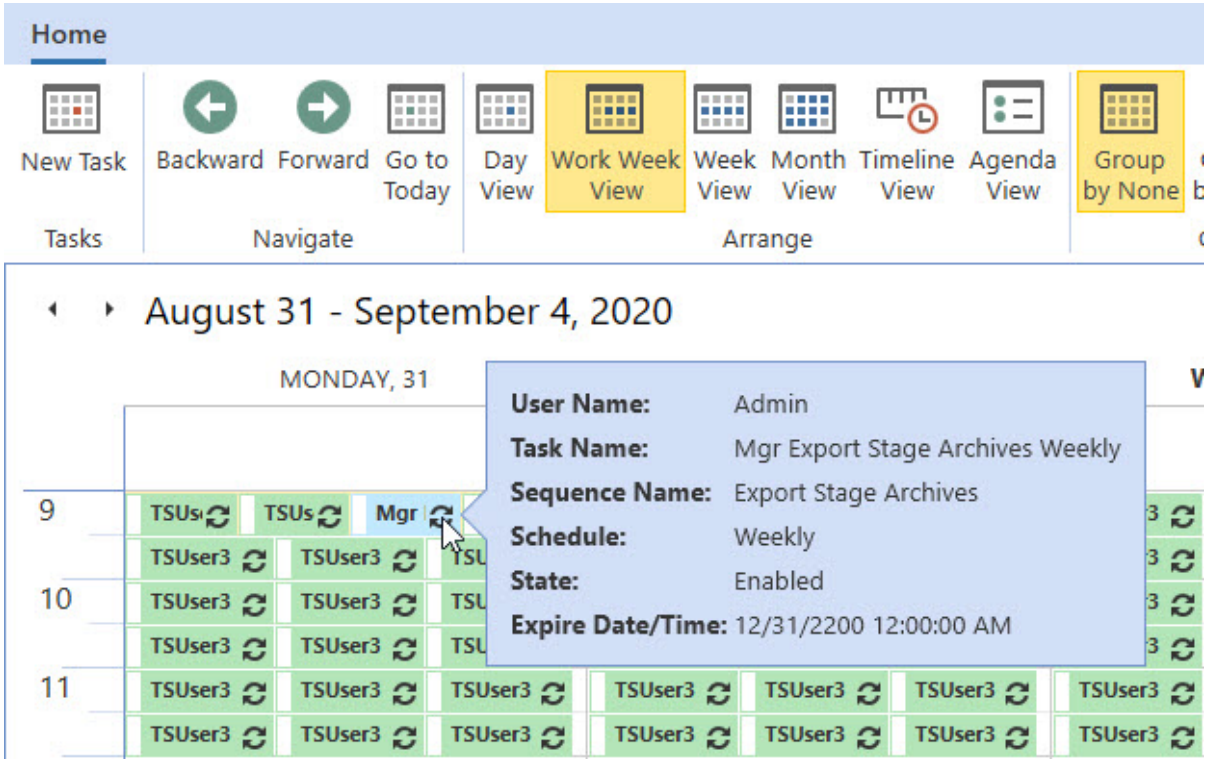


You can group by the user name, date, or no group.

When activated, you can choose to view the calendar by time scales or working hours.

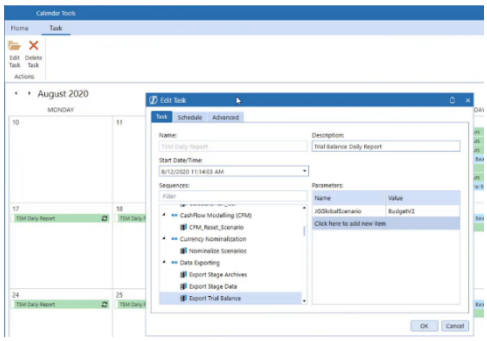


When you hover over a task on the calendar you will see the information specific to the task including User Name, Task Name, Sequence Name, Schedule, State, Expire Date/Time, if enabled, the Invalidate Date/Time.

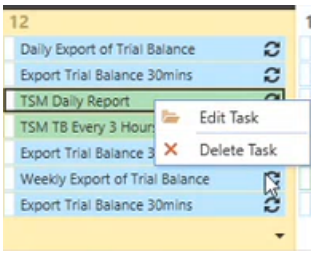


You can also create a new task, edit an existing task, or delete a task from the calendar view.

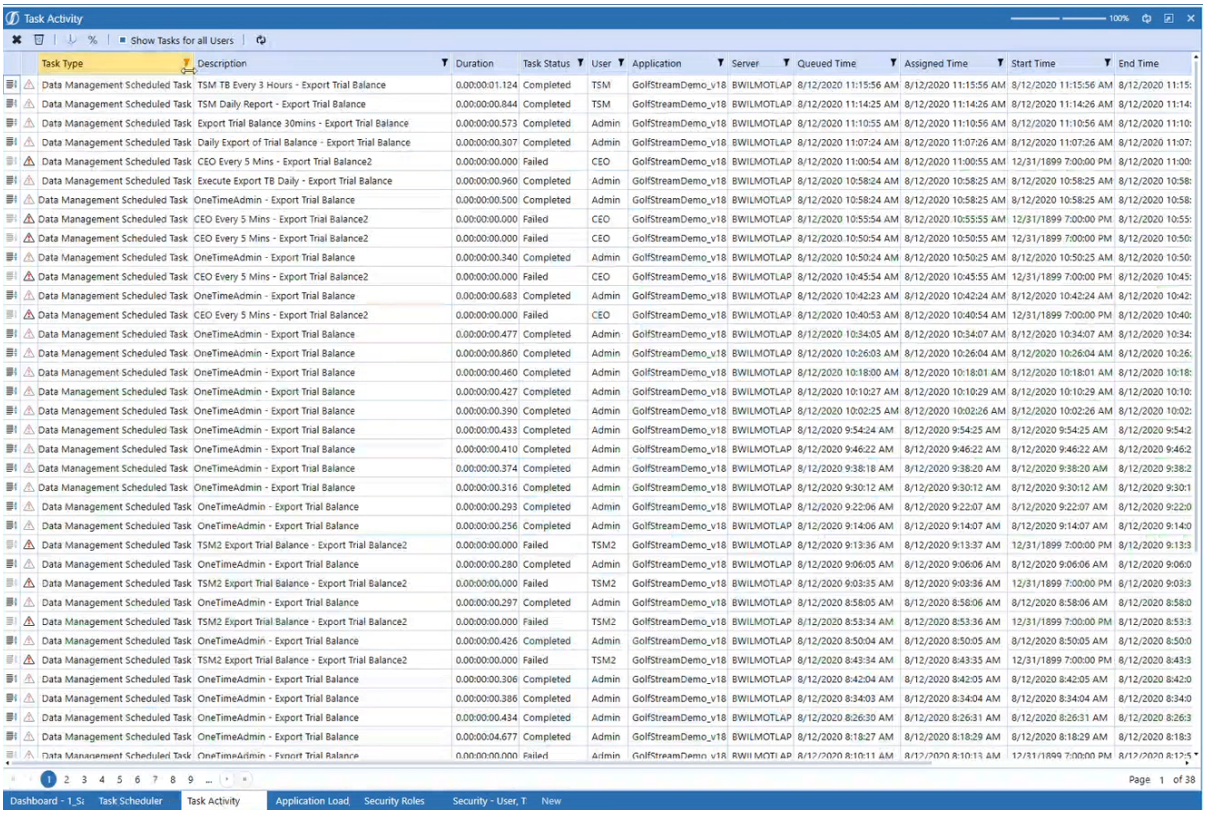
Click New Task or select a task in the calendar and click Edit Task or Delete Task.



You can also double-click on the task to edit or you can right-click to edit or delete the task.



Once the job runs, you can see the status of the job in Task Activity.

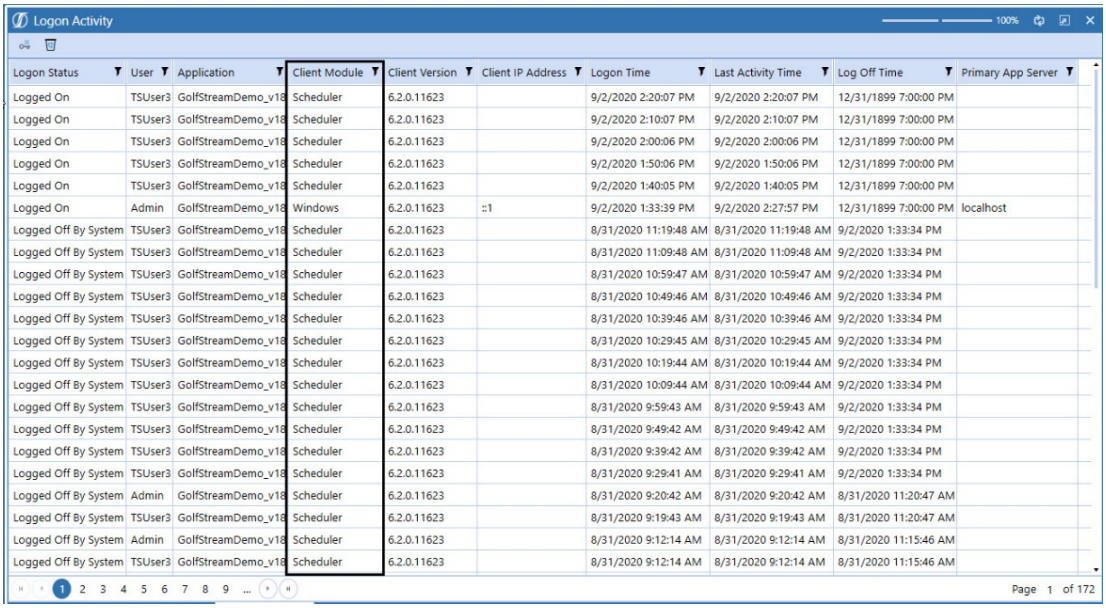


The Task Type is Data Management Scheduled Task.

The Description is the name of the task separated by a hyphen followed by the sequence.

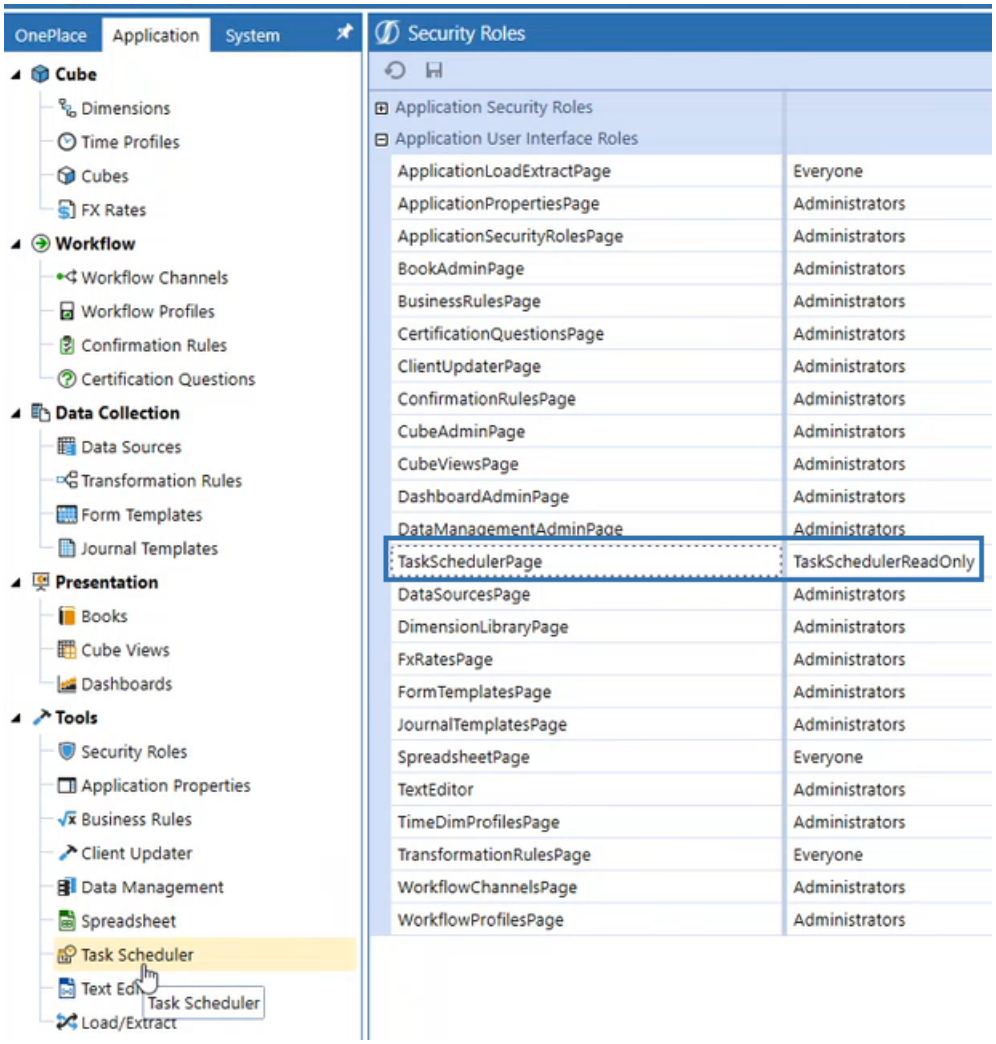
Logon Activity

1. Go to **System > Logon Activity**.
2. In the Client Module column, you can see that the log in was through Scheduler.

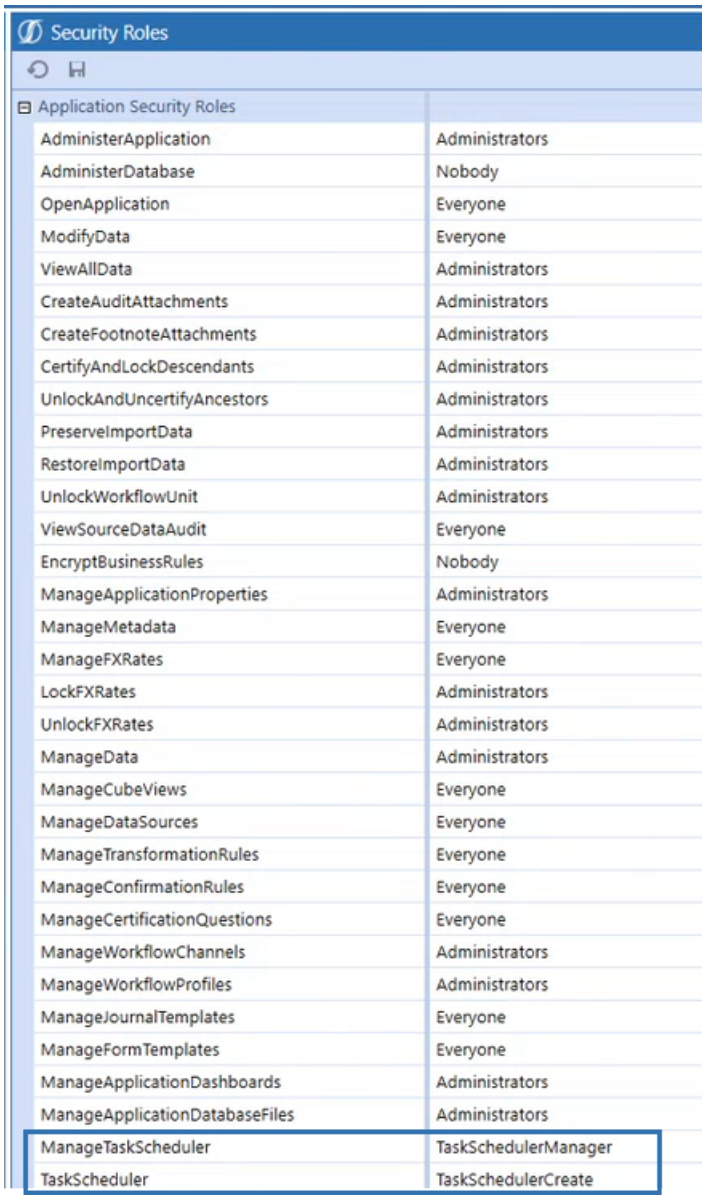


Security Roles

1. Go to **Application > Tools > Security Roles > Application User Interface Roles**.
2. You must have the TaskSchedulerPage role to see the Task Scheduler page. You can only view all user tasks with this role.

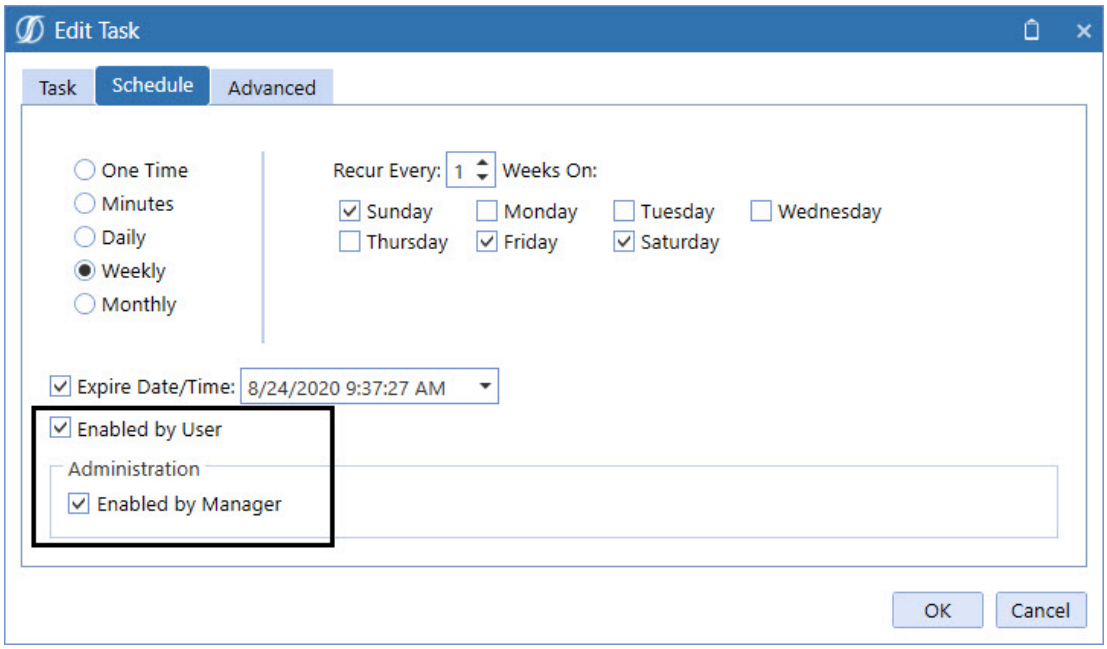


3. Go to Application Security Roles and you will see ManageTaskScheduler and TaskScheduler.



TaskScheduler allows you to create new tasks, edit your tasks, validate tasks if the setting is on. You can view all user tasks but only edit your own. You will not have access to load and extract. You cannot change the task name.

ManageTaskScheduler allows you to create your own tasks, view every task no matter who created it, edit your own tasks and other user's tasks, delete or disable your own tasks.You can load and extract. You cannot change the task name.



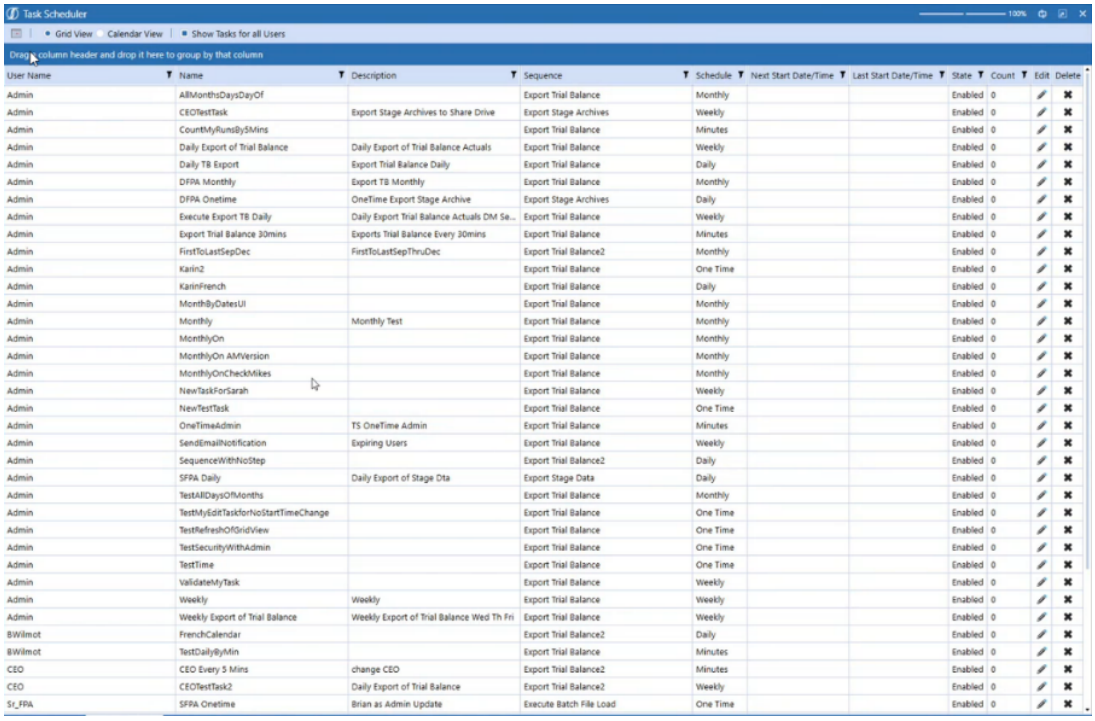
Load and Extract

If you are a manager, you can load additional task scheduler files.

Load

1. Go to **Application > Tools > Load/Extract**.
2. Select the file to load.
3. Click **Open**.
4. Click **Load**.

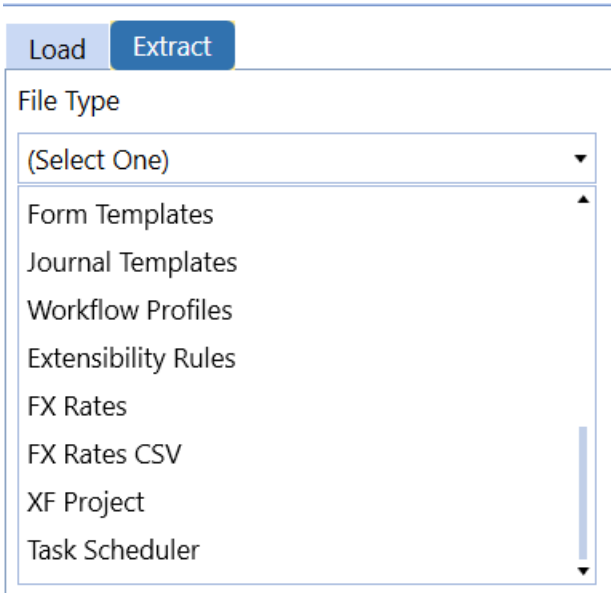
5. View in the Task Scheduler.



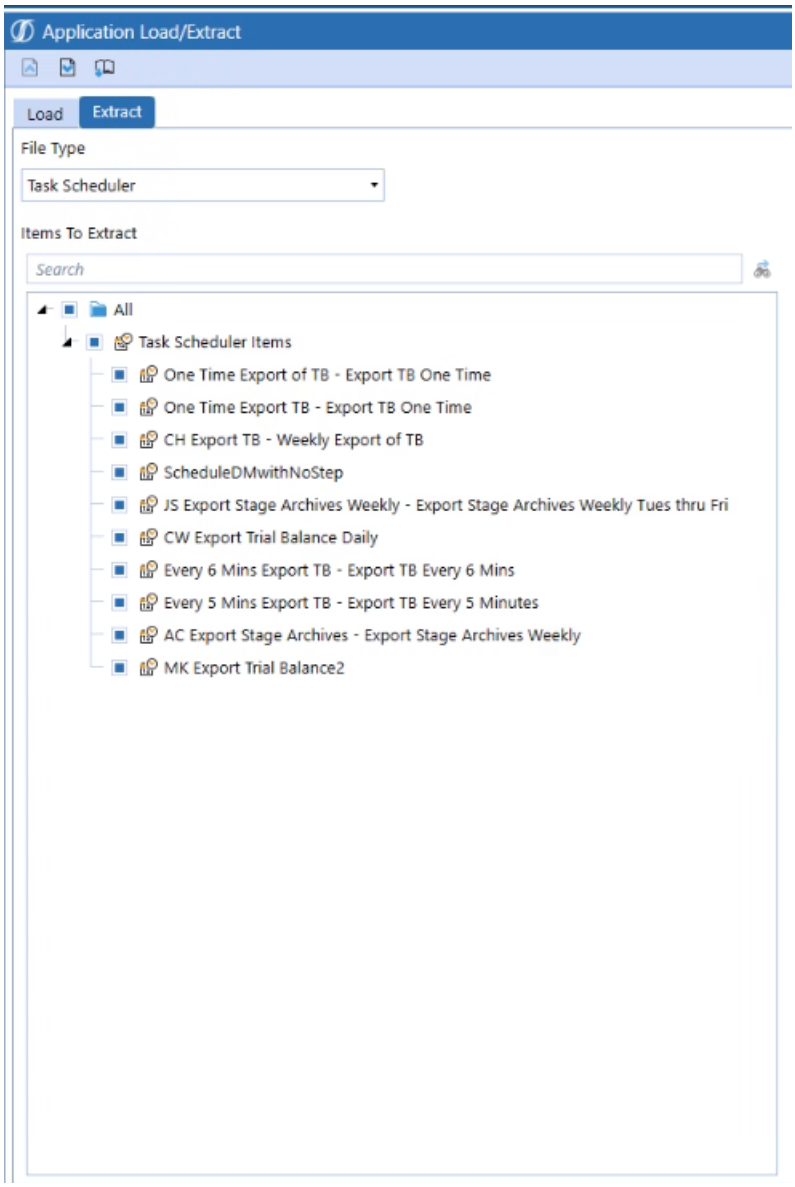
User Name	Name	Description	Sequence	Schedule	Next Start Date/Time	Last Start Date/Time	State	Count	Edit	Delete
Admin	AllMonthsDaysDayOf		Export Trial Balance	Monthly			Enabled	0		
Admin	CEOTestTask	Export Stage Archives to Share Drive	Export Stage Archives	Weekly			Enabled	0		
Admin	CountMyRunsBy5Mins		Export Trial Balance	Minutes			Enabled	0		
Admin	Daily Export of Trial Balance	Daily Export of Trial Balance Actuals	Export Trial Balance	Weekly			Enabled	0		
Admin	Daily TB Export	Export Trial Balance Daily	Export Trial Balance	Daily			Enabled	0		
Admin	DFPA Monthly	Export TB Monthly	Export Trial Balance	Monthly			Enabled	0		
Admin	DFPA Onetime	OneTime Export Stage Archive	Export Stage Archives	Daily			Enabled	0		
Admin	Execute Export TB Daily	Daily Export Trial Balance Actuals DM Se...	Export Trial Balance	Weekly			Enabled	0		
Admin	Export Trial Balance 30mins	Exports Trial Balance Every 30mins	Export Trial Balance	Minutes			Enabled	0		
Admin	FirstToLastSepDec	FirstToLastSepThruDec	Export Trial Balance2	Monthly			Enabled	0		
Admin	Karin2		Export Trial Balance	One Time			Enabled	0		
Admin	KarinFrench		Export Trial Balance	Daily			Enabled	0		
Admin	MonthByDatesUI		Export Trial Balance	Monthly			Enabled	0		
Admin	Monthly	Monthly Test	Export Trial Balance	Monthly			Enabled	0		
Admin	MonthlyOn		Export Trial Balance	Monthly			Enabled	0		
Admin	MonthlyOn AMVersion		Export Trial Balance	Monthly			Enabled	0		
Admin	MonthlyOnCheckMikes		Export Trial Balance	Monthly			Enabled	0		
Admin	NewTaskForSarah		Export Trial Balance	Weekly			Enabled	0		
Admin	NewTestTask		Export Trial Balance	One Time			Enabled	0		
Admin	OneTimeAdmin	TS OneTime Admin	Export Trial Balance	Minutes			Enabled	0		
Admin	SendEmailNotification	Expiring Users	Export Trial Balance	Weekly			Enabled	0		
Admin	SequenceWithNoStep		Export Trial Balance2	Daily			Enabled	0		
Admin	SFPA Daily	Daily Export of Stage Dta	Export Stage Data	Daily			Enabled	0		
Admin	TestAllDaysOfMonths		Export Trial Balance	Monthly			Enabled	0		
Admin	TestMyEditTaskforNoStartTimeChange		Export Trial Balance	One Time			Enabled	0		
Admin	TestRefreshOfGridView		Export Trial Balance	One Time			Enabled	0		
Admin	TestSecurityWithAdmin		Export Trial Balance	One Time			Enabled	0		
Admin	TestTime		Export Trial Balance	One Time			Enabled	0		
Admin	ValidateMyTask		Export Trial Balance	Weekly			Enabled	0		
Admin	Weekly	Weekly	Export Trial Balance	Weekly			Enabled	0		
Admin	Weekly Export of Trial Balance	Weekly Export of Trial Balance Wed Th Fri	Export Trial Balance	Weekly			Enabled	0		
BWilmot	FrenchCalendar		Export Trial Balance2	Daily			Enabled	0		
BWilmot	TestDailyByMin		Export Trial Balance	Minutes			Enabled	0		
CEO	CEO Every 5 Mins	change CEO	Export Trial Balance2	Minutes			Enabled	0		
CEO	CEOTestTask2	Daily Export of Trial Balance	Export Trial Balance2	Weekly			Enabled	0		
Sr_FPA	SFPA Onetime	Brian as Admin Update	Execute Batch File Load	One Time			Enabled	0		

Extract

1. Go to **Application > Tools > Load/Extract**.
2. Click **Extract**.
3. Select the File Type of Task Scheduler.



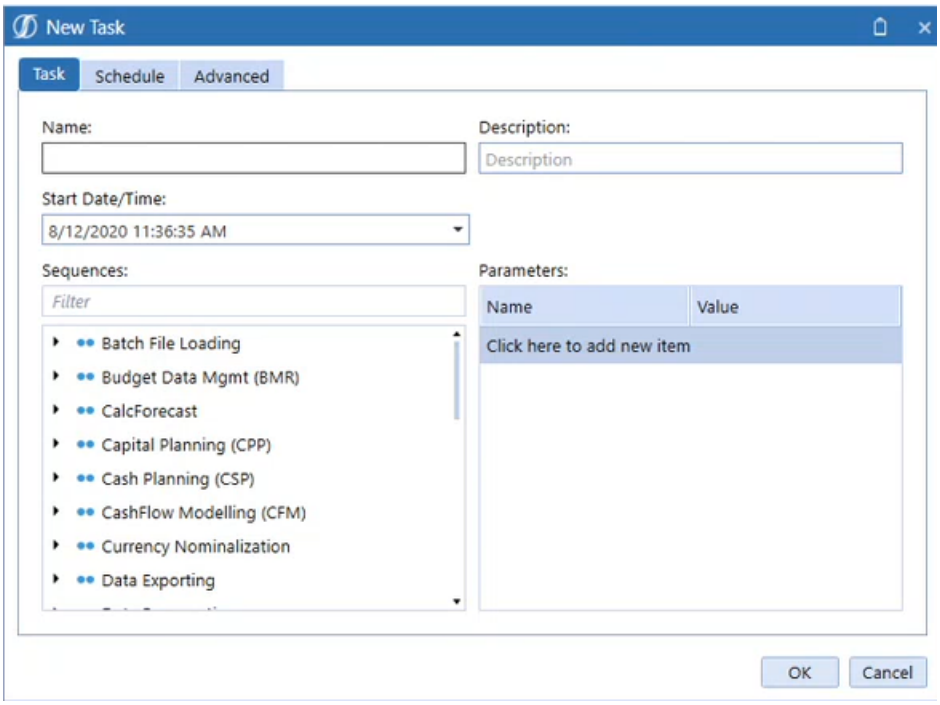
4. Select the task and click **Extract**.



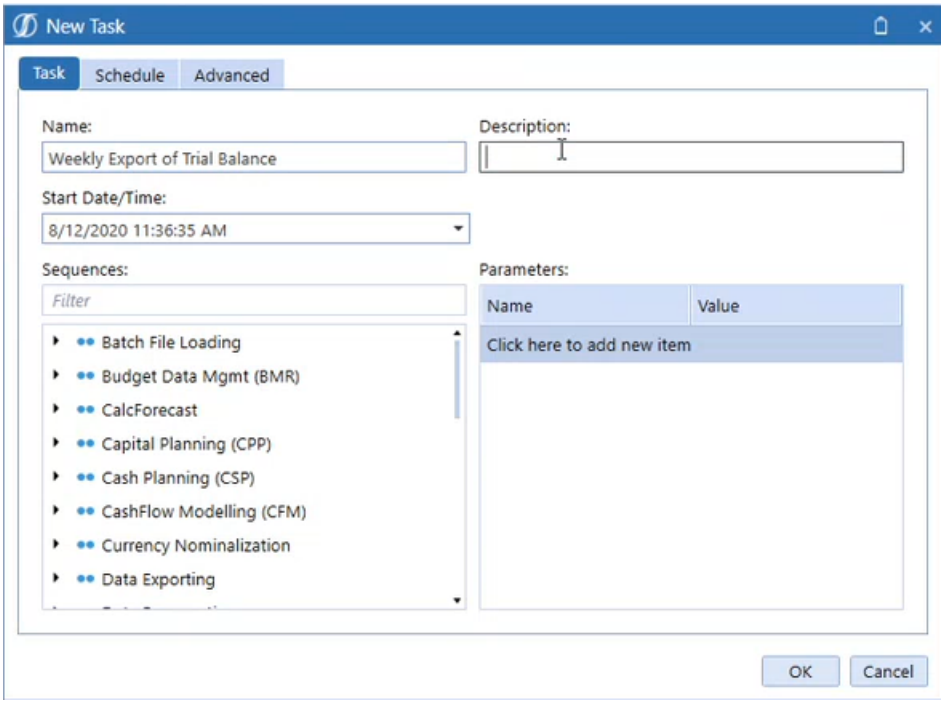
Create a New Task

> **NOTE:** When upgrading to platform release 8.4, existing tasks must be edited by opening the task and clicking "**Ok**". Doing so applies the time zone logic without recreating tasks.

1. Click Create Scheduled Task .The New Task dialog box opens.



2. Click **Task**.

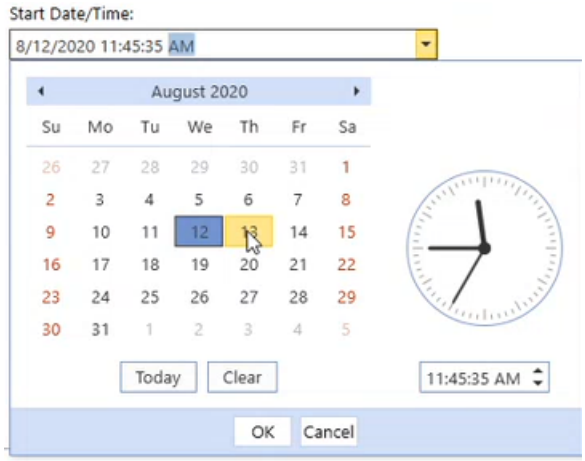


3. Enter the following details:

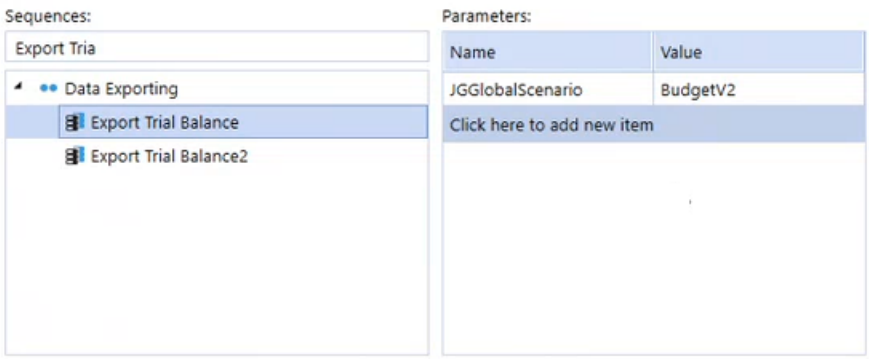
    a. Name of the task.

    b. The Description.

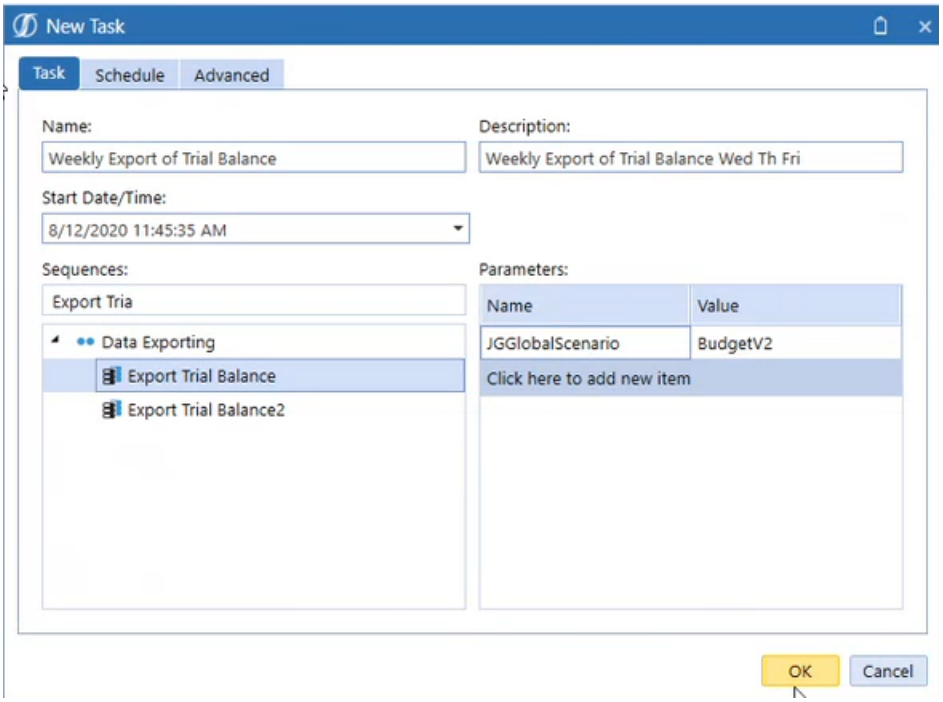
    c. Start Date/Time, which you can enter or click the drop-down to select. This should be set to the time you want the task to start running. If you don't specify a time, the value defaults to the current date and time.



d. Select the data management Sequence either by scrolling through the list or entering the sequence in the Filter field.

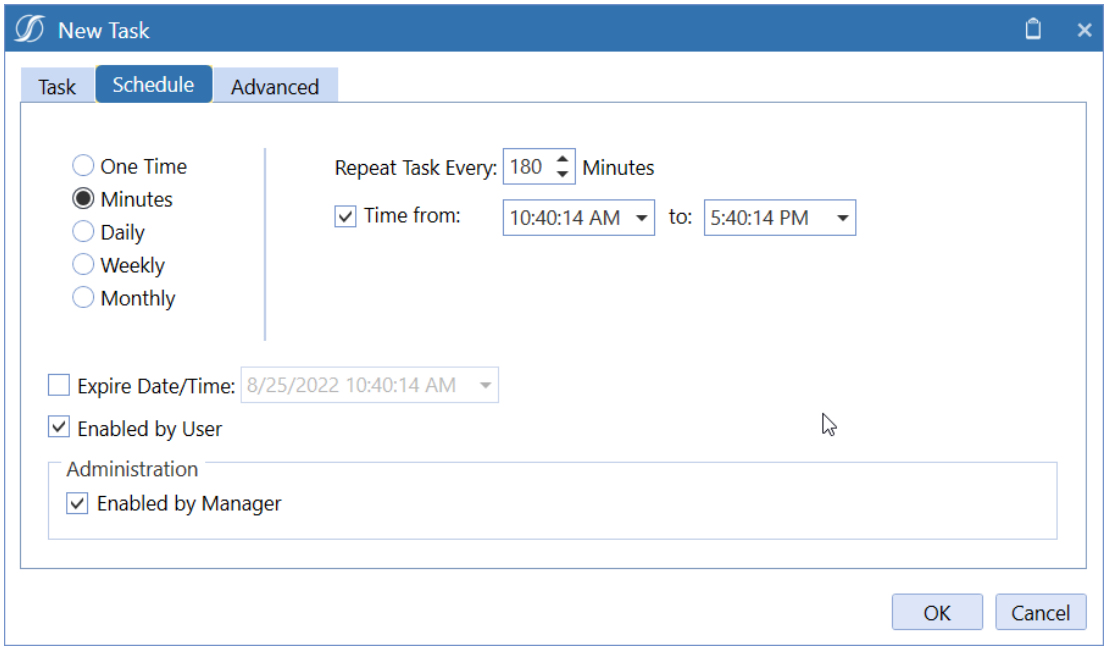


e. When you select a sequence, if there are Parameters that have been set up for the sequence, they will show. You can add additional validated parameter settings, if the parameters are not valid, the job will fail.



f. When new Weekly and Monthly recurring Tasks are scheduled, they are created in the inputting user's local time zone and saved in the UTC time zone. The default selection is Save using the current user's Time Zone.

4. Click **Schedule**.



5. Enter Schedule details:

    a. One Time triggers the task to run once based on the time in the Start/Date Time: field on the Task tab.

    b. Minutes triggers the task to run on a recurring basis from 5 – 180 minutes. You can set tasks to run during predetermined times by typing the start time in the Time From: field, then typing the stop time in the To: field. By default, the Time from: field is unchecked.

    If you create a task, for example, that is set to run every 30 minutes, starting at 2:30pm and ending at 5:30pm, the first run of the task occurs at 2:30pm and runs every 30 minutes during that timeframe.

> **NOTE:** Calendar entries are created in the Calendar view even though they may fall outside of the selected run time frame. This means, for example, that a task scheduled to run every 30 minutes between the hours of 2:00pm and 5:00pm, will display all day every 30 minutes.

c. Daily you can choose how many times it recurs.

d. Weekly you can choose how many times it recurs and the days that it runs.

e. Monthly you can choose how many times it recurs and the days that it runs.

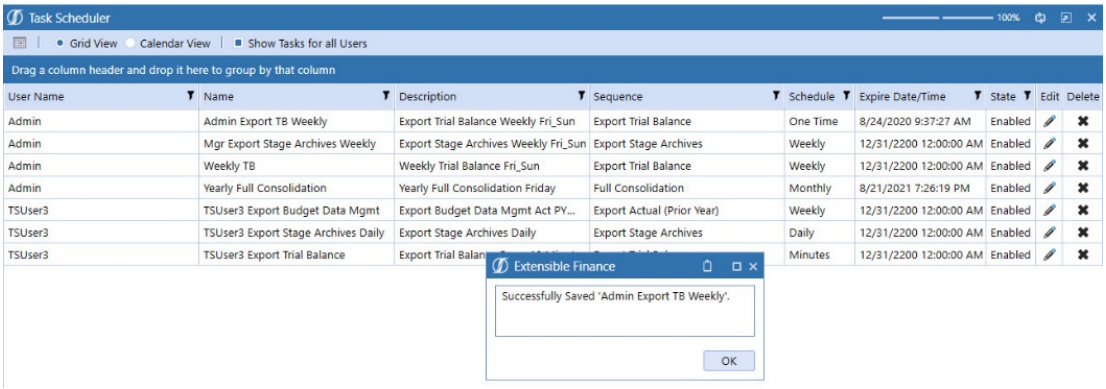
f. Expire Date/Time when and what time it should end.

g. Enabled if checked to run.

h. Administration Enabled if checked to run.

6. Click **Advanced** to set the number of times to retry a task if it fails, the maximum is three.

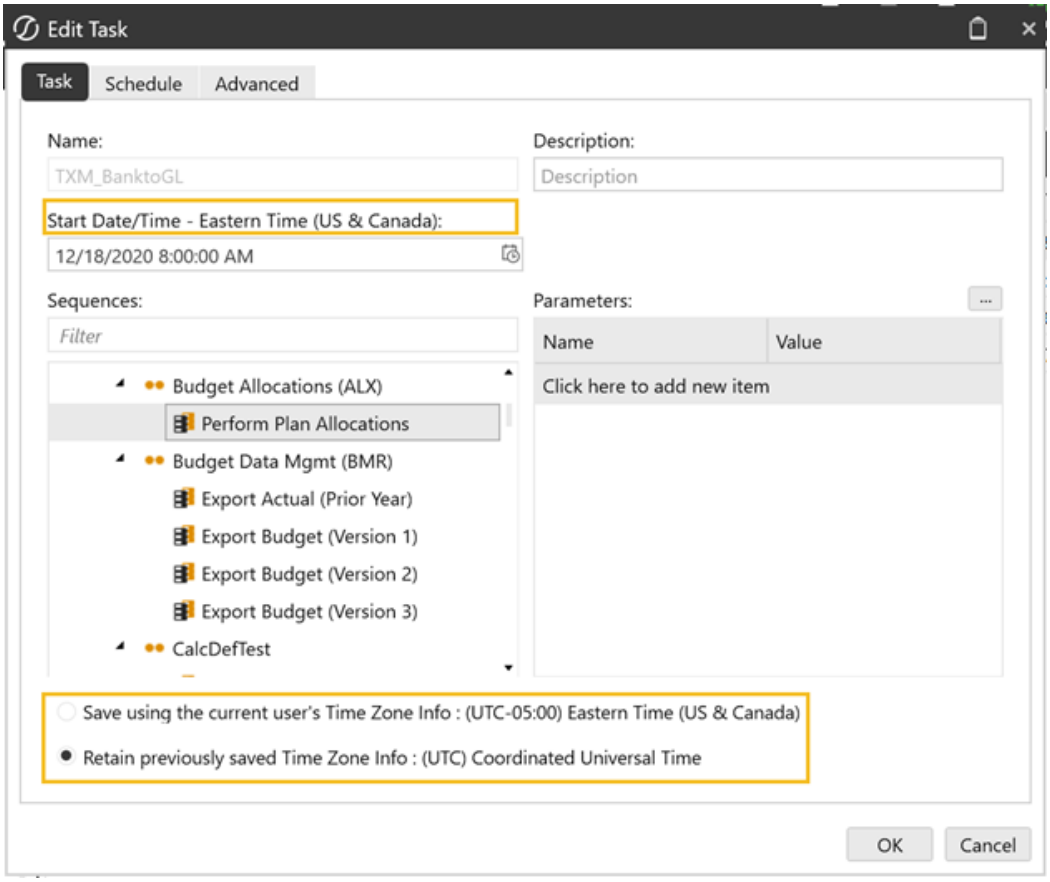
7. Click **OK** and the new task has been added to the Grid View and the Calendar View.



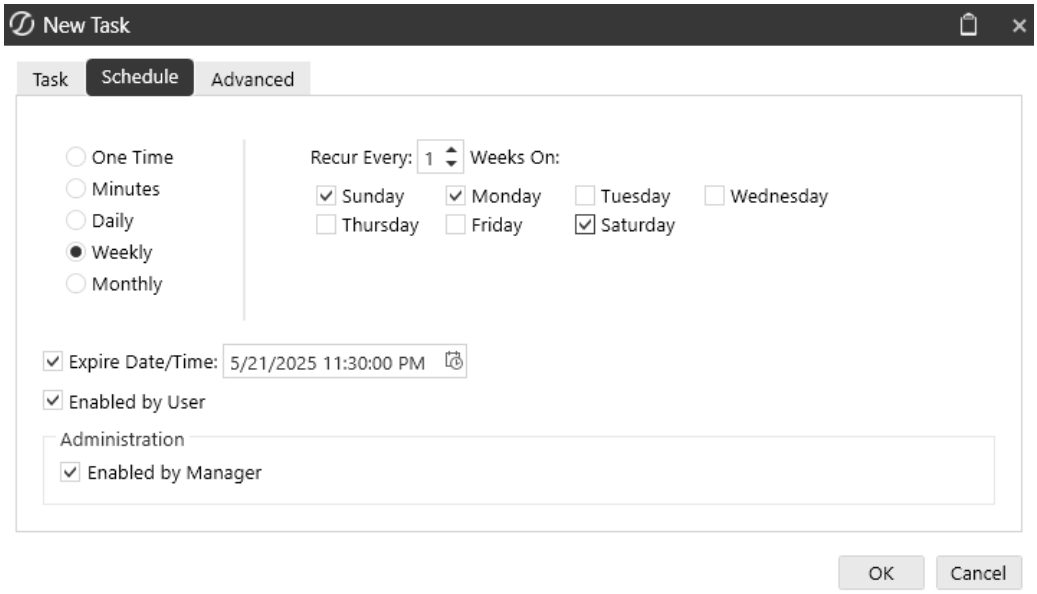
Edit Task

In the Edit task dialog, the Start Date/Time label also displays User's current time zone to provide the user the time zone . For example, Eastern Time (US & Canada).

> **NOTE:** Run logic begins with the "Start Date" then the reoccurring day of the week.


Edit Task
Task
Schedule
Advanced
Name:
TXM_BanktoGL
Description:
Description
Start Date/Time - Eastern Time (US & Canada):
12/18/2020 8:00:00 AM
Sequences:
Filter
Budget Allocations (ALX)
Perform Plan Allocations
Budget Data Mgmt (BMR)
Export Actual (Prior Year)
Export Budget (Version 1)
Export Budget (Version 2)
Export Budget (Version 3)
CalcDefTest
Parameters:
Name
Value
Click here to add new item
Save using the current user's Time Zone Info : (UTC-05:00) Eastern Time (US & Canada)
Retain previously saved Time Zone Info : (UTC) Coordinated Universal Time
OK
Cancel

Example:



Start Date/Time: 5/7/2024 11:55:00 AM (Wednesday)

Recur Every: 2 Weeks

On: Sunday, Monday, Friday

Expire Date/Time: 5/28/2024 11:30:00 AM

Expected Result:

Friday 5/9	Run
Sunday 5/11	Run
Monday 5/12	Run
Friday 5/16	Skip

Sunday 5/18	Skip
Monday 5/12	Skip
Friday 5/23	Run
Sunday 5/25	Run
Monday 5/26	Run

Text Editor

This is used to create, edit and view text documents like those created in Microsoft Word. This component only works in the OneStream Windows App version.

The OneStream Windows App Text Editor feature can perform many of the tasks that can be completed in Microsoft Word, with some limitations. These limitations include, but are not limited to:

- Mail Merge
- Application of pre-defined Styles
- The Shape Fill and Shape Outline tools
- Previous and Next Comment buttons
- Spelling
- Insert Fields
- Add Watermarks

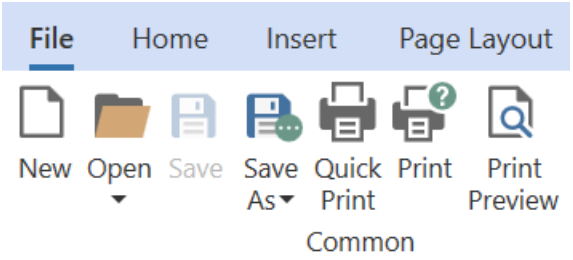
The following shortcut keys function differently in the Text Editor feature than they do in Microsoft Word. The Text Editor functionality is listed below:

- CTRL+N – Used to create a new blank document. If an existing document exists and has not been saved, using this keystroke combination will close the existing document without saving and create a new document.
- SHIFT + F3 – Used to launch the Find command
- SHIFT + F4 – Used to change case of selected word
- CTRL+SHIFT+Page Up – Used to move the cursor to the previous page
- ALT+I – Used to increase the letter/number increments within an outline or numbered list

Text Editor Ribbon

The ribbon is organized as follows:

File



New: This creates a new text editor document. When an open document has unsaved changes, a window appears asking if the current changes should be saved before the new document is created.

Open: File type to open. You can open a file from:

- **Local Folder**: Select a file to be opened from a location on the local computer/network.
- **OneStream File System**: Select a file to be opened from a location within the File Explorer.

> **NOTE:** Displayed files can also be opened from here or from OnePlace | Documents by right-clicking and selecting one of the three options – "Open in Text Editor Page"; "Open" (opens file in Compatibility mode, directly in Word if the application is found on local computer,) or "Open With…" (user specifies program).

- **Application Workspace File**: Select an Application Workspace file to be opened in Text Editor.
- **System Workspace File**: Select a System Workspace file to be opened in Text Editor.

Save: Save changes to the open file using the current file name. Only available after the file has been given a name.

Save As: File type to be saved as. You can save a file from:

- **Local Folder**: Select location on local computer/network to save a file to
- **OneStream File System**: Select location within File Explorer to save a file to
- **Application Workspace**: Select Application Workspace to save a file to.
- **System Workspace File**: Select System Workspace to save a file to.

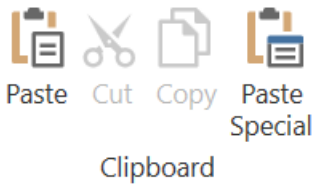
Quick Print: Send the document to the default printer without changing any printer/printing properties.

Print: Displays the Print dialog box for setting options to print the displayed document.

Print Preview: Displays the Print Preview dialog box that shows how printed document looks. The Print Preview Ribbon associated with this button remembers the setting used last that dictates if it is hidden or displayed. The next time Print Preview is opened, the Ribbon is initialized the same way it was left.

Home

Clipboard



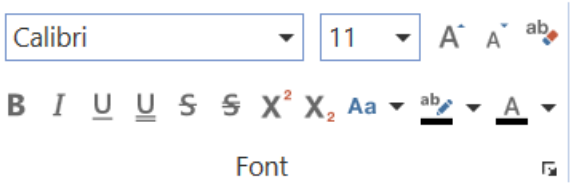
Paste: Inserts copied data into the document.

Cut: Removes and transfers selected data from the document to the clipboard for placement in a different location.

Copy: Copies selected data to the clipboard for inclusion in a different location.

Paste Special: Displays the Paste Special dialog box for additional pasting options.

Font

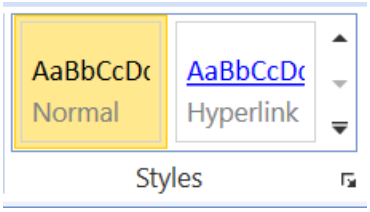


Provides options for changing the text found in the document. Examples include bold, italic, underline and text color.

Paragraph

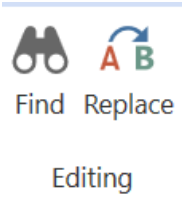
Provides options for formatting text in the document such as bullets, numbering, indenting and paragraph alignment.

Styles



Provides styling for formatting text in the document such as Normal and Hyperlink.

Editing

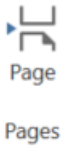


Find: Enter text to be found within the document.

Replace: Enter information to replace located text within the document.

Insert

Pages



Page: Insert a page break at the current location within the document.

Tables

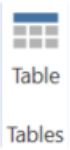


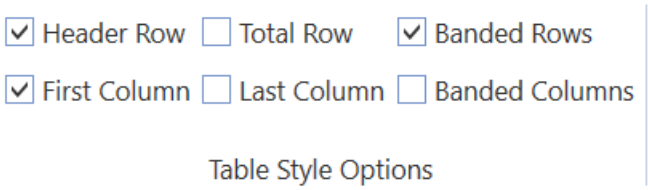
Table: Inserts a table at the current location within the document.

Design

This tab will only display when working within a table.

Table Style Options

This toolbar within the Design tab only displays when working within a table.



Provides options for formatting tables in the document such as Header Row, First Column, Total Row, and more.

Table Styles and Borders & Shading

This toolbar within the Design tab only displays when working within a table.

Table Styles: Provides options for table styles such as format titles, quotes, and other text using the gallery of styles.

Line Style and Weight: Provides options to change the style and width of the line used to draw borders.

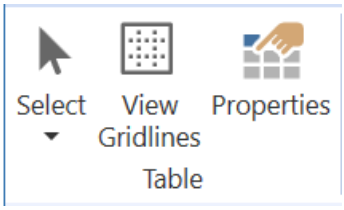
Pen Color: Used in combination with the Borders button. This button changes the color of the border lines selected when using the Borders button. Click this button first to select the border color, then use the Borders button to define where border lines should be displayed.

Borders: Used to customize the borders of the selected cells. Used in combination with the Pen Color button.

Shading: Used to add color to the background of the selected cells.

Layout

This tab will only display when working within a table.



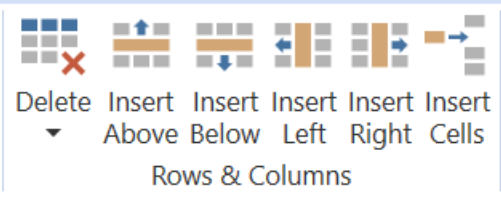
Select: Used to select the current cell, column, row or entire table.

View Gridlines: Used to show or hide the gridlines within a table. When turned on, the gridlines only appear where the display of cell borders has been turned off.

Properties: Used to display the Table Properties dialog box. Advanced formatting options such as margins, alignment, text wrapping, borders and shading can all be managed from here.

Rows & Columns

This toolbar within the Layout tab only displays when working within a table.



Delete: Used to delete cells, rows, columns or the entire table.

Insert Above: Used to add a new row directly above the selected cell.

Insert Below: Used to add a new row directly below the selected cell.

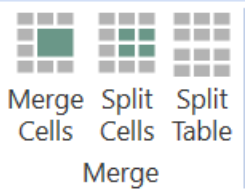
Insert Left: Used to add a new column directly to the left of the selected cell.

Insert Right: Used to add a new column directly to the right of the selected cell.

Insert Cells: Used to insert a single cell into the table. The Insert Cell dialog box appears, with options to shift cells right, shift cells down, insert an entire row or insert an entire column.

Merge

This toolbar within the Layout tab only displays when working within a table.



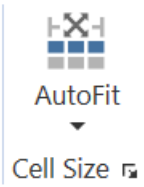
Merge Cells: Used when two or more cells are selected to join/merge them into one cell.

Split Cells: Used to split the selected cell(s) into smaller cells. The Split Cells dialog box displays and the number of new columns and rows needed can be entered.

Split Table: Used to split the table selected into two tables. The row that the selected cell(s) belongs to will become the first row of the new table that is created.

Cell Size

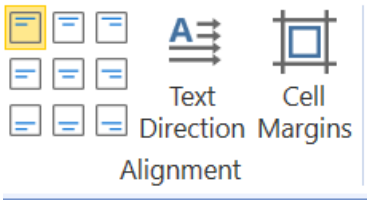
This toolbar within the Layout tab only displays when working within a table.



AutoFit: Used to automatically resize the width of the column based on the text within. Options include autofitting each cell based on the content within it, autofitting the table to take up the width of the window and setting the columns to a fixed width.

Alignment

This toolbar within the Layout tab only displays when working within a table.

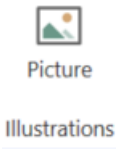


Various options are available to adjust the alignment of text with the table cells. Examples include align top left, center left, bottom left, top center, center, bottom center, top right, center right, bottom right.

Text Direction: Change the direction within the selected cells.

Cell Margins: Customize cell margins and the spacing between cells.

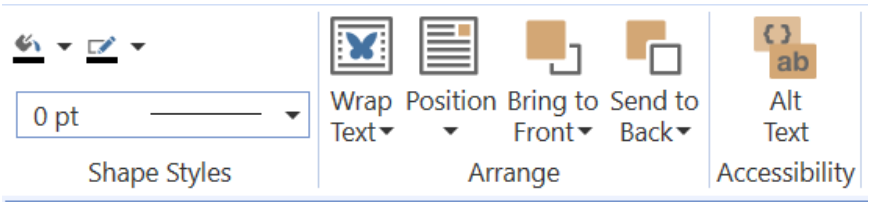
Illustrations



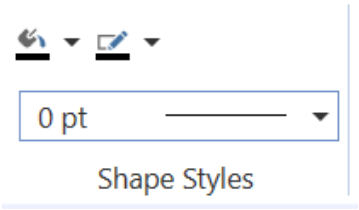
Picture: Inserts a picture file at the current location within the document. Standard picture file options are available to select from.

Format

This toolbar within the Format tab only displays when an inserted picture has been selected.



Shape Styles



Shape Styles: Lets you style the picture through shape fill, shape outline, and shape outline weight.

Arrange

This toolbar within the Format tab only displays when an inserted picture has been selected.

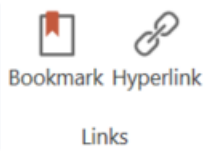
Wrap Text: Changes the way text wraps around the selected object. Seven different wrap options are available including In Line with Text, Square, Tight, Through, Top and Bottom, Behind Text and In Front of Text.

Position: Positions the selected object on the page. Text is automatically set to wrap around the object.

Bring to Front: Brings the selected object forward so that it is hidden by fewer objects in front of it. Three options are available including Bring Forward, Bring to Front and Bring in Front of Text.

Send to Back: Sends the selected object backward so that it is hidden by the objects in front of it. Three options are available including Send Backward, Send to Back and Send Behind Text.

Links

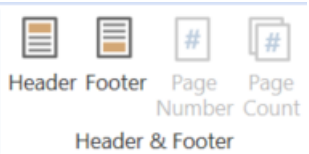


Bookmark: Creates a bookmark for selected text and assigns a name to that specific area of the document. Hyperlinks can be made to move directly to that location.

Hyperlink: Creates a link to a webpage, a file, an application, an email address or a place in the same document.

> **NOTE:** To send an email enter the following in the address field – mailto:emailusername@domainhostname. For example, mailto:jdoe@onestreamsoftware.com.

Header & Footer



Header: Insert a Header into the document or go to the Header section if a Header already exists within the document.

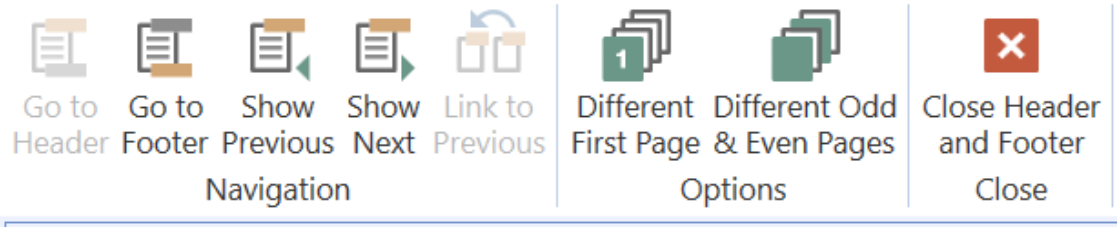
Footer: Insert a Footer into the document or go to the Footer section if a Footer already exists within the document.

Page Number: Inserts the current page number, wherever the cursor is located within the Header/Footer areas of the document.

Page Count: Inserts the total number of pages in the document, wherever the cursor is located within the Header/Footer areas of the document.

Navigation

This toolbar within the Design tab only displays when working within the Header/Footer areas of the document.



Go to Header/Go to Footer: Activates the Header/Footer section on the page so it can be edited.

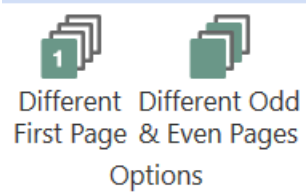
Show Previous: If the document has been broken into sections, this navigates to the previous section's Header/Footer.

Show Next: If the document has been broken into sections, this navigates to the next section's Header/Footer.

Link to Previous: Creates a link to the previous section so the Header/Footer in this section contains the same content as the previous section.

Options

This toolbar within the Design tab only displays when working within the Header/Footer areas of the document.

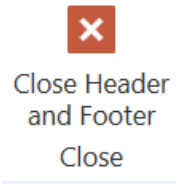


Different First Page: Used to insert a unique Header/Footer for the first page only of the document.

Different Odd & Even Pages: Used to insert different Header/Footers on even and odd pages.

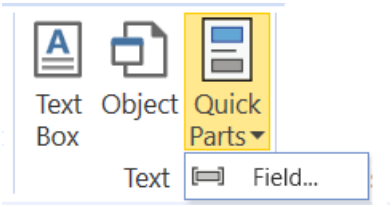
Close

This toolbar within the Design tab is only displayed when working within the Header/Footer areas of the document.



Close Header and Footer: Used to close out of the Header/Footer Tools and return to the document.

Text



Text Box: Inserts a text box into the document.

Object: Inserts an embedded object, such as another Word Document or an Excel Chart.

Quick Parts: Insert document variable syntax for content and formatting strings.

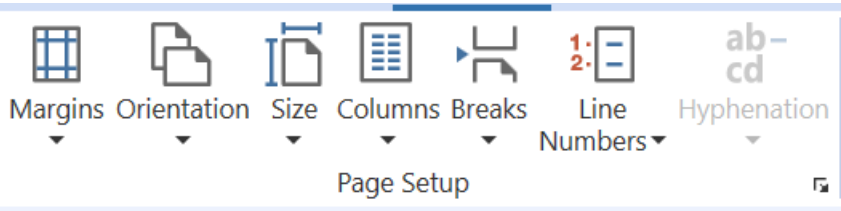
Field: The field to insert document variables and arguments.

Symbols

Symbol: Inserts standard characters and those not found on the keyboard into the document.

Page Layout

Page Setup



Margins: Used to set pre-defined or custom margin sizes for the entire document, or the current section.

Orientation: Used to change the pages between landscape and portrait layouts.

Size: Used to set the paper size for the current section.

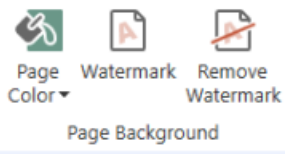
Columns: Used to split the text into two or more columns.

Breaks: Used to insert page, column or section breaks (Next, Continuous, Even or Odd).

Line Numbers: Used to insert line numbers in the margins on each line of the document. Options include restarting numbering on each page, each section, suppressing numbers for current paragraph and custom settings.

Page Setup: Opens the Page Setup dialog box where margins, orientation, paper size, headers/footers and header/footer placement can all bet set/edited.

Page Background



Page Color: Used to choose a background color for all the pages in the document.

Watermark: Insert and configure text and image watermarks within Text Editor documents. See Text Editor Watermarks.

Remove Watermark: Remove all watermarks from the Text Editor document.

References

Table of Contents

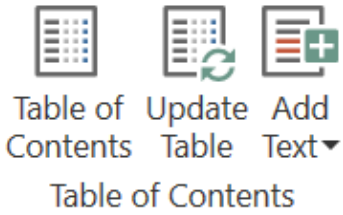
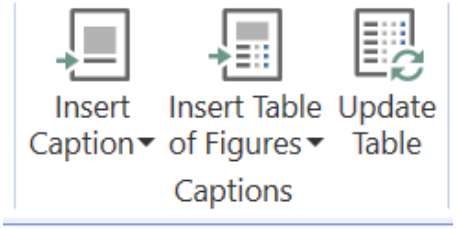


Table of Contents: Used to create a Table of Contents for the document.

Update Table: Used to update the Table of Contents so that all entries reference the correct page number.

Add Text: Used to add the current paragraph into the Table of Contents.

Captions



Insert Caption: Used to add a caption to a picture or another image. It is used to describe the object associated with it and appears below the object. Captions can be created for the following items:

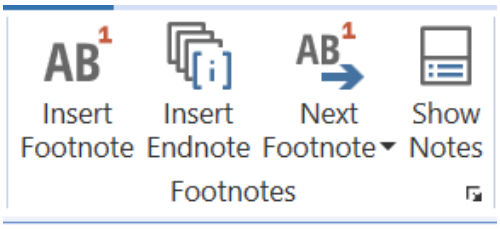
- **Figures Caption**: Used to add a caption to a figure or picture within the document.
- **Tables Caption**: Used to add a caption to a table within the document.
- **Equations Caption**: This option is not used or supported because equations cannot be created in the OneStream Windows App Text Editor.

Insert Table of Figures: Similar to the Table of Contents, this is used to insert a table that lists all the figures or tables that are contained in the document. The following types of tables can be created.

- **Table of Figures**: Includes a list of all the Figures in the document that have captions associated with them.
- **Table of Tables**: Includes a list of all the Tables in the document that have captions associated with them.
- **Table of Equations**: This option is not used or supported because equations cannot be created in the OneStream Windows App version of Text Editor.

Update Table: Used to update the Table of Figures to include all entries within the document.

Footnotes



Insert Footnote: Add a footnote.

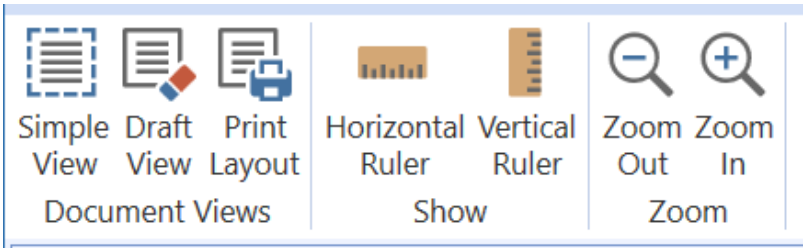
Insert Endnote: Add a endnote.

Next Footnotes: Jump to next footnote.

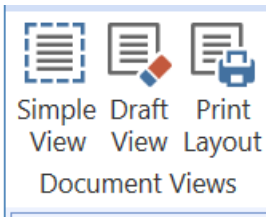
- **Next Footnote**: Jump to next footnote.
- **Previous Footnote**: Jump to previous footnote.
- **Next Endnote**: Jump to next endnote.
- **Previous Endnote**: Jump to previous endnote.

Show Notes: Jump to the footnote or endnote.

View



Document Views

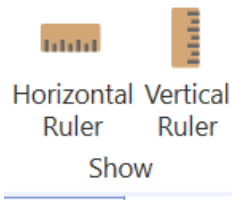


Simple View: Used to show the document as a simple memo. Page Layout features are ignored in this view.

Draft View: Used to view the document as a draft for quicker editing. Certain features, such as Headers/Footers are not visible when using this view.

Print Layout: Used to show how the document will look on the printed page.

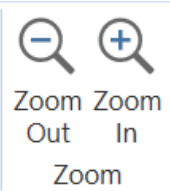
Show



Horizontal Ruler: Used to display the Horizontal Ruler, which is used to measure and line up objects in the document.

Vertical Ruler: Used to display the Vertical Ruler, which is used to measure and line up objects in the document.

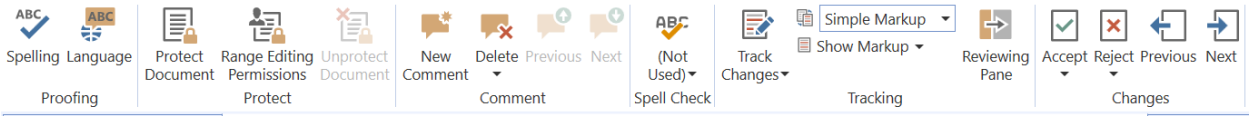
Zoom



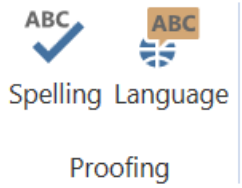
Zoom Out: Used to change the view to see more of the page at a reduced size.

Zoom In: Used to change the view to get a close-up view of the document.

Review



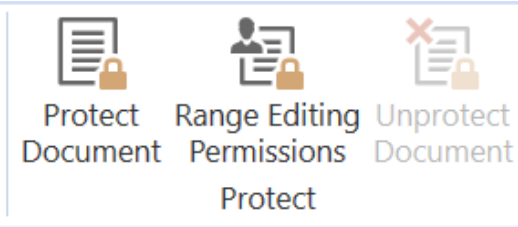
Proofing



Spelling: Used to check the spelling of text in the document.

Language: To set the language used to check the spelling and grammar of the selected text.

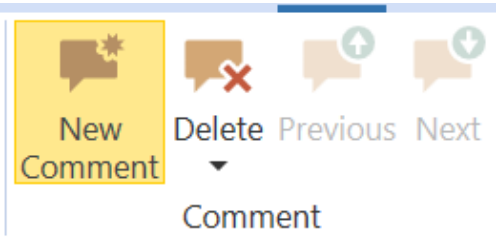
Protect



Protect Document: Used to add a password to the document, so that when protected, only users who know the password can edit the document or add comments. There are options for both of these, and only one can be implemented at a time.

Unprotect Document: Used to unprotect a document that has been locked for editing. Click the button and provide the previously created password to unprotect the document.

Comment



New Comment: Used to add comments to the document. Automatically numbers the comments as they are added to the document. The comment located at the top of the document is numbered 1 and additional comments are numbered sequentially down the document based on their position. If a comment is added or deleted, remaining comments in the document are automatically renumbered based on their location within the document.

Delete: Used to delete comments in the document. Options include the following:

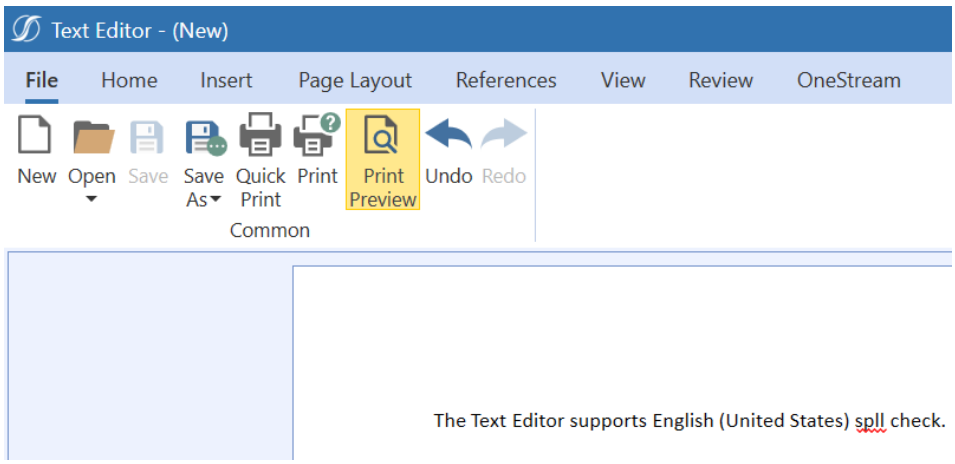
- **Delete Comment**: Used to delete the selected comment only.
- **Delete All Comments Shown**: This option is not supported in the Text Editor for OneStream Windows App.
- **Delete All Comments**: Used to delete all comments in the document.

Spell Checking

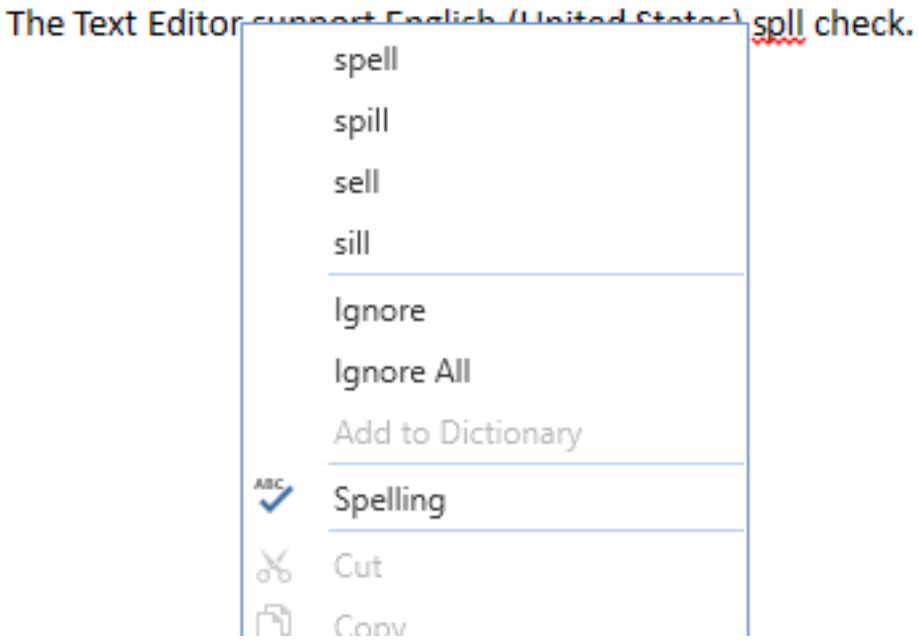
A text spell check feature is available only when using the Windows Application. This feature is set as a default to be inactive. To enable the Spell Check feature, users must have access to the Application / Tools / Text Editor. In the Text Editor Tool, the Review ribbon will allow the user to activate Spell Check using the Spell Check button.

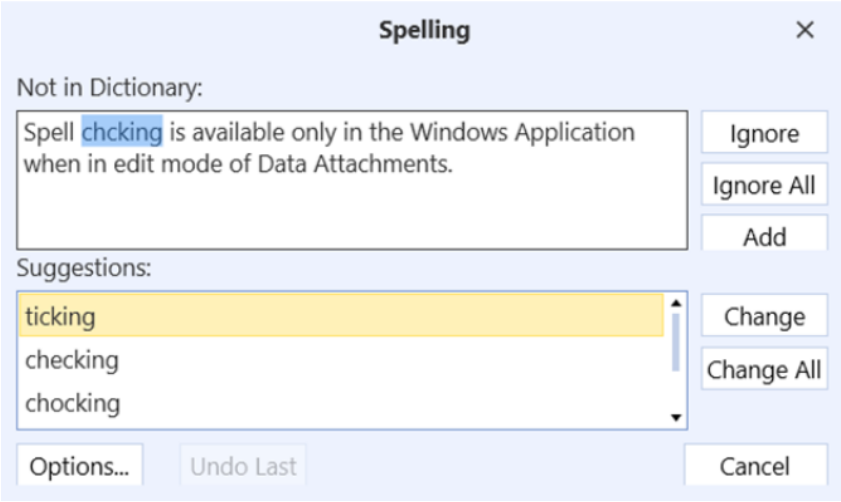
The Spell Check feature is enabled for English Culture only. The culture is determined by each user's culture assigned in OneStream User Security. The culture is assigned to the OneStream application on the Application Server Configuration Utility as "en-US". Users with cultures other than English (United States) will not have Spell Check available.

Spell Check in Text Editor: The Spell Check will be active as the user enters text, identifying any errors.

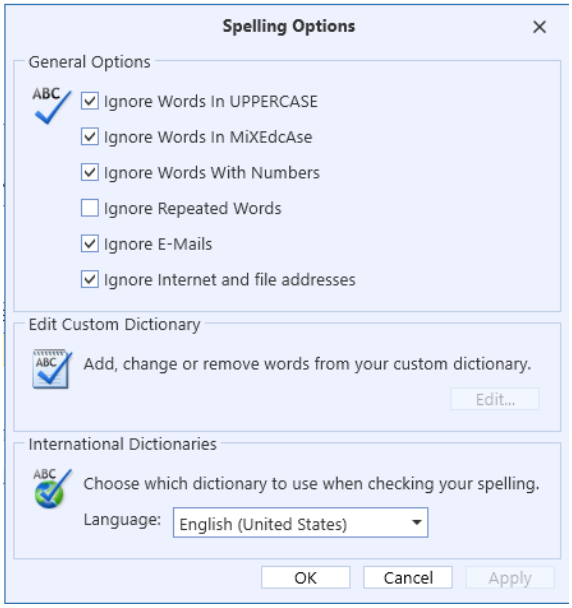


To access the Spell Check options, double-click to select the error, then a right-click will expose the Spell Check menu choices. Choosing the “Ignore” option will only be retained for the current session. Closing and re-opening to the edit mode will re-check any previously ignored items.

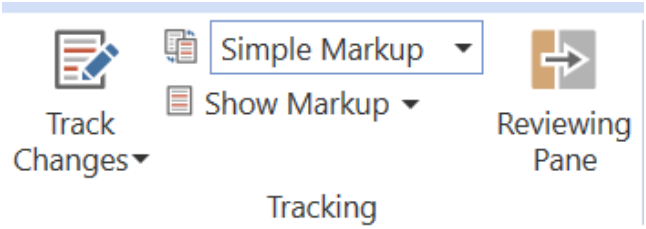




The Options button will allow the user to modify the Spell Check behavior within the current task session. The settings are not persisted as user preference.



Tracking



Track Changes: Keep track of changes made to this document. Options include the following:

- **Track Changes**: Keep track of changes made to this document.
- **Lock Tracking**: Use a password to keep others from turning on and off Track Changes.

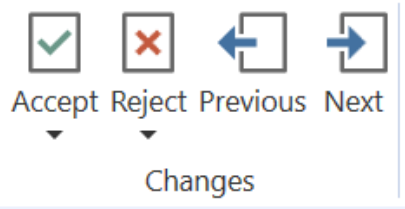
Display for Review: Choose how you'd like to see changes in the document. Options include: Simple Markup, All Markup, and No Markup.

Show Markup: Choose what types of markup to show in your document. Options include the following:

- **View Comments**: Used to show Comments that have been added to the document. They will be displayed in a column on the right side of the document.
- **Reviewers**: Select which reviewers' comments to highlight in the document.

Reviewing Pane: Used to show/hide Comments within the document in a separate window. The window opens on the left side of the document and can be used to select comments.

Changes



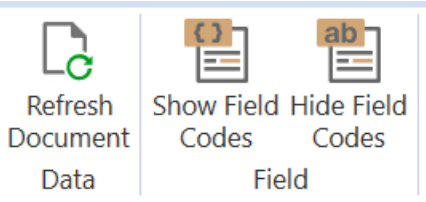
Accept: Keep this change and move on to the next one.

Reject: Undo this change and move on to the next one.

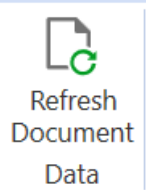
Previous: Jump to the previous tracked change.

Next: Jump to the next tracked change.

OneStream



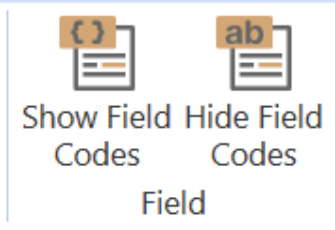
Data



Refresh Document: Allows you to save and refresh your changes after making edits to the document.

> **NOTE:** This is useful for Document Variables as the Refresh Document button will "re-process" variables.

Field



Show Field Codes: Allows you to show field codes for document variables of the embedded content.

Hide Field Codes: Allows you to hide field codes for document variables of the embedded content.

Text Editor Watermarks

Watermarks help you to protect a company's assets, such as documents, from unauthorized use. In OneStream, you can create watermarks using images or text.

Create an Image Watermark

1. In the Text Editor, click the **Page Layout** tab and then click **Watermark**. The Watermark dialog box opens.
2. Select the **Picture watermark** option.
3. Browse to an image on your local drive and select it.
4. Select the **Scale** option and adjust the image size by percentage, if needed.
5. Select the **Washout** checkbox to apply a washout effect to the image. Deselect to keep the original colors.
6. In the **Apply to** drop-down menu, select how to apply the watermark:
    - **Whole document**: Applies the watermark to every page in the document
    - **Section**: Applies the watermark to the section in which the cursor is currently positioned

    > **NOTE:** Sections are created using the Breaks menu on the Page Layout tab.

7. Click **Apply** or **OK**.

Create a Text Watermark

1. In the Text Editor, click the **Page Layout** tab and then click **Watermark**. The Watermark dialog box opens.
2. Select the **Text watermark** option

3. Type the watermark text or select from the pre-defined options.
4. Apply **Font**, **Font Size**, and **Color** options as needed.
5. Select the **Semitransparent** checkbox to apply a semitransparent effect to the text. Deselect to keep the original opacity.
6. Select **Diagonal** or **Horizontal** layout.
7. In the **Apply to** drop-down menu, select how to apply the watermark:
    - **Whole document**: Applies the watermark to every page in the document
    - **Section**: Applies the watermark to the section in which the cursor is currently positioned

    > **NOTE:** Sections are created using the Breaks menu on the Page Layout tab.

8. Click **Apply** or **OK**.

Modify a Watermark

You can modify a watermark by clicking the Watermark button. The Watermark dialog box opens where you can make modifications as needed.

Remove a Watermark

1. In the Text Editor, click the **Page Layout** tab and then click **Remove Watermark**. The Watermark dialog box opens. This removes all watermarks from the document.
2. Alternatively, open the Watermark dialog box by clicking the **Watermark** button.
    a. In the Watermark dialog box, select the **No Watermark** option.
    b. Click **Apply** or **OK**.

Load/Extract Application Artifacts

You can import and export sections of the Application using an XML Format.

> **TIP:** Only Administrators have access to this portion of the tool.

Extract

Choose an item from the drop-down list, click the Extract icon to start the extract process, then name the output file.

Load

After browsing to the file, click the Load icon to initiate the process.

Extract and Edit

This option is available for all extracts and allows the ability for the end user to edit the XML file as needed.

These are the list of items that can be extracted:

Application Zip File (All except for Data and FX)
This option will export all the Application structures except for data and the FX rates. After clicking on the extract icon, the end user will be presented with a pop-up box to save it in a zipped format. After clicking yes, the end user can then name his/her file. This is the most complete option in the list and can be used to create a copy of the Application.

Application Security Roles
This covers all the security roles found under Application | Tools | Security Roles. The screen will display Items To Extract and by default, All is chosen. The defined roles will be exported accordingly.

Application Properties

This covers all the Application Properties found under Application | Tools | Application Properties. After choosing this option, the screen will display Items To Extract and by default All is chosen.

Workflow Channels

This covers all the Workflow Channels found under Application | Workflow | Workflow Channels. The screen will display Items To Extract and by default All is chosen.

Metadata

This has multiple sections such as Business Rules, Time Dimension Profiles, Dimensions, and Cubes. The metadata can be found under Application | Cube | Dimension Library. OneStream can search for Dimension Member and Relationship changes by username and time stamp. For more details on the Find Modified Items feature, see Extracting and Loading Dimensions in Implementing Security.

Cube Views

This has multiple sections such as Groups and Profiles which go together. Cube Views can be found under Application | Presentation | Cube Views. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Data Management

This has multiple sections such as Groups and Profiles which go together. Data Management can be found under Application | Tools | Data Management. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Application Workspaces

This has multiple sections such as Maintenance Units and Profiles. Under the Groups, Dashboard Components, Dashboard Adapters, and Dashboard Parameters go together. Workspaces can be found at Application | Presentations | Workspaces. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Confirmation Rules

This has multiple sections such as Groups and Profiles which go together. Confirmation Rules can be found under Application | Workflow | Confirmation Rules. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Certification Questions

This has multiple sections such as Groups and Profiles which go together. Certification Questions can be found under Application | Workflow | Certification Questions. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Transformation Rules

This has multiple sections such as Business Rules, Groups, and Profiles which go together. Transformation Rules can be found under Application | Data Collection| Transformation Rules. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Data Sources

This has multiple sections such as Business Rules and Data Sources which go together. Data Sources can be found under Application | Data Collection | Data Sources. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Form Templates

This has multiple sections such as Groups and Profiles which go together. Form Templates can be found under Application | Data Collection | Form Templates. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Journal Templates

This has multiple sections such as Groups and Profiles which go together. Journal Templates can be found under Application | Data Collection | Journal Templates. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

Workflow Profiles

This has multiple sections such as Workflow Profiles and Workflow Profile Templates. Workflow Profiles can be found under Application | Workflow | Workflow Profiles. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off. When loading Workflow Profiles via an XML file extracted previously or from a different Application, the Load process clears old property settings if they are not specified in the loaded XML file. This approach ensures that any property edits made to the Workflow Profile upon extracting it are honored when the same XML file is re-loaded.

Extensibility Rules

This only exports the Extensibility Rules in the Business Rules section; all others are exported with the object to which they are tied. (e.g., if there is an Account description rule under the Parser section, it will be exported with the Data Source.) Extensibility Rules can be found under Application | Tools | Business Rules | Extensibility Rules. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

FX Rates

Rates have their own Cube. This has multiple sections such as All FX Rate Types and All Time Periods. FX Rates can be found under Application | Cube | FX Rates. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section, or turn the sections on or off.

FX Rate CSV

This is a CSV extract of the application FX Rates which contains FxRateType, Time, SourceCurrency, DestCurrency, Amount, HasData. FX Rates can be found under Application | Cube | FX Rates.

User Authentication and Single Sign-On

When you log in to an application, your username and password are authenticated to confirm that you are a valid OneStream user.

If you are in a OneStream-hosted environment, see the *Identity and Access Management Guide* for information about authentication with OneStream IdentityServer.

If you are in a self-hosted environment, administrators can configure an environment to use native authentication, one external identity provider, or both native authentication and one external identity provider.

The following external identity providers are supported:

- Microsoft Active Directory (MSAD)
- Lightweight Directory Access Protocol (LDAP)
- Three OpenID Connect (OIDC) identity providers:
    - Azure Active Directory (Azure AD [Microsoft Entra ID])
    - Okta
    - PingFederate
- SAML 2.0 identity providers (for example, Okta, PingFederate, Active Directory Federation Services [ADFS], and Salesforce)

For instructions on configuring authentication in a self-hosted environment, see the *Installation and Configuration Guide*.

How Users are Configured for Authentication

When an administrator creates users in **System** > **Security**, they specify authentication settings.

- For native authentication:
  - **External Authentication Provider**: In the drop-down menu, select **(Not Used)**.
  - **External Provider User Name**: Leave this field blank.
- For an external identity provider:
  - **External Authentication Provider**: In the drop-down menu, select the external identity provider.
  - **External Provider User Name**: Enter the user name configured in the external identity provider in this field.

See About Managing Users and Groups.

If you are in a OneStream-hosted environment, see the *Identity and Access Management Guide* for information about authentication with OneStream IdentityServer.

For instructions on configuring authentication in a self-hosted environment, see the *Installation and Configuration Guide*.

About Managing Users and Groups

See the following topics for help defining and managing users and groups:

- Ways to add Users and Groups to an Application.
- Creating and Managing Users .
- Creating and Managing Groups.

For information about how group-based assignment to system security roles determines a user's access to artifacts and capabilities, see Manage System Security.

Ways to add Users and Groups to an Application

You can add users and groups to an application in three ways:

- Define them manually in System Security. See Creating and Managing Groups and Creating and Managing Users .
- Load them in a bulk import using an XML load file that contains user and group properties and parameters. We suggest building this file using sample security Excel template provided with the Sample Templates OneStream Solution. See Creating and Managing Users and Creating and Managing Groups.
- Use APIs. See Managing Users and Groups Using BRApi Functions.

Creating and Managing Users

By default, only an Administrator can perform the tasks described in the following topics. An Administrator can enable other users to perform these tasks by giving them specific system security roles.

In this topic:

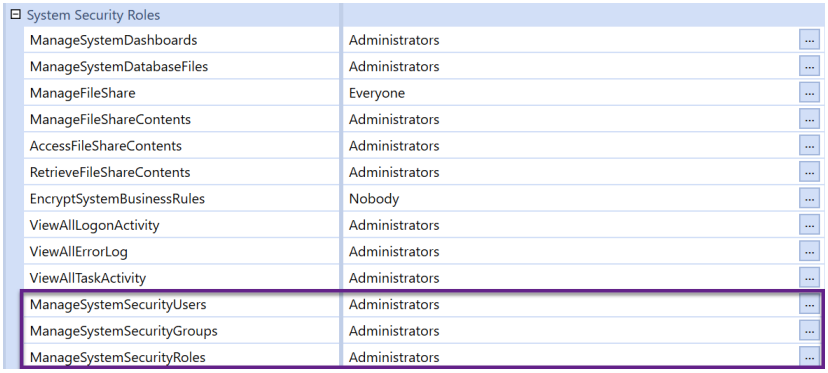
- Tips and Best Practices.
- Creating Users.
- Managing Users.
- Loading and Extracting Users.

Requirements

To let non-Administrators create and manage users and groups, grant them the required security roles. This involves assigning the group to which non- Administrator belongs, to the required roles.

1. Click **System** > **Security** > **System Security Roles**.
2. Click **Edit** by these roles to assign the group in which the user is a member:
    - **ManageSystemSecurityUsers**: Assignees can create, modify and mange users.
    - **ManageSystemSecurityGroups**: Assignees can define, modify and mange groups.
    - **ManageSystemSecurityRoles**: Assignees can manage roles to provide group-

based, functionally-tailored access to artifacts and tools.



System Security Roles	
ManageSystemDashboards	Administrators
ManageSystemDatabaseFiles	Administrators
ManageFileShare	Everyone
ManageFileShareContents	Administrators
AccessFileShareContents	Administrators
RetrieveFileShareContents	Administrators
EncryptSystemBusinessRules	Nobody
ViewAllLogonActivity	Administrators
ViewAllErrorLog	Administrators
ViewAllTaskActivity	Administrators
ManageSystemSecurityUsers	Administrators
ManageSystemSecurityGroups	Administrators
ManageSystemSecurityRoles	Administrators

To revoke the ability to manage users, groups and roles, remove assignees from the relevant role. See Manage System Security.

Tips and Best Practices

- The Administrator is assigned to all system security roles (roles) so can always manage application and system-wide users, groups, artifacts, data, and tools. Removing or reassigning the Administrator group or assigning other users and groups to roles does not revoke the Administrator's access.
- The Administrator cannot be disabled and is unaffected by inactivity thresholds that disable users who try to log in after a specific period of time elapses.

Creating Users

1. Click **System** > **Security** > **Users**.
2. Click **Create User**.
3. Enter a name and description.

4. From **User Type**, select the license type purchased that governs the user's access to artifacts and associated OneStream offerings:
    - **Interactive**: They can use all features and tools.
    - **View**:They can access data, reports, and dashboards in a production environment and associated database, but cannot load, calculate, consolidate, certify, or change data.
    - **Restricted**:They cannot use some OneStream Solution features such as Lease, Account Reconciliation and more due to contractual limitations.
    - **Third Party Access**:They can access applications with a third-party application by logging in using a named account. They cannot change data, modify artifacts or access the Windows application or a browser-based application.
    - **Financial Close:** They can use Account Reconciliation and Transaction Matching OneStream Solutions.
5. Set **Is Enabled** to **True** to activate the user. Select **False**to deactivate the user.
6. The information in **Status** will reflect the user's activity, such as their latest login. **Inactivity Threshold** displays the number of days a user can remain active in the system without logging in. The user receives an error if they try to log in after the specified number of days elapses. See Creating Users.
7. Read About User Authentication, then Specify Authentication Settings.

About User Authentication

You can add and authenticate users as:

- Native users that are managed locally in OneStream.
- External users referenced by an external identity provider (IdP).

Specify Authentication Settings

1. From **External Authentication Provider**, indicate how to authenticate the user:
    - To use native authentication: Select **Not Used** and enter the user's password in **Internal Provider Password**. The first time the user logs in, they can change their password.
    - To use external authentication:
        a. Select the appropriate external IdP from **External Authentication Provider**.
        b. In **External Provider User Name**, enter the user name in the IdP. For example, if a user's name in Azure AD (Microsoft Entra ID) is Azure_LHall@azure.com, enter Azure_LHall@azure.com. This name must be unique and match the user name in the IdP.
2. Specify Preferences and Group Membership .

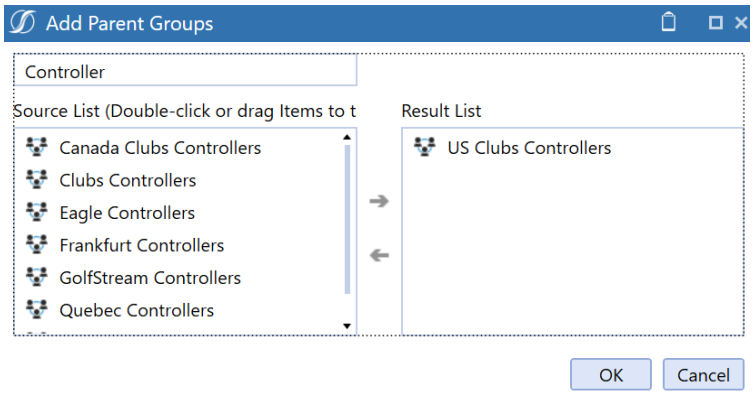
Specify Preferences and Group Membership

1. In **Email** enter the email address with which the user can receive alerts and messages, such as those generated with business rules.
2. In **Culture** select the user's locale. Supported locals and languages are specified during OneStream server configuration. See International Settings.
3. In **Grid Rows Per Page** specify how many rows to display on grids before a page break. Consider the rate of connectivity and screen resolution.
4. Use **Custom Text** to personalize aspects of functionality given the user's responsibilities. For example, you could define a text field to:

- Act as a metadata tag, limiting who the user can email.
- Filter a distribution list or to provide text and images for the user's default workflow profile.
- Launch a functionally-tailored view of the user's workspace, such as reporting for controller or executive.

See "Text 1-8" in Entity Dimension.

5. In **Group Membership**, click **Add Groups**to include the user in the groups that provide access to the features and tools that the user needs.



If the appropriate group does not exist, define it. See Creating and Managing Groups.

6. Click **OK**, **Save** and then **Load**.

When processing finishes, review the user list and user settings to ensure the loaded users are correctly defined.

Managing Users

Perform the following tasks to edit, copy, delete and perform other management tasks. You can also manage users with API functions.

1. Ensure that you have the necessary security role. See Managing Users.
2. Click **System** > **Security** > **Users**.
3. Select a user, then the action to take. For example:
    - **Delete Selected Item**: Permanently delete the user.
    - **Copy Selected Item**: Create a new user based on the selected user's settings. See Copying Users.
    - **Show all parent groups for user**: See the groups in which the user is a member. Since group assignment to system security roles and user interface roles determines what a user can do, use this option to identify their access to artifacts.
4. Click **Save**.

Copying Users

1. Click **System** > **Security** > **Users**.
2. Select the user, then click **Copy Selected Item**.
3. In **New Name**, enter a user name.
4. Select **Copy References made by parent groups** to add the user to the original user's groups except exclusion groups.
5. Click **OK** and modify other settings as needed. See Managing Users.
6. Click **Save**.

Disabling Users

Administrators can disable users, preventing them from logging in by:

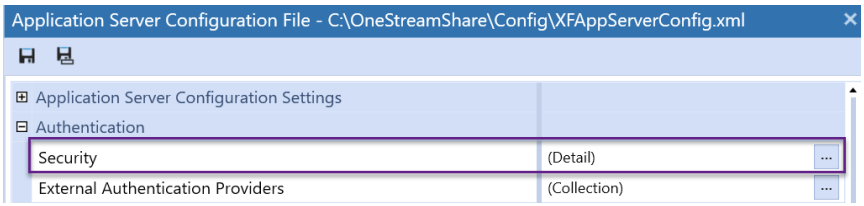
- Modifying a user's account to set **Is Enabled** to **False**. See Disabling a User Manually.
- Specifying an inactivity threshold to deactivate all users - native and external - that try to log in after a particular amount of time elapses. This threshold applies to users in the Administrator group, but not the Administrator. Users that try to log in after the threshold expires receive an error. See Defining an Inactivity Threshold.

Disabling a User Manually

1. Click **System** > **Security**.
2. Expand **Users** and select the user to disable.
3. From Is **Enabled**, select **False**.
4. Click **Save**.

Defining an Inactivity Threshold

1. Launch the Server Configuration Utility.
2. Select **File** > **Open Application Server Configuration File** and browse to: **OneStreamShare\Config\XSFAppServerConfig.xml**.
3. In **Authentication**, click **Security**.



4. In **Logon Inactivity Threshold**, enter the number of days after which inactive users who try to log in, receive an error.
5. Click **OK** then **Save**.

6. In a command prompt, reset IIS.
7. In the application, select **System** > **Security** > **Users**.
8. Select a user and review their **Logon Inactivity Threshold**.

Enabling Users

1. Select **System** > **Security**.
2. Expand **Users** and select the user.
3. In **General**, set **Is Enabled** to **True**.
4. Click **Save**.

Removing Users From Groups

Because group membership determines access to artifacts and tools, be sure that you want to remove a user from a group.

1. Select **System** > **Security**.
2. Expand **Users** then click the user.
3. In **Group Membership**, select the group and click **Remove Selected Item**.
4. Click **Save**.

Managing Users With BRApi

You can perform some user and group management tasks using BRApi functions such as CopyUser, DeleteUser, and CopyGroup. The ability to use these functions is governed by your role as an Administrator and may require a Manage System Security role. See Manage System Security.

For example, if a dashboard's design includes a button that uses a BRApi function to support adding users, users that click the button are validated to ensure they are in the Administrator group or associated with the **ManageSystemSecurityUsers** role.

Loading and Extracting Users

You can bulk load user accounts, groups, and security instead of defining them manually. You can load these artifacts in two ways:

- Use the **SecurityTemplate.xlsx** file provided by the SampleTemplates OneStream Solution to generate an xml load file. We recommend this option. See the *OneStream Sample Templates User Guide*.
- Extract users in the sample GolfStream application, modify the xml export file to specify user and group properties, and then load the file. See Loading Users.

Administrators must load users and groups into a new application or one without existing users and groups. If user accounts exist, loaded users are validated by comparing their current security settings to those in the xml load file.

Requirements

By default, only Administrators can load and extract users. To let other users load and extract users, grant them these system security roles:

- SystemLoadandExtractPage System. This is a User Interface role.
- ManageSystemSecurityUsers
- ManageSystemSecurityGroups
- ManageSystemSecurityRoles

Loading Users

1. Click **System** > **Tools** > **Load/Extract**.
2. Select **Load**and browse to an xml load file.
3. Click **Load**.
4. When processing finishes, review the user list and user settings to ensure the loaded users are correctly defined.

Extracting Users

You can extract users, groups, and security roles to an xml export file.

1. Click **System** > **Tools** > **Load/Extract**.
2. Click **Extract** then select **File Type** > **Security**.
3. In **Items to Extract**, select **Users**. Select **Export Unique IDs** to also extract each user's ID.
4. Perform a task:
    - Click **Extract** to specify where to save the extract file.
    - Click **Extract and Edit** to view and modify the data in an XML Editor.

Creating and Managing Groups

By default, only an Administrator can perform the tasks described in the following topics. An Administrator can enable other users to perform these tasks by assigning specific system security roles. See Requirements.

In this topic:

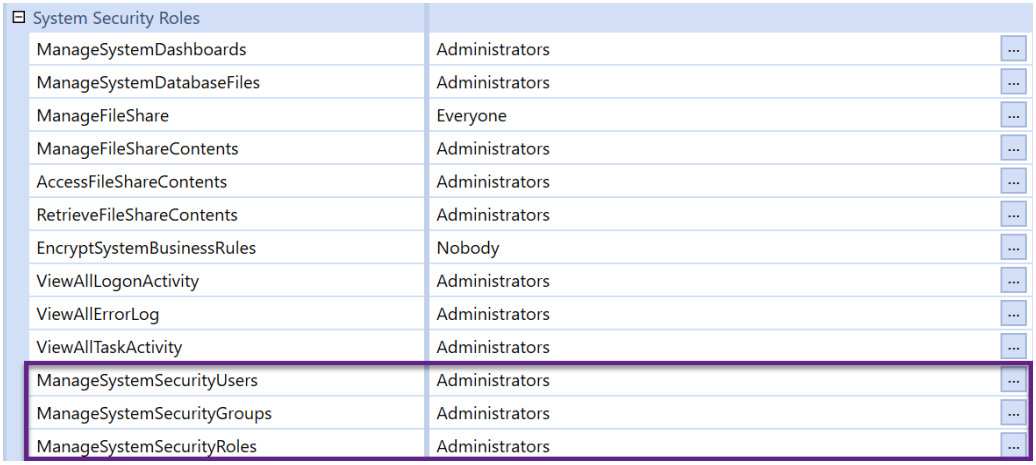
- About Groups and Inherited Security.
- About Exclusion Groups.
- Tips and Best Practices.
- Creating Groups.
- Creating and Managing Groups.
- Creating and Managing Groups.
- Creating and Managing Groups.

Requirements

To let non-Administrators create and manage users and groups, grant them group-based access to the required security roles.

1. Click **System** > **Security** > **System Security Roles**.
2. Click **Edit** by these roles to assign the group in which the user is a member:
    - **ManageSystemSecurityUsers**: Assignees can create, modify and mange users.
    - **ManageSystemSecurityGroups**: Assignees can define, modify and mange groups.
    - **ManageSystemSecurityRoles**: Assignees can manage roles to provide group-

based, functionally-tailored access to artifacts and tools.



System Security Roles	
ManageSystemDashboards	Administrators
ManageSystemDatabaseFiles	Administrators
ManageFileShare	Everyone
ManageFileShareContents	Administrators
AccessFileShareContents	Administrators
RetrieveFileShareContents	Administrators
EncryptSystemBusinessRules	Nobody
ViewAllLogonActivity	Administrators
ViewAllErrorLog	Administrators
ViewAllTaskActivity	Administrators
ManageSystemSecurityUsers	Administrators
ManageSystemSecurityGroups	Administrators
ManageSystemSecurityRoles	Administrators

To revoke the ability to manage users, groups and roles, remove assignees from the relevant role. See Manage System Security.

About Groups and Inherited Security

You cannot assign individual users access to tools and artifacts. System security roles (roles) to which you assign groups determine this access. Create groups to grant large numbers of users or other groups the functionally-tailored access that they require.

You can define nested, hierarchical groups to best suit your organizational entities, workflows and reporting structures. Nested groups contain lower-level child groups. In this case, child groups - and the users that they contain - inherit the access defined for the parent group.

> **NOTE:** Groups cannot be externally authenticated.

About Exclusion Groups

Use exclusion groups to grant almost everyone - except a particular group or a small number of users - access to a tool or artifact. For example, if everyone but Jim Fey and Lee Diaz must create dashboards, define an exclusion group in which Jim and Lee are members set to "Deny Access". All other members are set to "Allow Access". To easily grant access to dashboards, assign the exclusion to which gives everyone access except Lee and Jim access.

Because groups providing access to data and artifacts may contain many groups, removing just a few users - to reflect a corporate reorganization for example, from the group hierarchy can be time consuming. To handle the reorganization, create an exclusion group with the users involved and apply it to the roles they no longer need.

Tips and Best Practices

- The Administrator is assigned to all roles, so they can always manage artifacts, data and tools. Assigning other groups to security roles does not revoke the Administrator's access.
- Child groups, nested in higher-level parent groups, can access the tools and artifact that the parent group can, given the parent group's assignment to system security roles (roles). This access is inherited in group hierarchies, from a parent level downward to child groups.
- Removing child groups from parent groups revokes access to the tools and artifacts that the parent group provides, based on its assignments to roles.
- In an exclusion group, access to artifacts is determined based on the exclusion order you specify, regardless of a user's membership in a group. To ensure that users who are in several groups can not access artifacts but everyone else can, put:
    - The groups to which the users belong at the top, set to "Allow Access".
    - The individual users below the groups and set the users to "Deny Access".

Creating Groups

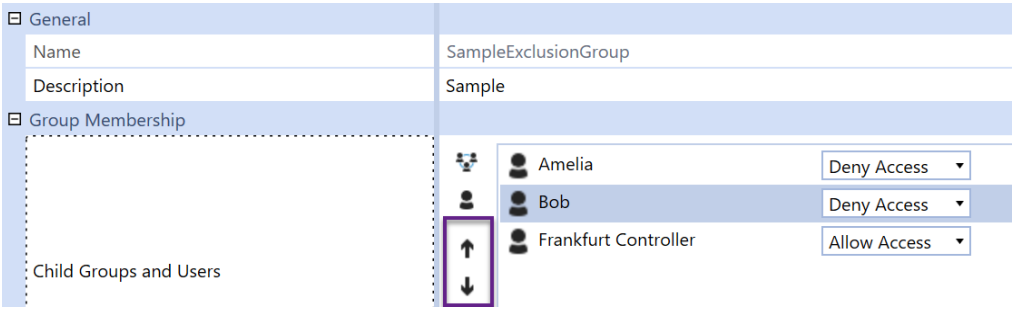
1. If you are not an Administrator, be sure you meet the Creating Groups.
2. Click **System** > **Security**.
3. Click **Groups** then **Create Group**.
4. Enter an intuitive name and description.
5. In **Group Membership** specify the users or child groups to add to the group you are defining, or select a parent group to which to add your new group. Perform any task:
   - To include users and groups in the new group, click **Add User** | **Add Child Group** and selecting the user or group. Members can access the artifacts and tools for each system security role to which the group is assigned.
   - To revoke group membership, select users and groups and click **Remove**.This revokes their access to the tools and artifacts group membership provides based on role assignments.
   - To nest the group in a parent group, click **Add Groups** in **Parent Groups That Contain This Group** and select a higher level group. Members in this parent group inherit it's access and permissions based on role assignments.
6. Click **Save**.

Creating Exclusion Groups

Define an exclusion group to grant almost everyone - except a particular group or a small number of users - group access to a tool or artifact.

1. Click **System** > **Administration** > **Security**.
2. Click **Groups** then **Create Exclusion Group**.
3. Enter an intuitive name and description that indicates who is being restricted by this group. For example, to omit a department, consider Everyone-But-<Department name>.
4. In **Group Membership** click **Add Child Groups** or **Add Users** to specify who to include in the group.
5. To prevent members accessing artifacts to which the group has role-based access, set particular users or groups to **Deny Access**. Otherwise, set members to **Allow Access**.
6. Use the arrows to order the exclusions carefully, because access is granted - regardless of a user's membership in a group, based on the order you specify.

   For example, if Amelia and Bob are in the Frankfurt Controller group, the order below does not restrict them from artifacts, even though they are listed first, because they are in the Frankfurt Controller group.



   To ensure Amelia and Bob can not access artifacts, put Frankfurt Controller first, set to **Allow Access**. Put Amelia and Bob below the group, set to **Deny Access**.
7. Click **Save**.

Managing Groups

1. If you are not an Administrator, meet the Managing Groups.
2. Click **System** > **Security**.
3. Select **Groups** or **Exclusion Groups** , then select a group.
4. Click to perform any task:
    - **Delete Selected Item**: Permanently delete a group.
    - **Rename Selected Item**: Change a group's name.
    - **Copy Selected Item**: Create a group based on the existing group's settings. Select **Copy child user and group references** to add the same users or groups to the new group.
    - **Show All Users**: View the users in the group.
    - **Show All Groups**: View the lower level groups that are nested in the group.
5. In **Group Membership**, change who is in the current group and modify the parent groups associated with the current group. Perform any task:
    - To withdraw users and groups, select them and click **Remove**. This revokes their access to artifacts and tools based on the system security roles (roles) to which the group is assigned.
    - To add users and groups, click **Add User | Add Child Group**. This grants them access to the artifacts and tools the group provides based on it's role assignments.
    - To withdraw the current group from a parent group, select the parent and click **Remove**. This revokes access to the artifacts and tools provided by the parent group based on it's role assignments.

- To include the current group in another parent group, click **Add Groups** and select the group.

Managing Group Membership

You can add groups and users to, and remove them from, other groups. This may created a hierarchy or impact system security role-based access to artifacts and tools. See Managing Groups.

1. Click **System** > **Security**.
2. Expand **Groups** and select the group.
3. In **Group Membership**, perform any task:
   - To include a user or group, click **Add User** or **Add Group** and select the user or group to add.
   - To withdraw the current group from a parent group, select the parent and click **Remove**. This revokes access to the artifacts and tools provided by the parent group based on it's role assignments.
   - To include the current group in a parent group, click **Add Groups** and select the group.

Loading and Extracting Groups

You can bulk load groups, users and security roles instead of defining them manually. You can load groups in two ways:

- Use the **SecurityTemplate.xlsx** file provided by the SampleTemplates OneStream Solution to generate an xml load file. We recommend this option. See the *OneStream Sample Templates User Guide*.

- Extract the groups in the sample GolfStream application, modify the xml export file to specify your group properties, and then load the file. See Loading Groups.

Administrators must perform the load for a new or empty application that does not contain users or groups. To enable other users to load and extract groups, assign them the necessary security roles. See Creating and Managing Users .

Requirements

To enable other users to load and extract groups, Administrators must assign the group in which these users are members, access to these roles:

- SystemLoadandExtractPage (user interface role).
- ManageSystemSecurityGroups
- ManageSystemSecurityUsers
- ManageSystemSecurityRoles

See Manage System Security.

Loading Groups

1. If you are not an Administrator, see Requirements.
2. Click **System** > **Tools** > **Load/Extract**.
3. Select **Load** and browse to an XML load file.
4. Click **Load**.
5. When processing finishes, review the **Groups** list and individual group settings to ensure the loaded groups are correctly defined.

Extracting Groups

1. If you are not an Administrator, see Requirements.
2. Click **System** > **Tools** > **Load/Extract**.
3. Click **Extract** then select **File Type** > **Security**.
4. In **Items to Extract**, select **Security Groups**.
5. Perform a task:
    - Click **Extract** to specify where to save the extract file.
    - Click **Extract and Edit** to modify the extracted data in an XML Editor.

System Security Roles

The following roles apply to the entire system, not just one Application.

Manage System Dashboards - Use to manage all System Dashboards regardless of access to certain Dashboards. This role links the SystemPane Role and the System User Interface Roles section, meaning this security role must include this group in order to be active.

Manage System Database Files -There are two file systems in the Framework database (such as the System database) and each Application database. Users with the ManageSystemDatabaseFiles and ManageApplicationDatabaseFiles roles have read and write access to all folders and files in these file systems, respectively. Other users can be given read and/or write access to specific folders in the database file systems using the a folder's security settings. This option ties the FileExplorerPage role from the System User Interface Roles section, meaning this security role must include this group in order to be active.

Manage File Share - The File Share is a Windows folder that Application Servers can read and write. It is configured in the XFAppServerConfig.xml file using the FileShareRootFolder setting. The File Share is a server-side storage area where external systems or IT can stage and upload ifles. Users with the ManageFileShare role can edit these folder and files using File Explorer.

Application Server Configurations

Application server configurations can now be made by Administrators and advanced IT persona. Common requests that resulted in support calls can now be addressed internally by advanced OneStream users with the proper access. This process also eliminates the need for an IIS restart for the changes to take place. Understand that as settings changed and saved, they are automatically applied every two minutes.

There are two system roles that manage six categories of configurations. The application maintains an audit trail so changes to the configurations are accessible. Upon install, the application server configurations are enabled by default. To mitigate misuse of configurations settings, Customer Support can disable full features, sections, and property changes via XML/App config.

System Roles

Two roles exist to grant Administrators and advanced IT users the ability to change server configurations:

System Security Role: ManageSystemConfiguration

- Default to Nobody
- Not automatically granted to Administrator

System User Interface Role: SystemConfigurationPage

- Default to Administrator Group - Administrators will have read-only rights
- System User Interface Role should be “view only” to all users assigned

System Configurations

There are six categories of system configurations:

- General
- Environment
- Memory
- Multithreading
- Recycling
- Database Server Connections

> **NOTE:** Memory and Multithreading system configurations must be enabled. Customers should contact support to have this configuration change made for them.

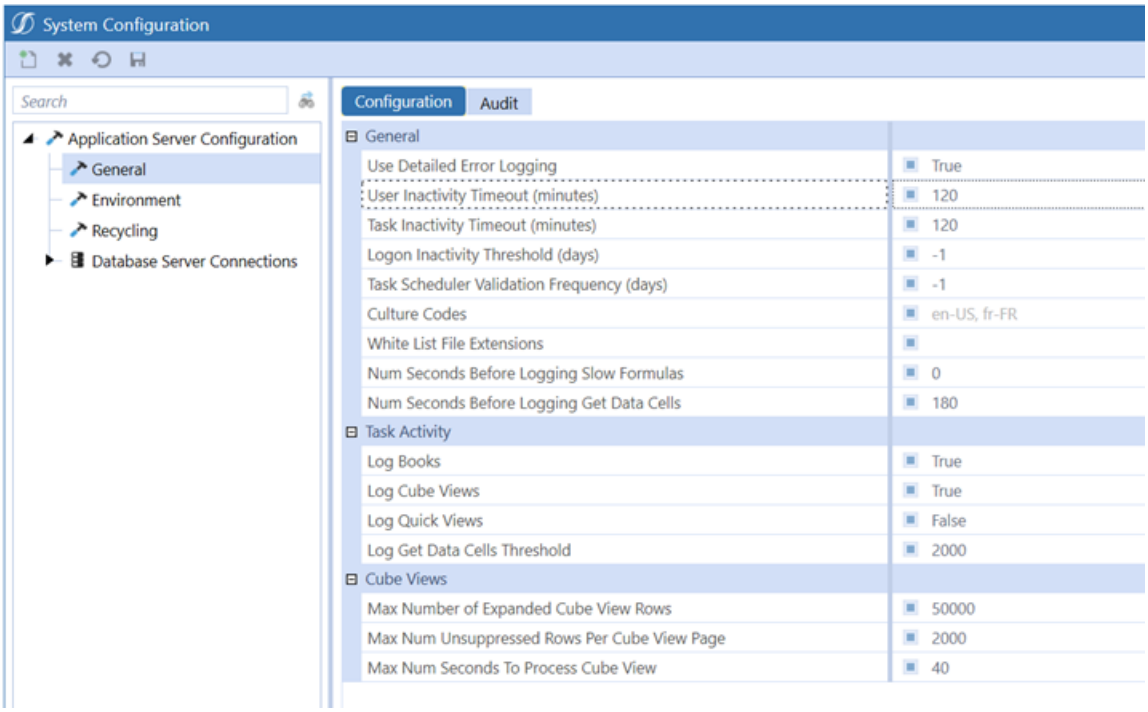
An IIS restart is not required to capture changes as they are automatically applied every two minutes. Saved changes are tracked via the Audit tab, which is available for each server configuration.

When enabled, permitted users can adjust system configurations in app by navigating to System>Administration>System Configuration.

General System Configurations

When the box and setting are greyed out, the property is set based on the configuration file. Deselect the box to adjust your custom value then save. This overrides the configuration file. Select the property's box again to default back to the configuration file.

> **NOTE:** Improperly modifying certain server configurations can impact overall environment or application performance. Consult OneStream support for recommendations.



General Menu

Use Detailed Error Logging: When true, stack trace information is shown. When false, stack trace information is not shown.

User Inactivity Timeout (minutes): Set the number of minutes a user is timed out due to inactivity.

Task Inactivity Timeout (minutes): Set the number of minutes a task is timed out due to inactivity.

Logon Inactivity Threshold (days): Set the Logon Inactivity Threshold (days) to the number of days of inactivity before the user can no longer access the system. Set to -1 to disable the setting.

Task Scheduler Validation Frequency (days): Set the number of days in which the Task Scheduler validation runs.

Culture Codes: Set the appropriate code for display settings using the standard Microsoft Local designation for each language. (ex. en-US)

White List File Extensions: When blank, any file type can be saved to a root folder then uploaded. Add custom file types by clicking the box then the ellipses to restrict the types of files which will be supported in the File Explorer.

Num Seconds Before Logging Slow Formulas: Set the number of seconds before slow formulas are logged. This will enable logging of formulas and impact consolidation performance. Disable when no longer required.

Number Seconds Before Logging Get Data Cells: Set the number of seconds before Get Data Cells are logged. Default is 180 and should only be increased for debug purposes

Task Activity Menu

Log Books: When set to True (default), a log is created in Task Activity when the items are included as Task Activity steps for that specific book. The intention of this feature is to verify entries in the Task Activity grid and the settings in the configuration file work as expected.

Log Cube Views: When set to True, a log is created in Task Activity when a Cube View is opened, a report is run or an export to Excel is completed in the data explorer. The intention of this feature is to analyze data analysis performance.

Log Quick Views: When set to True, a log is created in Task Activity when a new Quick View is created or when rows/columns are shifted/moved around. The intention of this feature is to analyze data analysis performance.

Log Get Data Cells Threshold: This logs the calls to GetDataCells and GetDataCellsUsingScript. It includes context information such as the Excel file name or the Cube View name. It only creates logs if the number of Data Cells being requested is equal to or greater than the value provided in this field.

Cube Views Menu

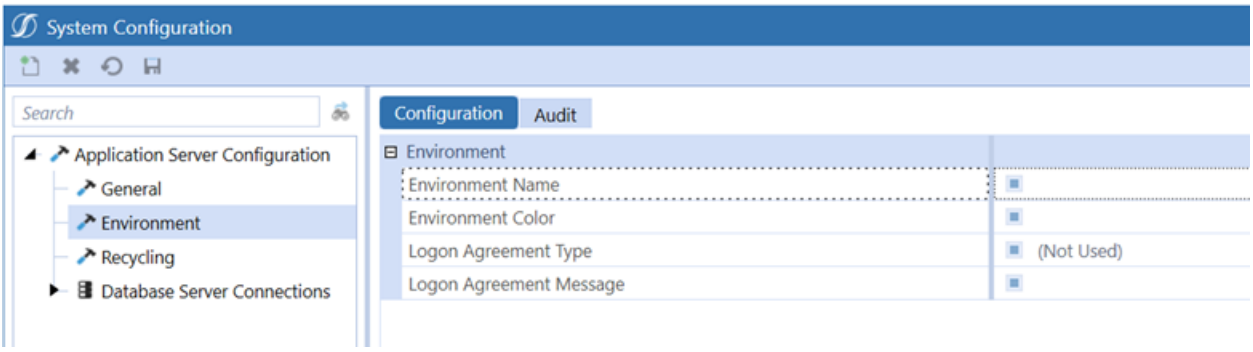
The Cube View configurations will impact all Cube Views in an application environment. They are designed to optimize the Cube View performance through managing the page size and initiating paging which will maximize the availability of server resources. These settings can be overridden to tailor the design and performance of a specific Cube View, using the General Settings/Common/Paging properties found on each Cube View.

Max Number of Expanded Cube View Rows: Set the max number of rows displayed when using an expanded Cube View.

Max Number of Unsuppressed Row Per Cube View Page: The default value of 2000 is used, which is determined by the settings on the OneStream Server Configuration Utility, The maximum value is 100,000 rows. If the Cube View performs well, but you want 2500 rows to display, for example you may want something in the tree to display in the first page, then you would increase the rows.

Max Number Seconds To Process Cube View: This setting impacts paging behavior. The default value of 20 will be used which is determined by the OneStream Server Configuration Utility. The maximum value is 600 seconds.

Environment System Configurations

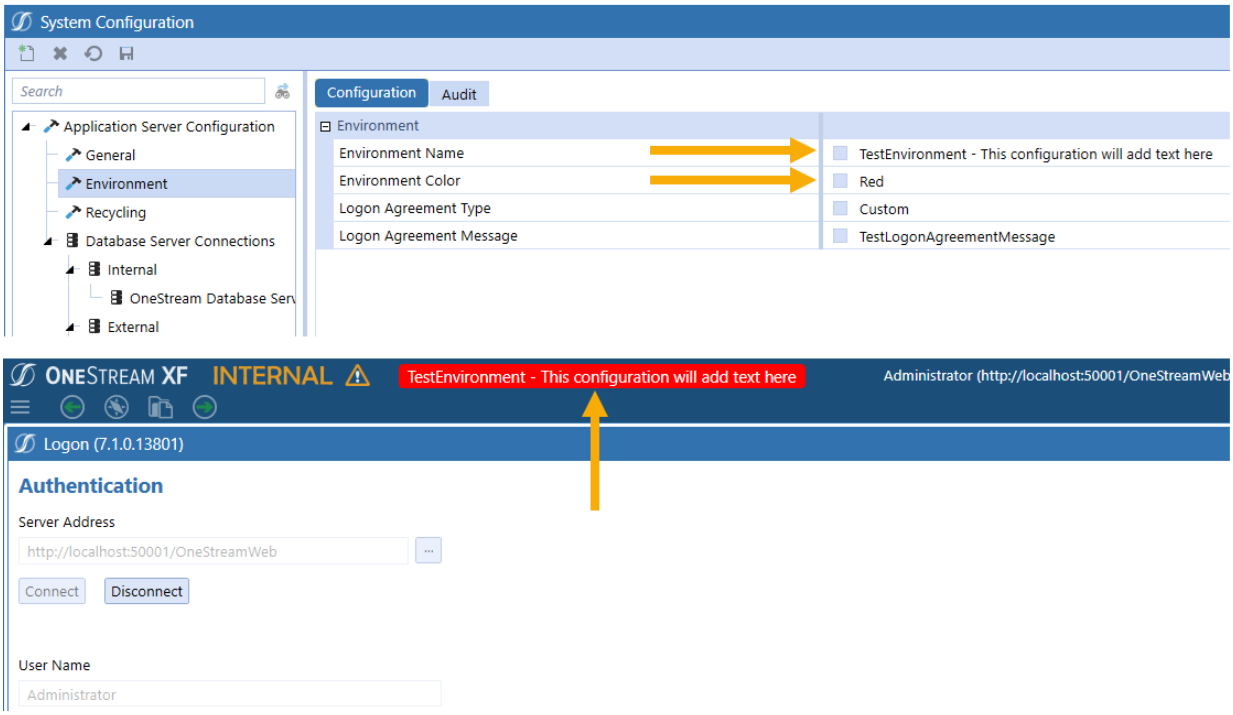


Environment

The configurations allow the Administrator to tailor the environment login process for the user by providing a custom label or by triggering an acceptance criteria upon each login.

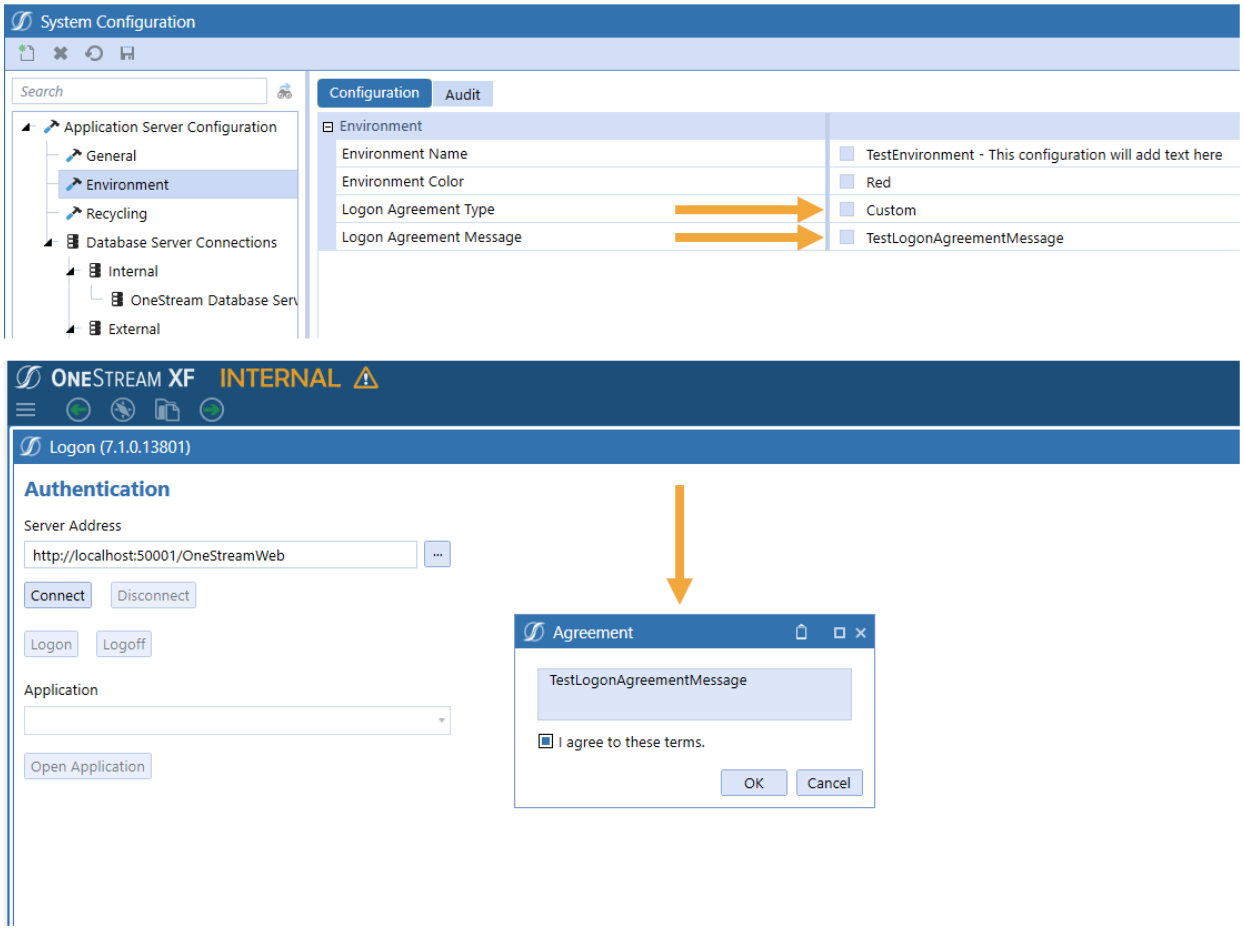
Environment Name: Enter the name to be displayed (in white) for the environment. You can enter up to 150 characters.

Environment Color: Specify a provided environment color or enter a hex value to display the name on a colored background.

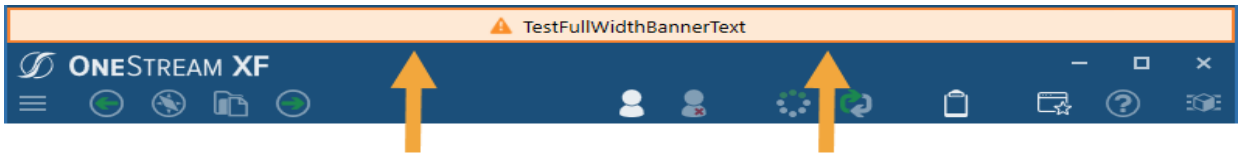


Logon Agreement Type: To display a specific message after a user logs on, select Custom and enter the message text.

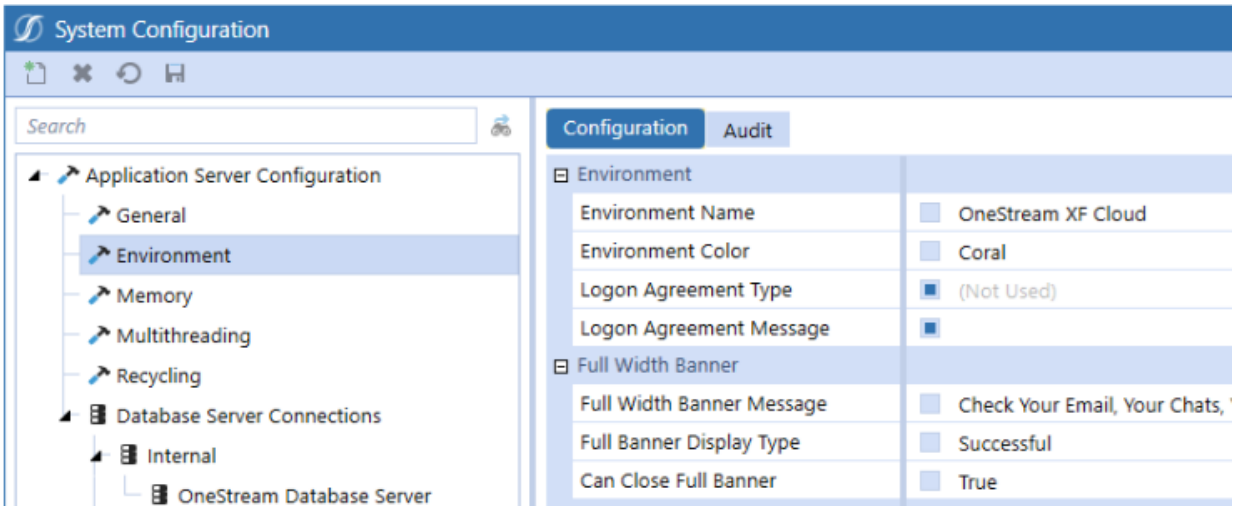
Logon Agreement Message: Enter the message text.



Full Width Banner



The configurations allow you to display information across the top banner indicating varying levels of severity. The banner can be permanently displayed or dismissible.



Full Width Banner Message: Enter the banner message text to be displayed.

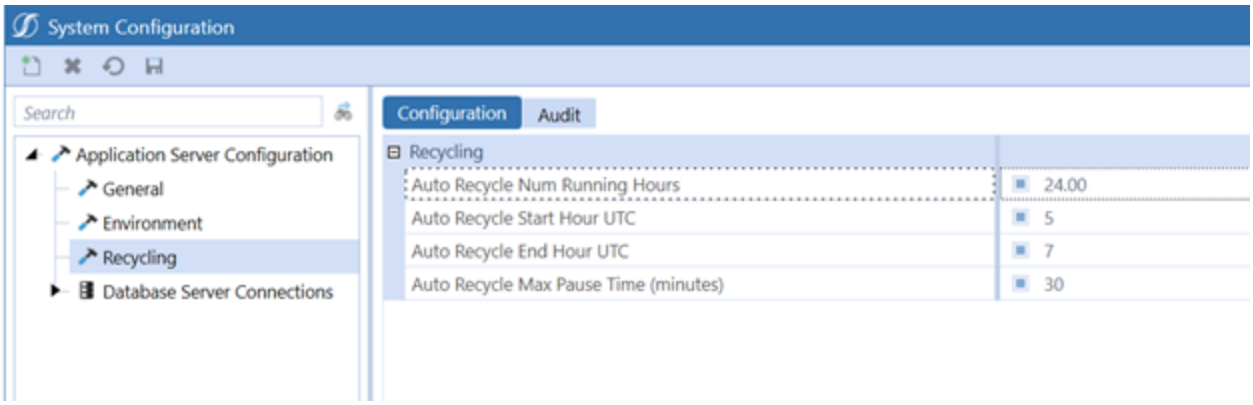
Full Banner Display Type: Indicate the severity of the message by selecting Informational, Warning, Successful, or Critical from the drop-down menu.

Type	Sample
Informational	Informational Banner Sample Text
Warning	Warning Banner Sample Text
Successful	Successful Banner Sample Text
Critical	Critical Banner Sample Text

Can Close Full Banner: When True, the banner may be dismissed after log in. When False, the banner cannot be dismissed.

Recycling System Configurations

Overtime, server memory may become increasing fragmented, which can affect performance and stability. The default configuration of a daily recycling process is standard. The System Configuration page allows the administrator to tailor when these events occur to be best suited for their implementation.



Recycling Menu

Auto Recycle Num Running Hours: Default is 24, which means once a day, the server will recycle. Automatic Recycling allows Application Servers a chance to recycle, which is a recommended practice. These first four settings control this behavior.

Auto Recycle Start Hour UTC: Default is 5, which means 05:00 UTC time. This is the earliest time in a day when a server can automatically recycle. It is best to set this and the End Hour to be a range of time with the lowest amount of Application Server activity.

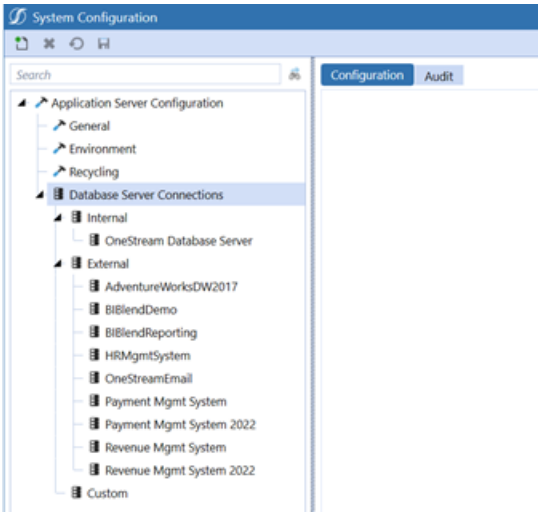
Auto Recycle End Hour UTC: Default is 7, which means 07:00 UTC time. This is the latest time in a day when a server can automatically recycle.

Auto Recycle Max Pause Time (minutes): Default is 30. This means that when it is time to recycle a server automatically, it will first pause from accepting more server tasks, but allow for existing assigned tasks to complete processing for 30 minutes before recycling. If there are no active tasks for this server, it will recycle when the time comes.

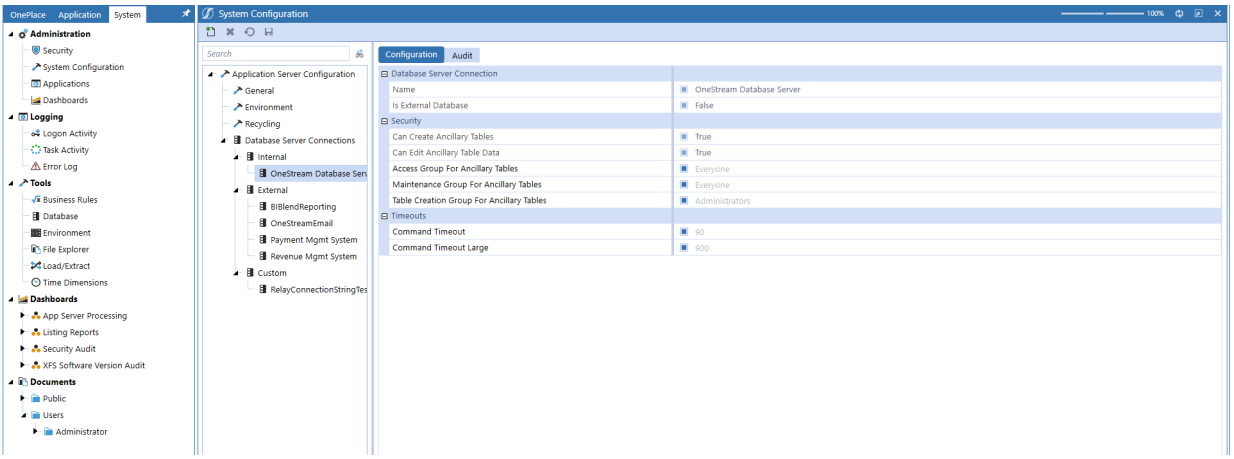
Database System Configurations

The Database Server Connections provides the Administrator with greater visibility into the structure of their environment. Additionally, the administrator can quickly and easily establish database connections to support their model's design requirements.

An IIS restart is not required to capture changes as they are automatically applied every two minutes. Saved changes are tracked via the Audit tab, which is available for each server configuration

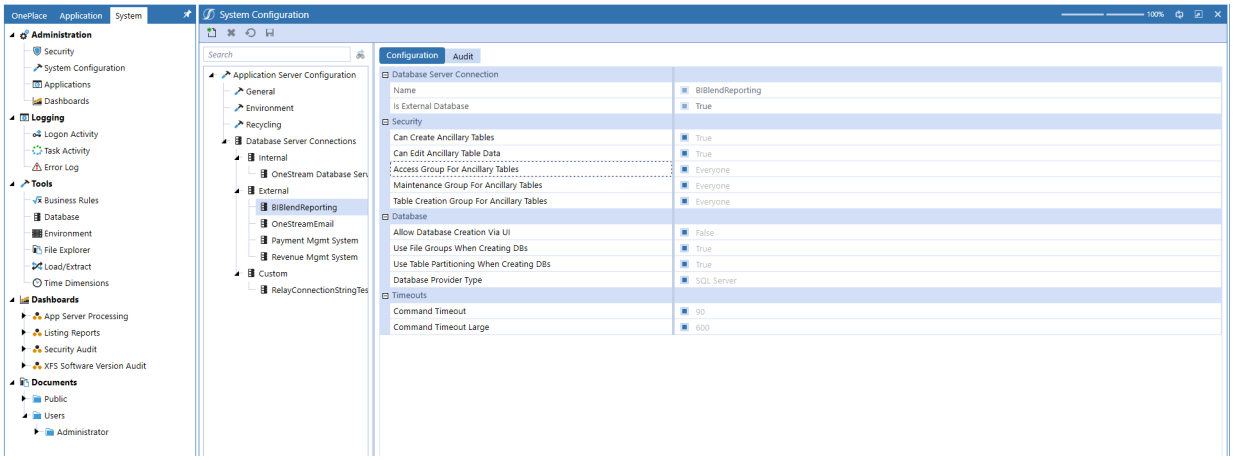


Database Server Connections



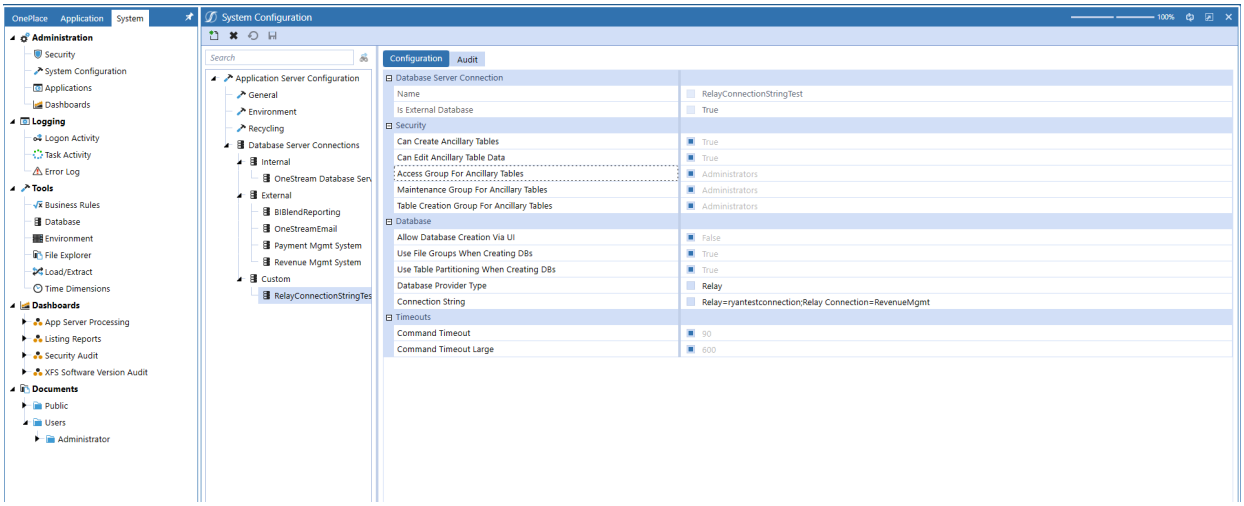
Internal Menu

Configure Database Server Connection, Security, and Timeouts settings from this menu.



External Menu

Configure Database Server Connection, Security, Database, and Timeouts settings from this menu.



Custom

Configure Database Server Connection, Security, Database, and Timeouts settings from this menu.

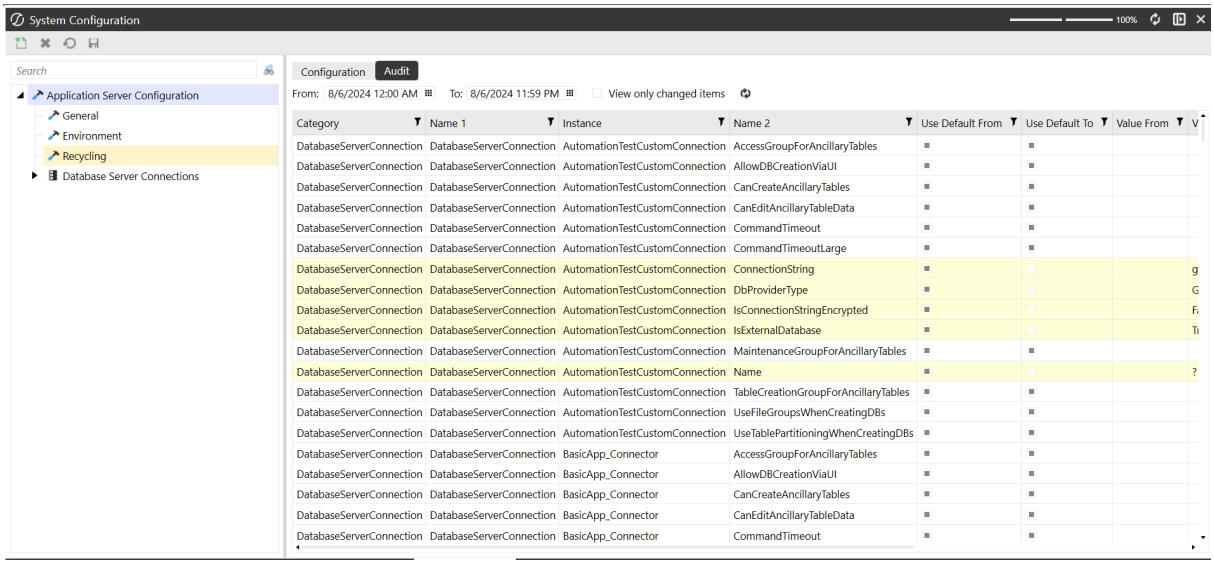
External only and the Connection String will be visible without displaying password and Customer can Edit.

- Connection String will allow for Edits
- Connection String will display “?” where password is used
- Connection String can be modified without changing password and saved with “?” not overriding original password
- Connection String password can be modified and upon save will display “?” in UI
    - Connections Strings using the following will be masked in the UI: pwd; pass; password
    - All others will display password and need to be addressed separately – majority of connection strings password will be handled above

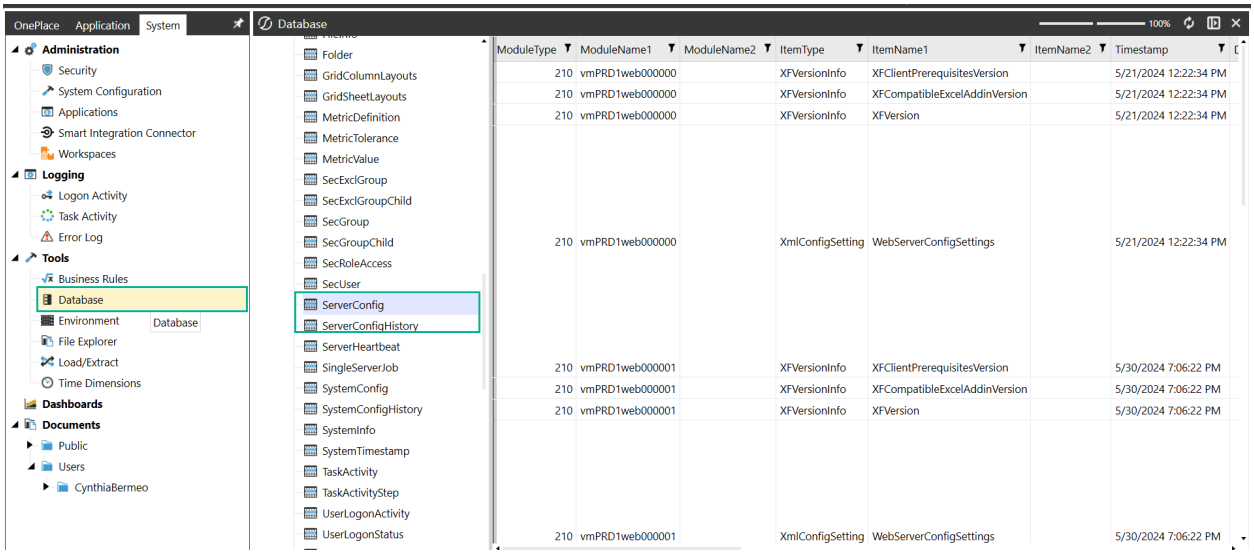
Pre-existing External Connections, when moving over to Customer Connections, should use the previous name to avoid breaking any connections with Task Scheduler, Business Rules, etc.

Audits

Configuration setting changes are tracked via the Audit tab. View changes by navigating to **System > System Configuration > Audit**.

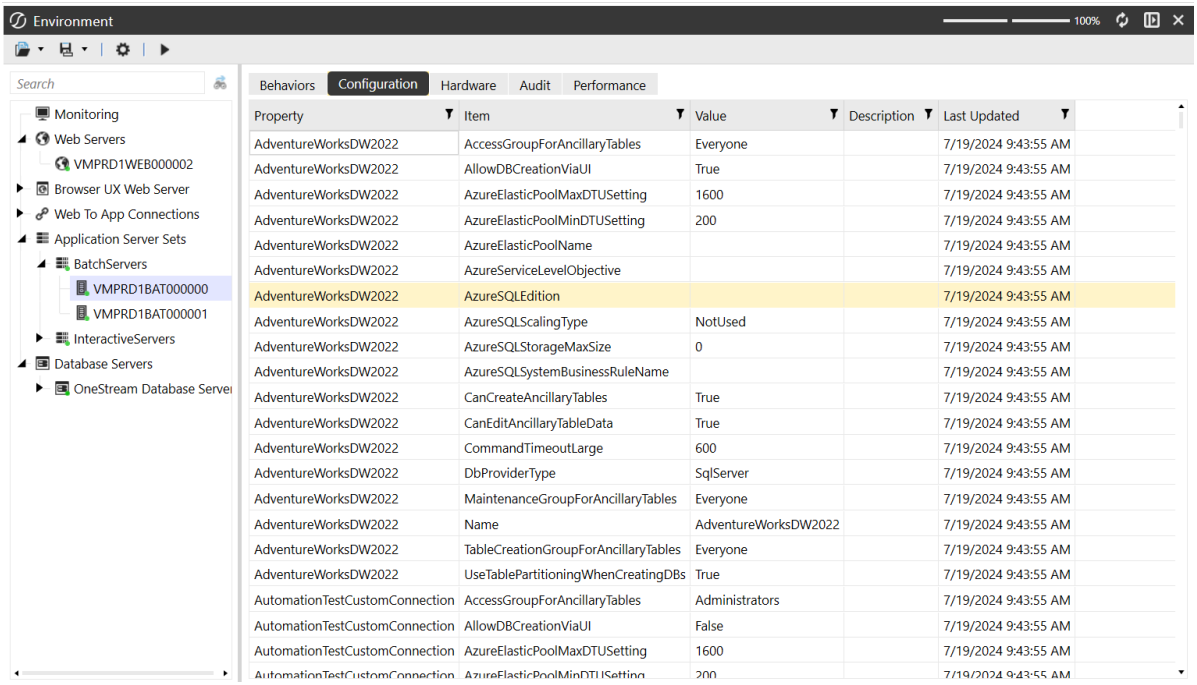


System Database Tables audits are viewed by navigating to **System > Tools > Database > SystemConfig**.



The Environment will show what settings the server is currently using. The server updates every 2 minutes. Here the user can see what settings are being picked up. The “Last Updated” column shows the date and time settings were changed. Regardless of what type of connection (Internal, External, Custom) any available configurations that can be changed will show up here.

Environment audits are viewed by navigating to **System > Tools > Environment > Application Server Sets**



Manage System Security

Manage System Security allows non-administrators to manage users, roles and groups and facilitates a comprehensive, functionally-tailored way to separate user and group responsibilities.

By default Administrators have access to all security roles. Assigning other groups to roles does not remove an Administrator's complete system access. Users with Manage System Security roles can access to the System User Interface Roles of SystemAdministrationLogon and SystemPane.

Roles are not exclusionary or limiting. If granted, users can get additional functionality through their membership in groups having corresponding role assignments. See System Security Roles.

For example:

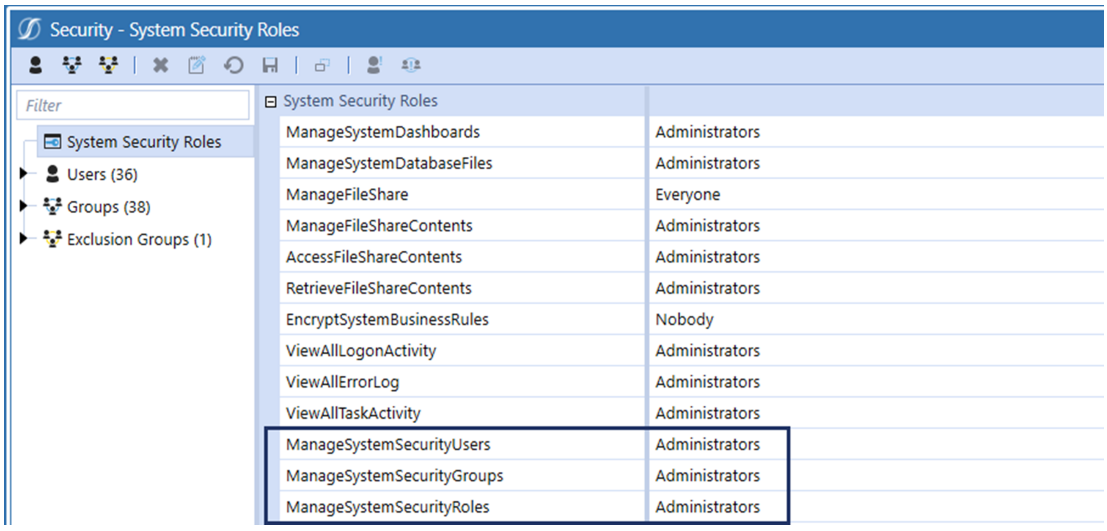
- A member of the IT team who is not an administrator might need to manage the system security roles.
- An employee in the Accounting department who reports to the Controller over consolidations may manage groups and not need access to other areas of the system.

You can delete these security roles to non-Administrators:

- **ManageSystemSecurityUsers**: Grants permission to manage users.
- **ManageSystemSecurityGroups**: Grants permission to manage groups, exclusion groups and group membership.
- **ManageSystemSecurityRoles**: Grants permission to manage system security role assignment.

Access System Security Roles

1. From the System tab, go to **Administration > Security**.
2. Select **System Security Roles**.



Manage System Security Users

This role enables you to:

- Create users
- Modify users
- Delete users
- Disable users

This role enables you to specify these user properties:

- General
- Authentication
- Preferences
- Custom Text

Limitations

Users with the Manage System Security role cannot create, modify, or delete administrators, directly or indirectly. Also, they cannot:

- Add or remove themselves to or from groups or roles.
- Delete themselves.
- Add other users to Manage System Security privileges.
- Add or remove groups they are members of from roles.

To manage group membership or copy users, the **ManageSystemSecurityGroup** is required.

Manage System Security Groups

This role lets you manage groups and exclusion groups. You can also:

- Add or remove members to or from groups and exclusion groups.
- Copy groups except groups with Administrator privileges.

Limitations

Users with this role cannot:

- Modify the Administrators group.
- Assign users to a group that establishes Administrator privileges.
- Modify your membership in other groups.
- Modify the parent group of a group in which the user is a member.

Manage System Security Roles

This role lets you manage system security roles. However, you cannot:

- Modify the **ManageSystemSecurityRole** itself because it requires Administrator level privileges.
- Assign the Everyone or Nobody groups that require Administrator level privileges.
- Add a group to a role of which you are a member.

Load and Extract

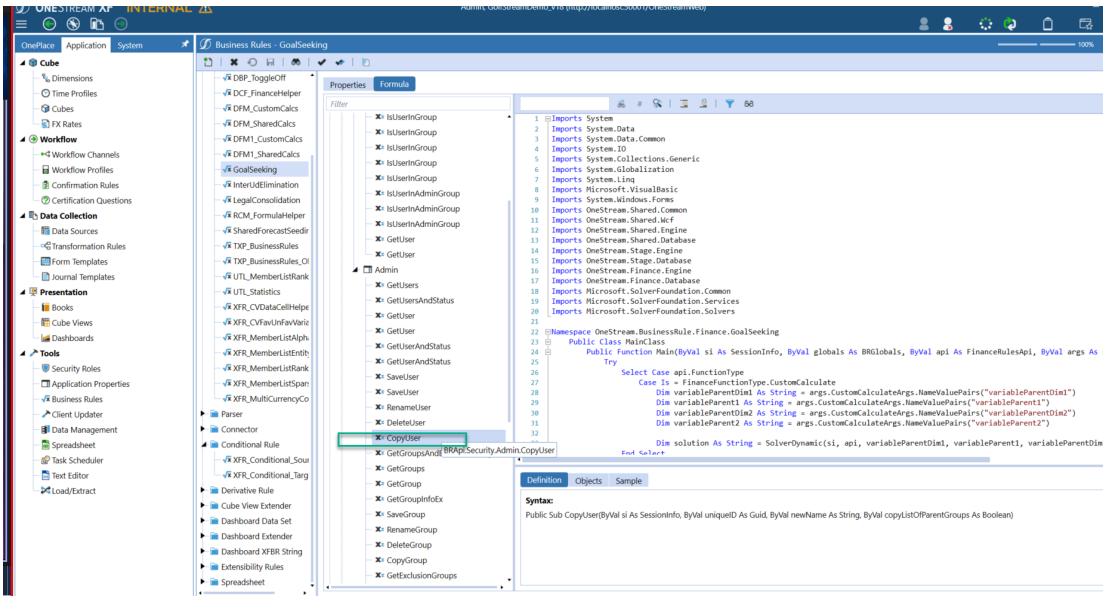
Load and Extract functionality of Security requires a user to have permissions for all three of the Manage System Security roles, as well as the System User Interface Roles of **SystemLoadandExtractPage**.

The controls limiting Manage System Security user's capabilities is enforced during the Load and Extract process. Validation occurs by comparing the current state of security in the target environment to the changed state determined by the processing of the source file. Therefore, the validation of XML loads for Manage System Security users requires that security is pre-existing to determine the changed state.

For example, although Manage System Security users cannot create Administrators, if the current Administrator Groups existed in the target environment prior to the XML load, then the XML will pass the validation and will be loaded. However, when an empty or new environment exists with no pre-existing users and groups, then an Administrator would need to perform the load.

BRApi

You can manage user and group system security using BRApi functions such as CopyUser, DeleteUser and CopyGroup. These are controlled by the assigned Manage System Security role. See System Security Users and Groups for more information.



For example, if a dashboard is created to insert new users, and a dashboard button executed a BRApi to insert a user, the system validates that the user clicking the button is in the Administrators Group or has role permission to ManageSystemSecurityUsers.

Combined Roles

When granted access to more than one of these roles, you gain more functionality within the scope of the designed capabilities and restrictions. For example, if you have both the role of users and groups, you can copy a User or you can also can add a user to a group. Certain functionality requires assignment of combined roles, such as Load and Extract.

File Share Security Roles

ManageFileShareContents

- Exposes the Contents Folder in File Explorer\FileShare under Application and System.
- Grants full rights to create, upload, download and delete folders.

AccessFileShareContents

- Exposes the Contents Folder in File Explorer\FileShare under Application and System.
- Allows only the ability to see the Contents folder and its sub-files, and allows download.

RetrieveFileShareContents

- The Contents Folder is not exposed to the user in the File Share for Application or System using the File Explorer.
- All files are accessible through the OneStream application, such as through Dashboards and Business Rules.

See Application Security Roles in Application Tools for more information on Application based security roles.

Encrypt System Business Rules - This allows a user to encrypt/decrypt a rule from the Business Rule screen in the System tab to obfuscate the contents of the rule from all users..

See Business Rules in under Application Tools for more information on Business Rule Encrypt (Decrypt) functionality and use.

View All Logon Activity - If assigned the required access to view the System tab and the System tab | Logging | Logon Activity page, users in the assigned security group can see the Logon activity for all users in the environment.

View All Error Log - If assigned the required access to view the System tab and the System tab | Logging | Error Log page, users in the assigned security group can see the Error Log for all users in the environment.

View All Task Activity - Users in the assigned security group can view the tasks and detailed child-steps through the Task Activity icon in the toolbar. If they have access to the System tab and page, they can also see this information in the System tab | Logging | Task Activity page.

System User Interface Roles

The following roles grant access to key features and tools.

SystemAdministrationLogon
This is for the System Administration application and set to Administration by default. When a security group is assigned to this role, it becomes available in the Application list during logon.

SystemPane
Based on the security group chosen, this role lets you access the System Tab found at the bottom left of the screen. By default, only administrators have access to this tab.

ApplicationAdminPage

Based on the security group chosen, this role lets you access the Application tab at the bottom left of the screen. By default, only administrators can access this tab.

SecurityAdminPage

Based on the selected security group, this role lets you view but not modify security artifacts and settings other than your password on System > Administration. By default, only administrators have access to this section.

SystemDashboardAdminPage

Based on the security group chosen, this will allow a user access to the Dashboard section found under System Tab > Administration > Dashboards. This section ties to the ManageSystemDashboards which is under the System Security Roles in the above section. By default, only administrators have access to this.

ApplicationServersPage

Based on the security group chosen, this will allow a user access to the Application Servers section found under System Tab >Tools > Application Servers. By default, only administrators have access to this.

DatabasePage

This role lets you access the Database Page on System > Tools. By default, only system administrators have access to this section.

FileExplorerPage

Based on the selected security group, this role lets you access the File Explorer on System >Tools. This section ties to the ManageSystemDatabaseFiles and ManageFileShare roles. By default, only administrators can access the File Explorer.

SystemLoadExtractPage

Based on the security group chosen, this role lets you access Load/Extract on System > Tools but you cannot actually import or extract items. By default, only administrators have access to this section.

ErrorLogPage

Based on the security group chosen, this role lets you access Error Log on System > Logging. By default, only administrators can access to this section.

LogonActivityPage

Based on the selected security group, this role lets you access to Logon Activity on System > Logging. You can view all users, but cannot log them off. By default, only administrators have access to this section.

TaskActvityPage

Based on the selected security group, this role lets you access Task Activity in System > Logging. By default, only administrators can access to this section.

TimeDimensionPage

Based on the security group chosen, this will allow a user access to the Time Dimension on System > Tools. By default, only administrators have access to this section.

System Security Users and Groups

Administrators click **System** > **Administration** > **Security** to define and manage users, groups and system security roles (roles). Users, groups and roles are not application-specific. See Creating and Managing Groups and Creating and Managing Users .

Every user must be assigned a user ID. Users can be added as native users or as references to users stored in external repositories (e.g. Active Directory). Users can be externally authenticated with these standard providers.

- LDAP
- MSAD
- Okta
- PingFederate
- Azure AD (Microsoft Entra ID)
- SAML

For information about external authentication with standard providers, see the *Installation and Configuration Guide*.

Managing Users and Groups Using BRApi Functions

Administrators can manage users and groups with the following BRApi functions:

```
BRApi.Security.Admin.GetUsers

BRApi.Security.Admin.GetUser

BRApi.Security.Admin.GetUser

BRApi.Security.Admin.SaveUser

BRApi.Security.Admin.RenameUser

BRApi.Security.Admin.DeleteUser

BRApi.Security.Admin.CopyUser

BRApi.Security.Admin.GetGroupsAndExclusionGroups

BRApi.Security.Admin.GetGroups

BRApi.Security.Admin.GetGroup

BRApi.Security.Admin.GetGroupInfoEx

BRApi.Security.Admin.SaveGroup
```

```
BRApi.Security.Admin.RenameGroup

BRApi.Security.Admin.DeleteGroup

BRApi.Security.Admin.CopyGroup

BRApi.Security.Admin.GetExclusionGroups

BRApi.Security.Admin.GetExclusionGroup

BRApi.Security.Admin.SaveExclusionGroup

BRApi.Security.Admin.RenameExclusionGroup

BRApi.Security.Admin.DeleteExclusionGroup

BRApi.Security.Admin.CopyExclusionGroup

BRApi.Security.Admin.GetSystemRoles

BRApi.Security.Admin.GetApplicationRoles

BRApi.Security.Admin.GetRole

BRApi.Security.Admin.CopyExclusionGroup
```

Examples

Get a UserInfo object and change the User Description

```
    Dim objUserInfo As UserInfo = BRApi.Security.Admin.GetUser(si, "Administrator")

    If Not objUserInfo Is Nothing Then

        objUserInfo.User.Description = "New Description"

        BRApi.Security.Admin.SaveUser(si, objUserInfo.User, False, Nothing, TriStateBool.Unknown)

    End If
```

Get a Group and UserInfo object and add the Group to the User's list of parent Groups

```
    Dim objGroupInfo As GroupInfo = BRApi.Security.Admin.GetGroup(si, "TestGroup")

    If Not objGroupInfo Is Nothing Then


        Dim objUserInfo As UserInfo = BRApi.Security.Admin.GetUser(si, "TestUser")

        If Not objUserInfo Is Nothing Then
```

```
            If (Not objUserInfo.ParentGroups.ContainsKey(objGroupInfo.Group.UniqueID)) Then
                Dim parentGroupIDs As List(Of Guid) = objUserInfo.ParentGroups.Keys.ToList()
                parentGroupIDs.Add(objGroupInfo.Group.UniqueID)

                BRApi.Security.Admin.SaveUser(si, objUserInfo.User, True, parentGroupIDs,
TriStateBool.Unknown)
            End If
        End If
    End If
```

Create a User

```
    Dim objUser As User = New User()
    objUser.Name = "NewUser"
    objUser.Text1 = "Test Text 1"
    BRApi.Security.Admin.SaveUser(si, objUser, False, Nothing, TriStateBool.Unknown)
```

Create a Group

```
    Dim objGroup As Group = New Group()
    objGroup.Name = "NewGroup"
    Dim objGroupInfo As GroupInfo = New GroupInfo()
    objGroupInfo.Group = objGroup
    BRApi.Security.Admin.SaveGroup(si, objGroupInfo, False, Nothing, TriStateBool.Unknown)
```

International Settings

Culture-aware number and text formatting can be used with:

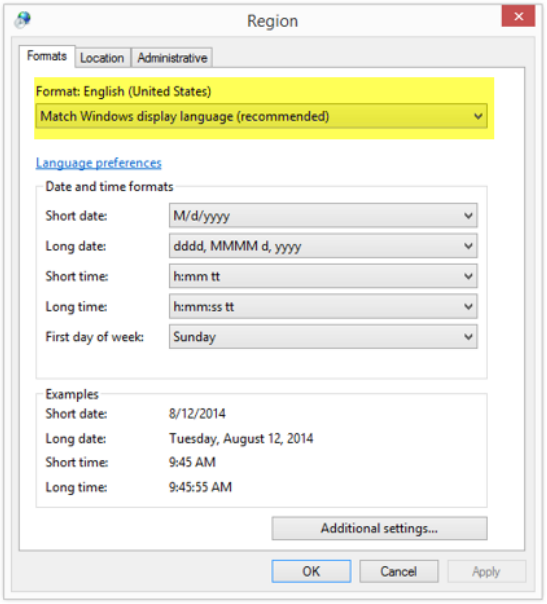
- The web-based client
- The Excel add-in
- Server components

For consistency, we recommend that each user's Culture setting match the Windows Regional settings of their primary computer.

Specifying Culture Settings on the Client Machine Operating System

Number and text formatting for Microsoft Excel is controlled by the client operating system's Windows Regional settings. To configure these settings:

1. Access the operating system's **Control Panel** and select **Region.**
2. Click **Format** and set properties to reflect regional preferences. For exmple:



3. Click **Location** and set the **Home Location** to reflect the operating system's location such as Canada, Germany or Australia.

Specifying Server Side User Culture Settings

For the web based client and server components, specify culture settings for users by modifying user profiles in Security. The necessary culture codes must first be defined in the Application Server Configuration file.

Working With System Applications

After logging on as an Administrator, you can open the System Administration application that lets you flexibly create a new application based on existing databases or copied databases. For existing Applications, click **System** > **Administration** > **Applications**.

Create Application Reference

Click to link an application and an existing database if they are from the same release version. Select a Database Server Connection and enter a Database Schema Name. The new Application will display.

Allow Database Creation Via UI must be **True** in the Application Server Configuration.

Create Application Database

Click to generate a new database.

Application Name

Enter the name of the new application.

Description

Provide a brief description (optional).

Database Server Connection

Select the database server from the drop-down list.

Database Schema Name

This is created by default and stored in SQL.

Time Dimension Definition

Click the ellipsis and select the desired Time Dimension XML file. See Time Dimensions in System Tools for details.

The new application displays in the Application list when you log on.

> **NOTE:** Database Status indicates if a selected application has an attached database. If the status is "Error", the database was deleted or detached and will not be available in the Application list.

Delete Selected Application Reference

Click to remove an application's database link. This will not delete database files, just references in the application.

Test Database Connection

Select an existing application and click the test checkbox to verify an active connection.

System Dashboards

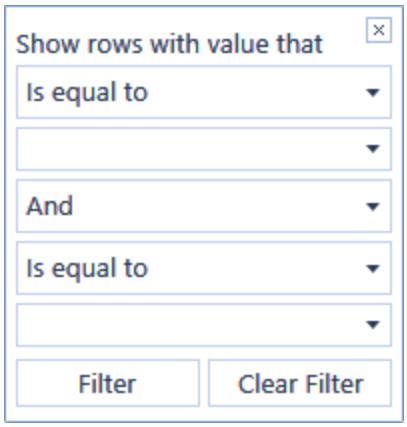
System Dashboards are similar to Application Dashboards but differ in that you can use OneStream Framework database objects such as security and users. Application Dashboards use Application-level data stored in Cubes.

See Application Dashboards in Presenting Data With Books, Cube Views and Dashboards for information about designing Dashboards.

Logging

Each logging section display content - that you can sort and filter - in a grid. To sort, click a column heading and select ascending or descending.

Click on to filter by criteria such as the following:



Navigate pages by clicking page numbers, the forward button, the back button and so on. These buttons are to the bottom left of a grid. To export data, right-click, select **Export** and then an export file type.

Logon Activity

Use the following to identify who is logged on who logged off.

Logoff Selected Session

Administrators can use this to logoff any user session

Clear Logon Activity

Clears all logon activity

This grid will display:

Logon Status

This shows when users logged on and logged off.

User

This shows the user ID.

Application

This shows the application the user is logged into.

Client Module

This will display items such as Excel.

Client Version

This will display the version of the application you are using.

Client IP Address

This will display the end users IP address.

Logon Time

A time stamp of when the user logged in.

Last Activity Time

A time stamp of the user's last activity.

Logoff Time

A time stamp of when the user logged off.

Primary App Server

This shows the application server the end user is utilizing.

Task Activity

You can access task activity to identify user actions in two ways:

- Click **Task Activity** at the top right section of the web client.
- Select **System** > **Logging** > **Task Activity**.

Clear task activity for current user

Clears all activity for that user.

Clear task activity for all users

Clears all activity for ALL users.

Selected task information

Gives the ability to drill down through all steps for that activity.

Running task progress

Gives the ability to view progress of other users' activities.

Refresh

This refreshes the Task Activity in order to see any changes made

Task Type

This shows the user's activity. (e.g., Consolidate, Process Cube, Load and Transform, Clear Stage Data, etc.)

Description

This shows the details of the activity. (e.g., the POV, Multiple Data Units, etc.)

Duration

This displays the length of time the activity took.

Task Status

This shows the status of the activity. (e.g., Completed, Failed, etc.) Canceling the task will transition it from Canceling to Canceled.

User

This shows the user ID.

Application

This shows the application to which the task was processed.

Server

This shows the application server being utilized.

Start Time

The starting time stamp of the activity.

End Time

The ending time stamp of the activity.

Queued CPU

This provides the % CPU utilization for when the task was initiated.

Start CPU

This provides the % CPU utilization when the job began from the queue.

For more details, Task Queuing, see Data Management in Application Tools.

Within the grid, there are two icons on the left side of the row. If highlighted, there is the ability to drill down by clicking on them.

The first icon shows child steps within a particular task that has run. The second shows detailed information of the error when present.

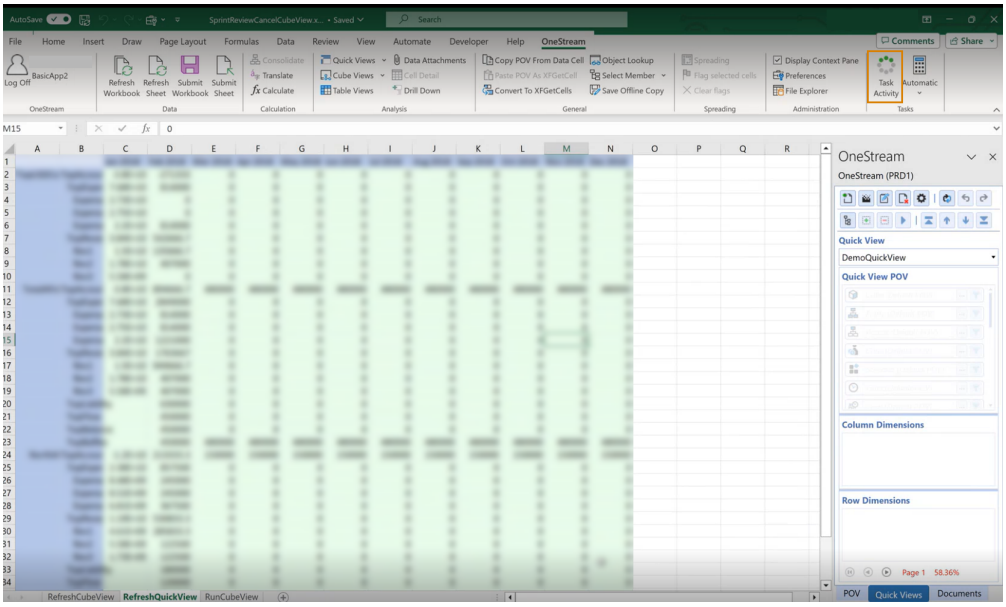
Enhancing Task Activity

Task Activity allows you to cancel a long-running cube view in these areas:

- Enhancing Task Activity: Cancel Cube Views
- Task Activity in Excel
- Cube View Paging
- Quick View Paging

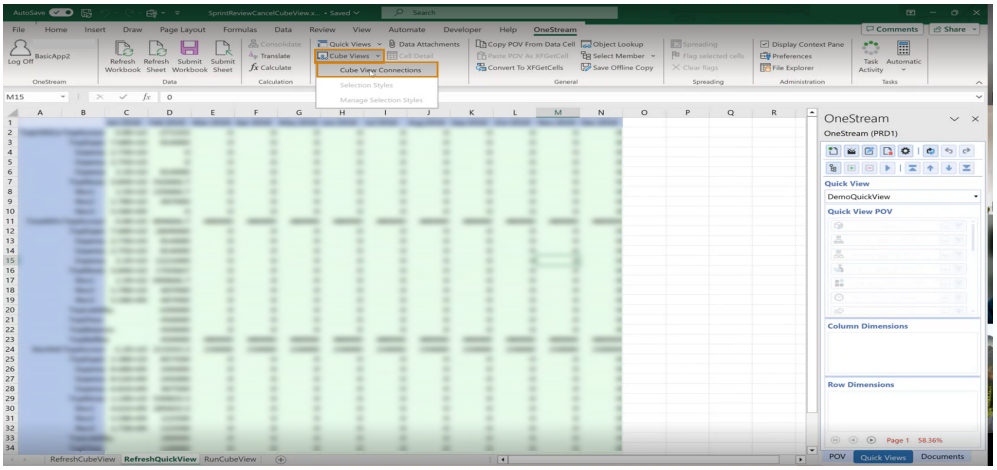
Task Activity in Excel

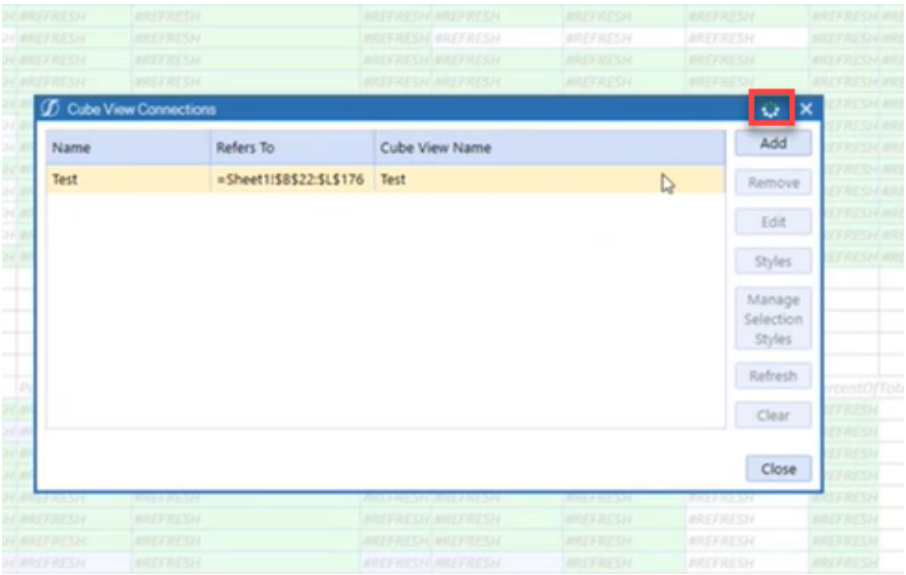
You can cancel a long-running cube view in Excel Add-In. You do not have to be logged into the Windows application. The Task Activity in Excel Add-In allows you to cancel your task. If you are an Administrator, you can cancel another user's running task. You can access the Task Activity icon in the Excel Add-In in the Tasks section in the **OneStream** ribbon.



You can also access the Task Activity icon in the Cube View connection dialog box. This allows you to cancel while in the process of establishing a connection to the longer-running cube view.

> **NOTE:** This Task Activity icon in the Cube View connection dialog box is also featured in the Windows Application.





You must click the **Task Activity** icon to review all cube views. Click the **Running Task Progress** button to bring the Task Progress dialog box that displays an indeterminate progress bar. In the Task Progress dialog box, you can cancel the task by clicking the **Cancel Task** and **Close** buttons. The **Close** button does not cancel or close the actual task; it only closes the Task Progress dialog box for that specific task.

In Task Activity, the Task Status column will display whether the task was canceled by a User or Administrator. The Task Status also displays "Running" while the Task is still running and "Completed" if the task successfully completed without being canceled. If you are an Administrator, you have the option to show tasks for all users and show running tasks only. Non-admins have the option to show running tasks only. You can also can cancel another user's running task. Each task will have its task type, such as Quick View and Cube View.

Task Activity: Cancel Quick Views

In Task Activity, a User or Administrator can cancel a long-running Quick View.

In the OneStream spreadsheet and Excel Add-In,

1. Create a Quick View or Refresh a Quick View to cancel the long running Quick View. Information will begin to populate and your cursor will start to spin.
2. Click **Task Activity** and a Quick View task type displays in the Task Activity dialog box.
3. Use the **Running Task Progress** button to open the Task Progress dialog box.
4. In the Task Progress dialog box, it will display the indeterminate load bar. Use the **Cancel Task** button to cancel your task.
5. Then, click **Close** in the Task Progress dialog to close it, or leave it open for it to close on its own when the cancel is completed.
6. Click the **Refresh** button in the Task Activity dialog.
7. Click on **Child Steps** or **Task Information** to review the canceled task.

> **NOTE:** You can click **Child Steps** in the Task Activity to review all specific task steps for Quick Views or Cube Views.

Task Activity: Cancel XFGetCell Refresh

You can cancel an XFGetCell function conversion within Excel Add-In or OneStream spreadsheet.

1. In the Windows Application, under Tools, click **Spreadsheet**, click on **Cube Views** in the OneStream ribbon. In Excel Add-In, under the **OneStream ribbon**, click **Cube Views**.
2. Click **Cube View Connection** and add the cube view connection.
3. Once Cube View is loaded, click on the **Convert to XFGetCells** button in the OneStream ribbon or right click in the cube view and select **OneStream** > **Convert to XFGetCells**.
4. Click **Refresh Sheet**.
5. Click **Task Activity** and a Get Excel Data task type displays in the Task Activity dialog box.

6. Use the **Running Task Progress** button to open the Task Progress dialog box.
7. In the Task Progress dialog box, it will display a progress bar. Use the **Cancel Task** button to cancel your task.
8. Refresh the Task Activity dialog box. The Task Activity dialog box will display the task status if the user or administrator canceled the task.

> **NOTE:** In Task Activity, when canceling a task, a progress bar with a percentage load will display for XFGetCell canceled tasks.

Enhancing Task Activity: Cancel Cube Views

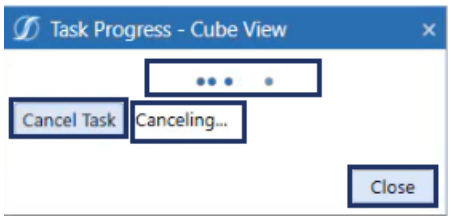
In Task Activity, a User or Administrator can track cube view progress and cancel a long-running cube view instead of resetting IIS.

> **IMPORTANT:** Cloud and self-hosted customers can cancel their own long-running cube views and turn on detailed logging to help support troubleshooting.

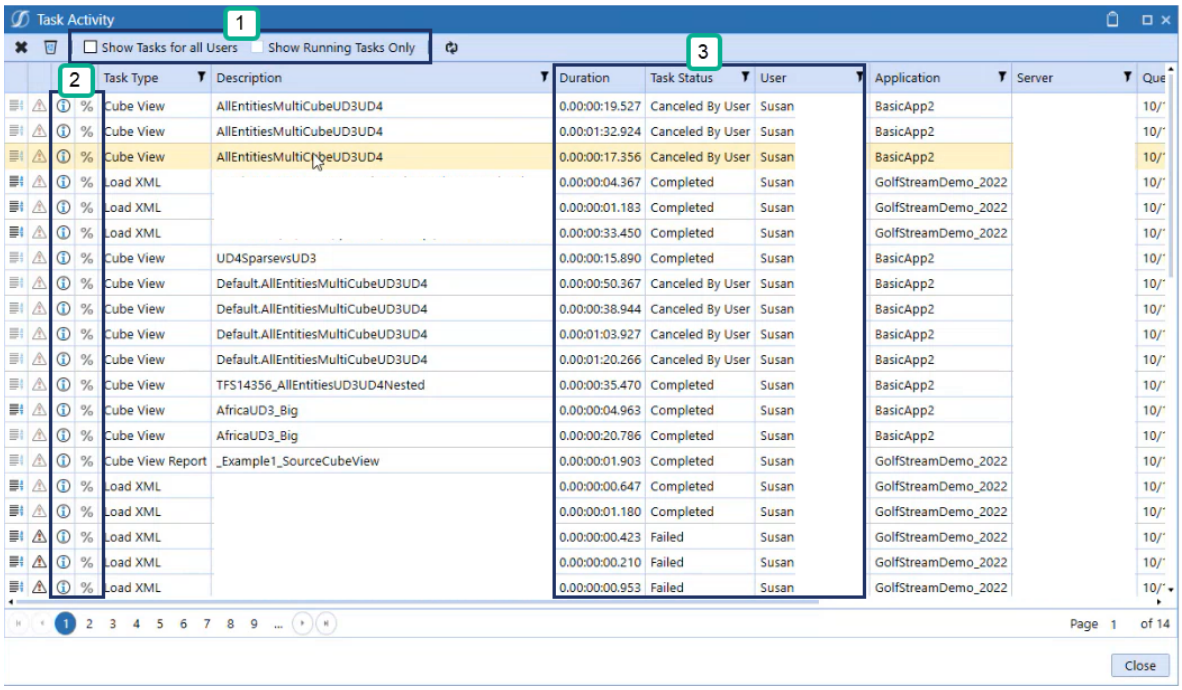
> **NOTE:** For all non-UI tasks, such as consolidation or other long-running tasks, the task activity icon will start blinking within a few minutes.

Cube View

When running a cube view through data explorer or clicking the refresh icon, if it takes longer than 10 seconds in the OneStream application, a dialog box will display an indeterminate progress bar and the ability to cancel the cube view or close the pop-up dialog through the **Cancel Task** and **Close** buttons. If you click the **Close** button, the dialog box will close, the Task Activity will blink (displaying to you that a task is running in the background), and the report will open when completed. If you click the **Cancel Task** button, a canceling message displays and the report will not run.



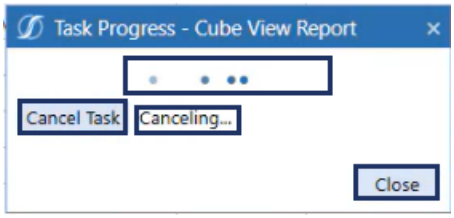
You can also click nothing when the dialog box displays. It will close on its own once the report loads. In Task Activity, the Task Status column displays whether the task was canceled by a User or Administrator. If you are an Administrator, you have the option to show tasks for all users and show running tasks only. You can also can cancel another user's running task. Non-admins have the option to show running tasks only. In Task Activity, the **Task Information** and **Running Task Progress** buttons are displayed inline with each record. You can also click **Running Task Progress** to cancel a task.



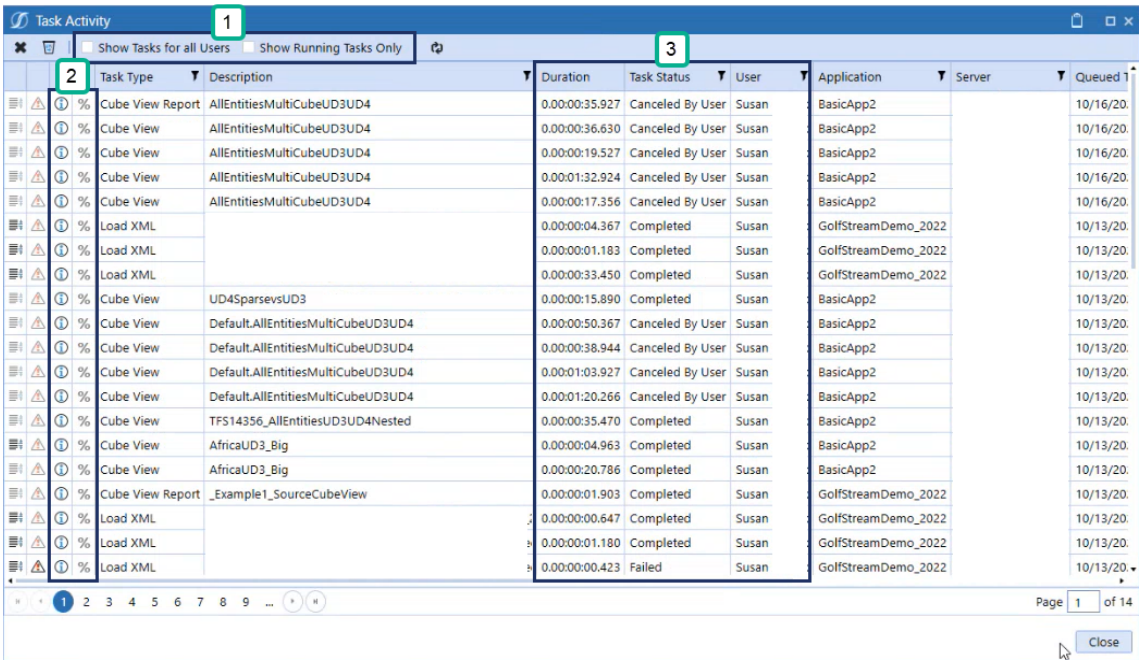
The Task Activity icon will blink when the cube view is taking longer than 10 seconds to complete and you are not in the Task Activity dialog box. The Task Activity icon will not blink if a task activity dialog box is open. Your cursor may spin after clicking the Task Activity icon. There may also be times when the task will complete and the task activity icon continues to blink. This should only last a few seconds or until you perform an action such as clicking in the application.

Show Report

When running a report (Show Report), if it takes longer than 10 seconds in the OneStream application, a dialog box will display an indeterminate progress bar and the ability to cancel the cube view or close the pop-up dialog through the **Cancel Task** and **Close** buttons. If you click the **Close** button, the dialog box will close, the Task Activity will blink (displaying to you that a task is running in the background), and the report will open when completed. If you click the **Cancel Task** button, a canceling message displays and the report will not run.



You can also click nothing when the dialog box displays. It will close on its own once the report loads. In Task Activity, the Task Status column displays whether the task was canceled by a User or Administrator. If you are an Administrator, you have the option to show tasks for all users and show running tasks only. You can also can cancel another user's running task. Non-admins have the option to show running tasks only. In Task Activity, the **Task Information** and **Running Task Progress** buttons are displayed inline with each record. You can also click **Running Task Progress** to cancel a task.

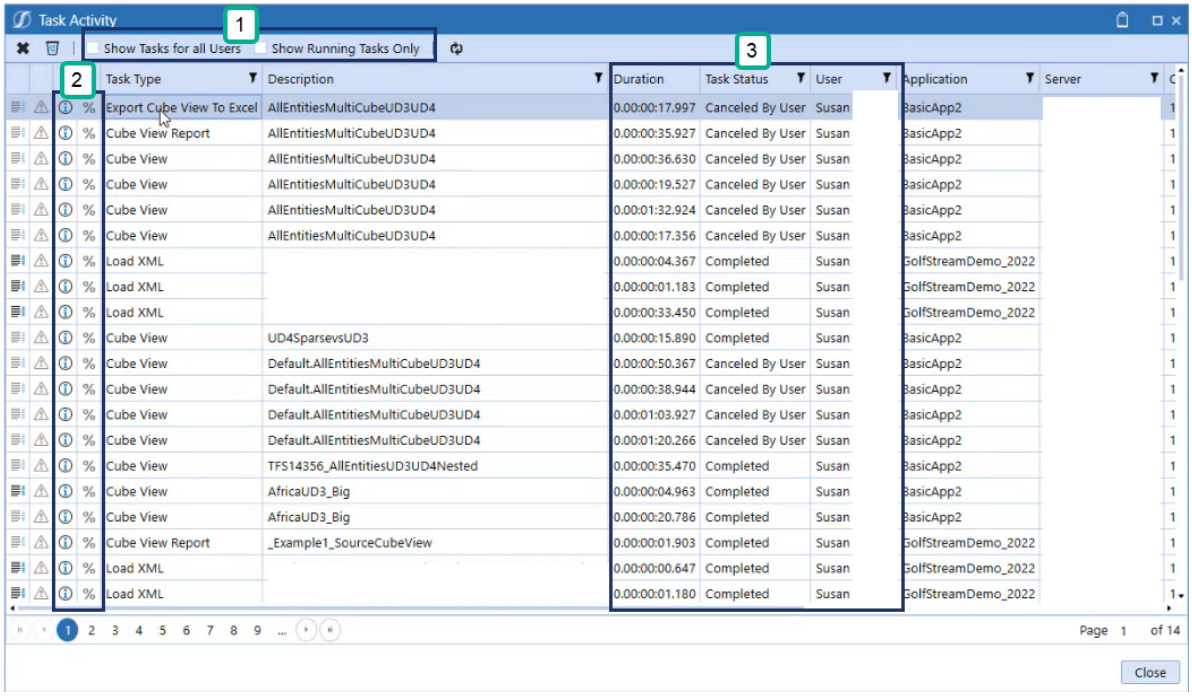


The Task Activity icon will blink when the cube view is taking longer than 10 seconds to complete and you are not in the Task Activity dialog box. The Task Activity icon will not blink if a task activity dialog box is open. Your cursor may spin after clicking the Task Activity icon. There may also be times when the task will complete and the task activity icon continues to blink. This should only last a few seconds or until you perform an action such as clicking in the application.

Export to Excel

When running an Export to Excel, if it takes longer than 10 seconds in the OneStream application, display an indeterminate progress bar and the ability to cancel the cube view or close the pop-up dialog through the **Cancel Task** and **Close** buttons. If you select the **Close** button, the dialog box will close, the Task Activity will blink (displaying to you that a task is running in the background) and the report will open once completed. If you select **Cancel Task**, a canceling message displays and Excel will not launch.

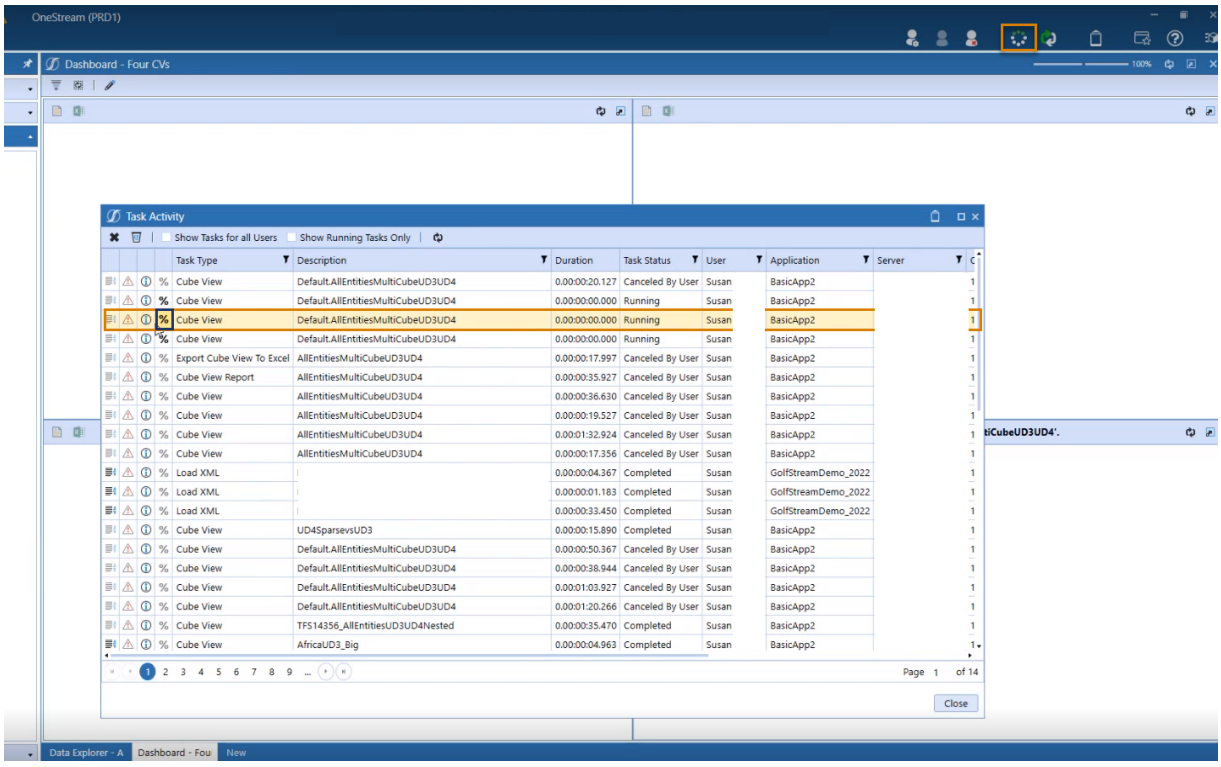
You can also click nothing when the dialog box displays. It will close on its own once the report loads. In Task Activity, the Task Status column displays whether the task was canceled by a User or Administrator. If you are an Administrator, you have the option to show tasks for all users and show running tasks only. You can also can cancel another user's running task. Non-admins have the option to show running tasks only. In Task Activity, the **Task Information** and **Running Task Progress** buttons are displayed inline with each record. You can also click **Running Task Progress** to cancel a task.



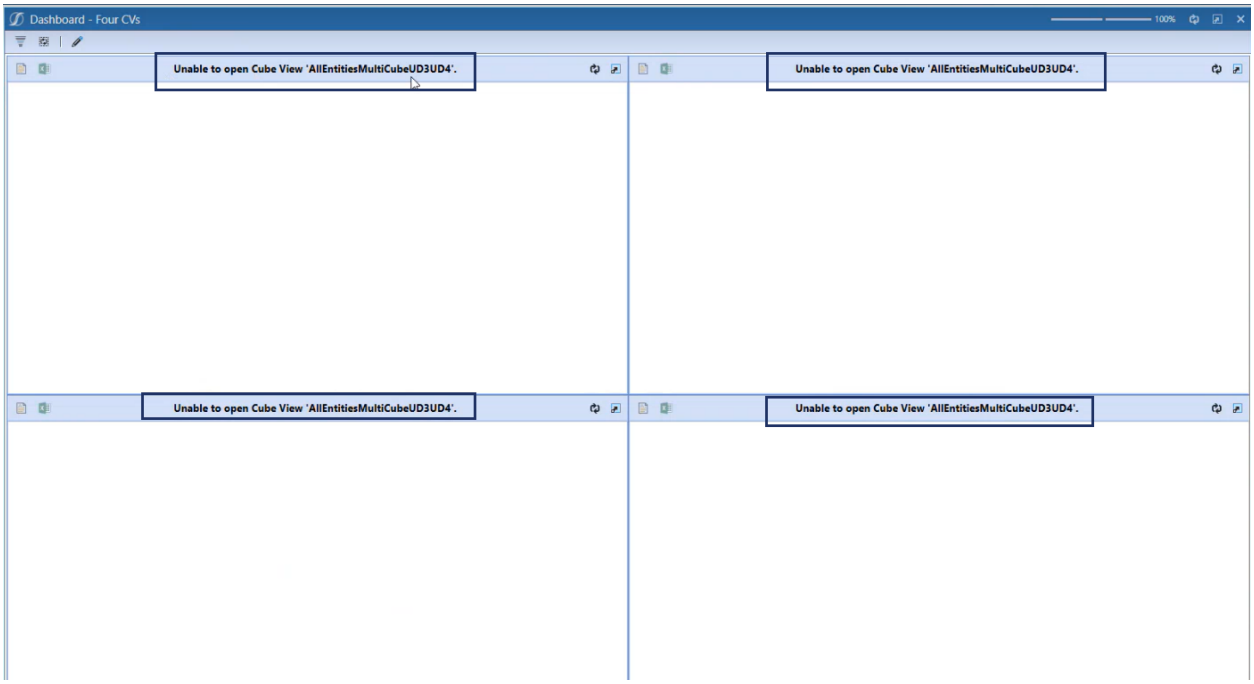
The Task Activity icon will blink when the cube view is taking longer than 10 seconds to complete and you are not in the Task Activity dialog box. The Task Activity icon will not blink if a task activity dialog box is open. Your cursor may spin after clicking the Task Activity icon. There may also be times when the task will complete and the task activity icon continues to blink. This should only last a few seconds or until you perform an action such as clicking in the application.

Dashboard with Cube View Components

When running a dashboard with a single cube view component or multiple cube view components, if it takes longer than 10 seconds, you will not see a pop-up dialog box. You must click **Task Activity** to review all cube views. You also can cancel each individual cube view by clicking the **Running Task Progress** button to cancel your tasks.



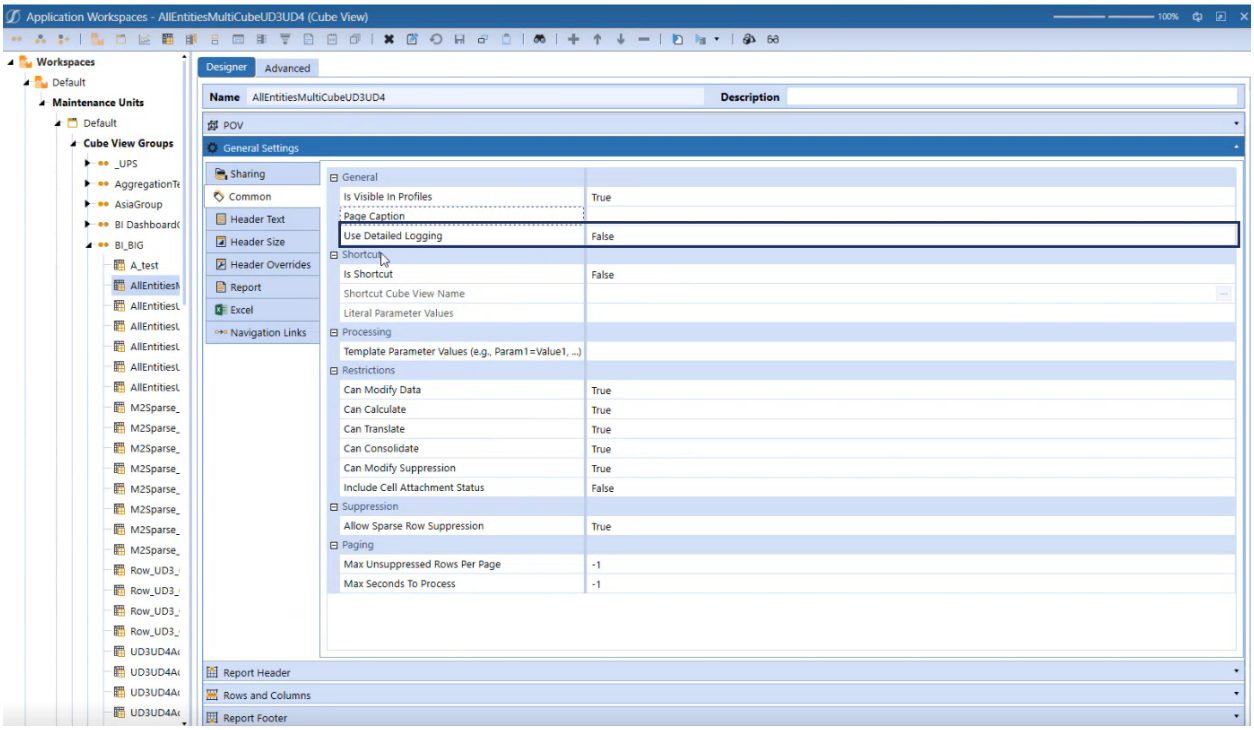
To verify if a cube view has been canceled, a message displays that it cannot open the cube view.



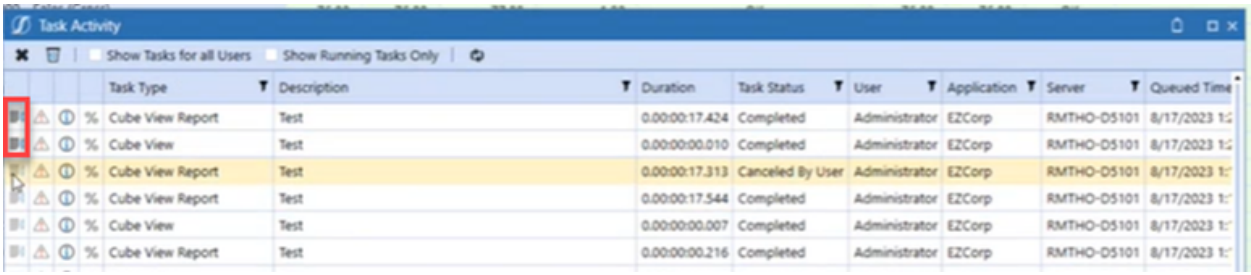
Detailed Logging

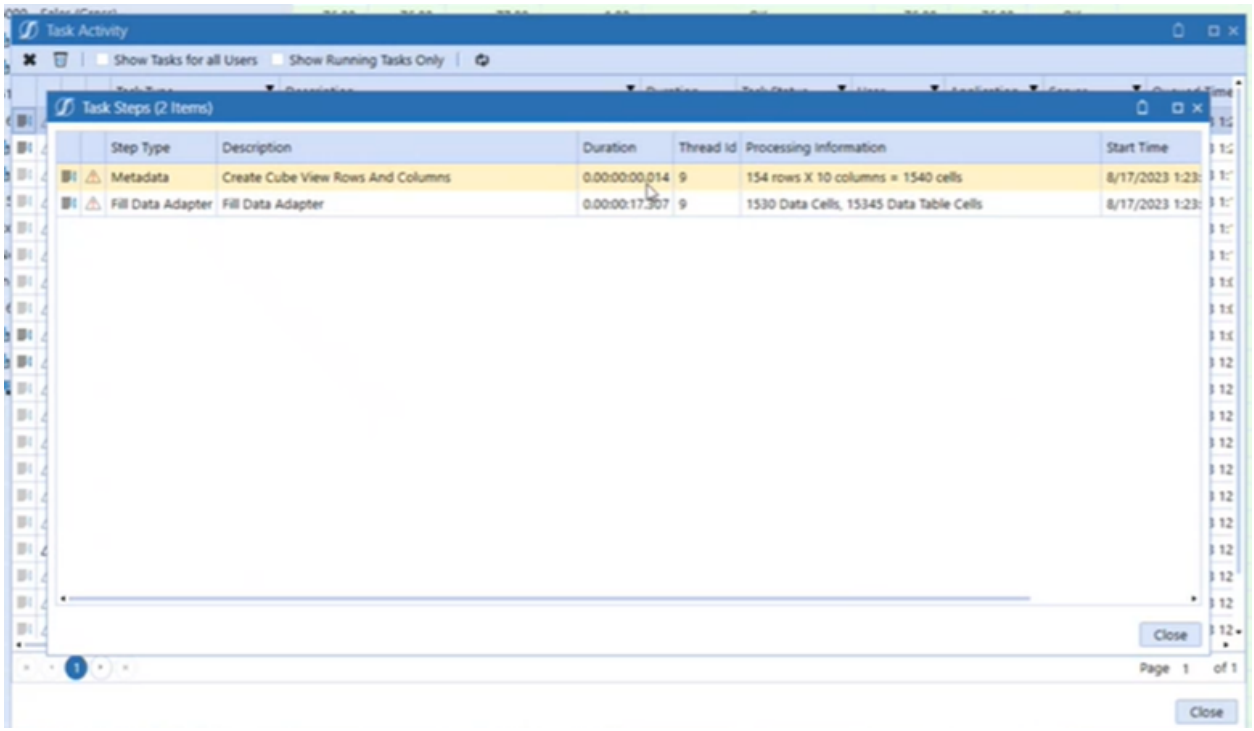
Use Detailed Logging has been removed from the App Server Config file, TALogCubeViews from the Task Activity section, and added to individual cube views. This setting can be found in both the Designer and Advanced tabs for Cube Views. It allows you to turn on detailed logging to gather more information about your cube views, such as data cells and any additional information in Task Activity.

1. On the **Application** tab, under **Presentation**, click **Workspaces**.
2. In the **Application Workspaces** pane, under **Workspaces**, expand **Default** > **Maintenance Unit** > **Default** > **Cube View Groups**.
3. Select a cube view under the **Cube View Groups** to edit.
4. Click the **Designer** tab and select **Common** under General settings.
5. For **Use Detailed Logging**, the default is set to False.



NOTE: When set to True, detailed logging will provide information in Task Activity, such as individual steps and additional information about the cube view.



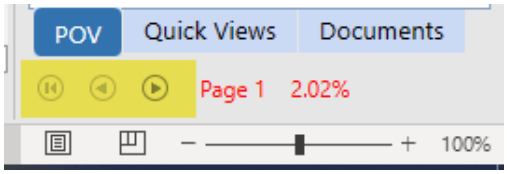


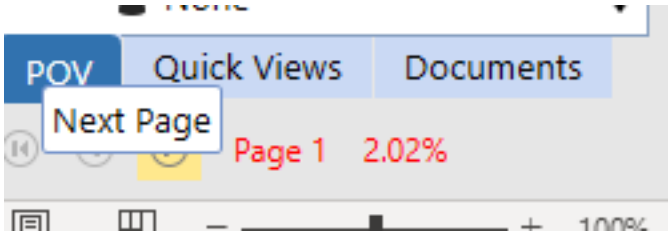
Cube View Paging

You can cancel a long running cube view that contains paging in the Excel Add-in or Spreadsheet. This allows you to cancel a single page using Task Activity instead of waiting for the entire cube view to load and process. A user or admin can cancel the cube view within the Excel Add-in or Spreadsheet by canceling from each page in Task Activity or during the initial load of the cube view.

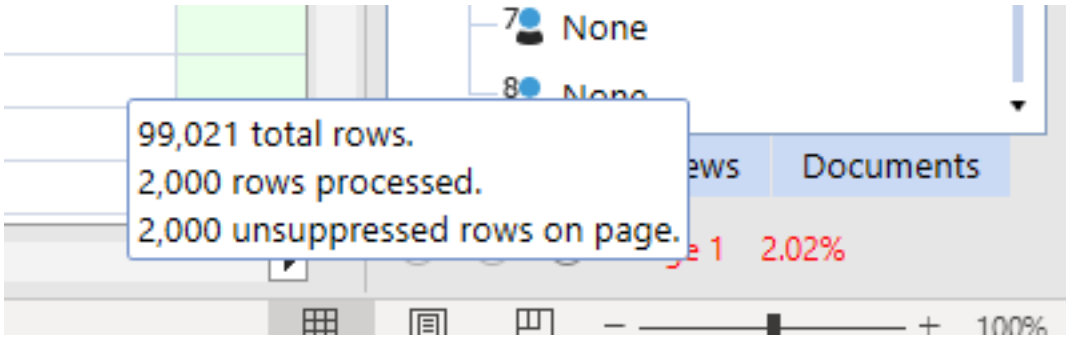
Paging Controls

You can navigate between pages by moving forward or back using the arrow keys (first page, previous page, and next page). Each page will display the page number and data loaded in a percentage below the tabs in the right-hand corner.





The paging tool tip shows the number of total rows, rows processed, and unsuppressed rows on a page.



Connect to a Cube View with Paging

You can connect to a cube view with paging within Excel Add-In or OneStream spreadsheet. Any user can use cube view paging in Excel Add-In or Spreadsheet.

1. In the Windows Application, under Tools, click **Spreadsheet**, click on **Cube Views** in the OneStream ribbon. In Excel Add-In, under the **OneStream ribbon**, click **Cube Views**.
2. Click **Cube View Connections**to add the cube view connection.

3. When the Cube View Connection dialog box appears, click **Add** button.
4. In the Cube View connection dialog box, select the ellipses (...) next to the Cube View field.
5. In the Object Lookup, use the search field to search for your Cube View, then select it in the hierarchy. Then click **OK.**

> **NOTE:** It may take a long time for the parameter dialog to appear for Cube Views containing parameters. You will need to enter or select a specific value after clicking **OK** and selecting the Cube View.

6. Click **Close** in Cube View connections dialog.
7. Click in the Cube View, on the right-hand corner, the paging controls will display.
8. You can navigate between pages with the arrow keys and data loaded will display within each page in a percentage.

Load an Excel File with Cube View Paging

You can load an Excel file that contains a Cube View that needs to be paged in Excel Add-In or Spreadsheet.

1. In the Windows Application, under Tools, click **Spreadsheet**, click on **File**>**Open** in the OneStream ribbon. In Excel Add-In, in the ribbon, go to **File**, click **Open.**
2. Select a file from your local folder or OneStream File System and click **Open.**
3. Click the **Refresh Sheet** button in the OneStream ribbon.
4. The Cube View data will refresh, on the right-hand corner, the paging controls will display.
5. You can navigate between pages with the arrow keys. The percentage of data loaded will be displayed next to the paging controls.

Task Activity: Cancel Cube View with Paging

In Task Activity, a user or administrator can cancel a Cube View in Task Activity within Excel Add-In or Spreadsheet.

1. In the Windows Application, under Tools, click **Spreadsheet**, click on **Cube Views** in the OneStream ribbon. In Excel Add-In, under the **OneStream ribbon**, click **Cube Views**.
2. Click **Cube View Connection** and add the Cube View connection.
3. When the Cube View Connection dialog box appears, click **Add** button.
4. In the Cube View connection dialog box, select the ellipses (...) next to Cube Views.
5. In Object Lookup, select the Cube View from the hierarchy, and click **OK**.
6. Click **Close** in Cube View connections dialog.
7. When you click in the Cube View, on the right-hand corner, the paging controls will display. Click the **Next Page** button.

   > **NOTE:** If you click on the paging controls to go to the next page and the load time takes more than 10 seconds, the dialog box will appear allowing you to cancel the Cube View.

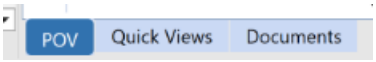
8. Click on the **Task Activity** button and a Cube View task type displays in the Task Activity dialog box.
9. Use the **Running Task Progress** button to open the Task Progress dialog box.
10. In the Task Progress dialog box, it will display the indeterminate load bar. Use the **Cancel Task** button to cancel your task.
11. Refresh the Task Activity dialog box. The Task Activity dialog box will display the task status if the user or administrator canceled the task.

> **NOTE:** Canceling the Next Page load for a Cube View/Quick View will display the #REFRESH text in the data cells, rather than the page blank for the Initial Load.

Quick View Paging

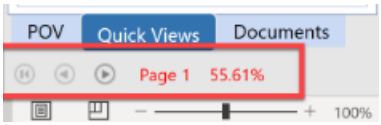
You can cancel a Quick View with Paging in the Excel Add-In or Spreadsheet. This allows you to cancel the Quick View without having to wait for all pages to load and process on the server. A user or admin can cancel the loading of a Quick View page from the Excel Add-In or Spreadsheet from Task Activity.

> **NOTE:** Previously, Quick View paging controls appeared in the Quick Views tab. It is now located below all three tabs.

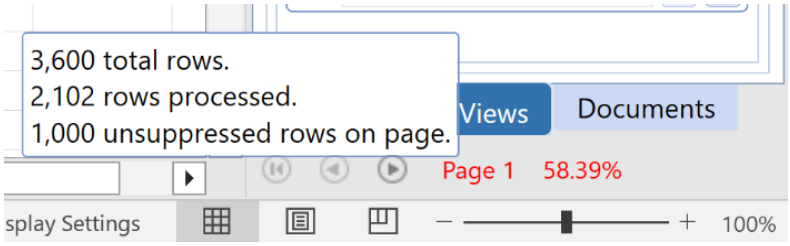


Paging Controls

You can navigate between pages by moving forward or back with the arrow keys (first page, previous page, and next page). Each page will display the page number and data loaded in a percentage below the tabs on the right-hand corner.



The paging tool tip shows the number of total rows, \ rows processed, and unsuppressed rows on a page.



Create a Quick View with Quick View Paging

You can create a quick view with quick view paging within Excel Add-In or Spreadsheet. Any user can use quick view paging in Excel Add-In or Spreadsheet.

1. In the Windows Application, under Tools, click **Spreadsheet**, click on **Quick Views** >**Create Quick View** option in the OneStream ribbon. In Excel Add-In, under the **OneStream ribbon**, click **Quick Views**.
2. Click **Create Quick View** button in the Quick View pane to create a new quick view.

   > **NOTE:** See Create and Modify Quick View for more information.

3. In the Create Quick View dialog, accept the default or insert a specific Name and then click **OK**.
4. Modify your Quick View to add and remove dimensions and members to set up your Quick View.

   > **NOTE:** See Modify Quick View for more information.

5. When making modifications and applying changes, the data will update to reflect all changes.
6. When a Quick View requires paging, the paging controls will display in the right-hand corner.

7. You can navigate between pages with the arrow keys. The percentage of data loaded will be displayed next to the paging controls.

Load an Excel File for Quick View Paging

You can load an Excel file that contains a Quick View with Paging for Excel Add-In or Spreadsheet.

1. In the Windows Application, under Tools, click **Spreadsheet**, click on **File**>**Open** in the OneStream ribbon. In Excel Add-In, in the ribbon, go to **File**, click **Open.**
2. Select a file from your local folder and click **Open.**
3. Click the **Refresh Workbook** button in the OneStream ribbon.
4. The Worksheet and data will refresh, on the right-hand corner, the paging controls will display.
5. You can navigate between pages with the arrow keys. The percentage of data loaded will be displayed next to the paging controls.

Task Activity: Cancel Quick View Paging

In Task Activity, a user or administrator can cancel a quick view with paging within Excel Add-In or Spreadsheet.

1. In the Windows Application, under Tools, click **Spreadsheet**, click on **Quick Views** in the OneStream ribbon. In Excel Add-In, under the **OneStream ribbon**, click **Quick Views**.
2. Click **Create Quick View** button in the Quick View pane to create a new quick view.

   > **NOTE:** See Create and Modify Quick View for more information.

3. In the Create Quick View dialog, accept the default or insert a specific Name and click **OK**.

4. Add or Remove dimensions and members to modify and set up your Quick View.

   > **NOTE:** See Modify Quick View for more information.

5. When making modifications and applying changes, the data will update to reflect all changes.
6. When a Quick View requires paging, the paging controls will display in the lower right-hand corner. Click the **Next Page** button.
7. Click on the **Task Activity** button and a Quick View task type displays in the Task Activity dialog box.
8. Use the **Running Task Progress** button to open the Task Progress dialog box.
9. In the Task Progress dialog box, it will display the indeterminate load bar. Use the **Cancel Task** button to cancel your task.
10. Refresh the Task Activity dialog box. The Task Activity dialog box will display the task status if the user or administrator canceled the task.

> **NOTE:** Canceling the Next Page load for a Cube View/Quick View will display the #REFRESH text in the data cells, rather than the page blank for the Initial Load.

Error Logs

Administrators can use the following to evaluate errors on System > Logging.

Clear error log for current user

Clears all logs for a user.

Clear error log for all users

Clears all logs for all users.

This grid will display:

Description

Displays a brief description of the error.

Error Time

Indicates when the error occurred.

Error Level

Displays the error type such as Unknown, Fatal, Warning etc.

User

Displays the user ID.

Application

Identifies the application a user is logged onto.

Tier

Displays the application tier such as App Server, Web Server or Client.

App Server

Identifies the application server to which a user was connected when they encountered an error.

System Tools

There are a variety of system tools that can be used to manage the OneStream application. These tools include system business rules, database, tools that allow you to check the overall health of the application environment, and the File Explorer to name a few. In this section you will learn how to use these and other tools to manage the application system.

System Business Rules

You can use System Extender Business Rules with Azure Server Sets for enhanced scalability at the Azure Database and Server Sets level. You can perform Server and eDTU scaling manually or using System Business Rules. If System Business Rules is selected as a Scaling Type, then a user-defined System Extender Business Rule determines if scaling is needed. You must implement the scaling function and return the proper scaling object by adding a System Extender Business Rule and assigning it appropriately.

Under each Case statement, use these rules and related Args and BRApis to check the current Server Set capacity, query metrics about a Server Set or Azure Database, and identify the impact of the Server Sets volume or level of Azure Database deployed. See "Azure Database Connection Setting and Server Sets" in the *Installation and Configuration Guide*.

Example starting point of empty System Extender Business Rule upon creation:

```
Namespace OneStream.BusinessRule.SystemExtender.ServerSet2
    Public Class MainClass
        Public Function Main(ByVal si As SessionInfo, ByVal globals As BRGlobals, ByVal api As Object, ByVal args As SystemExtenderArgs) As Object
            Try
                Select Case args.FunctionType

                    Case Is = SystemExtenderFunctionType.Unknown

                    Case Is = SystemExtenderFunctionType.GetDesiredServerSetCapacity

                    Case Is = SystemExtenderFunctionType.GetDesiredElasticDatabasePoolCapacity

                    Case Is = SystemExtenderFunctionType.GetDesiredExternalServerSetCapacity

                End Select

                Return Nothing
            Catch ex As Exception
                Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
            End Try
        End Function
    End Class
End Namespace
```

Sample System Business Rule

Metric are passed to this function to help users determine if a server or database needs scaling. For server scaling, Environment metrics and Scale Set metrics are passed. For database scaling, Environment metrics and SQL Server Elastic Pool metrics are passed.

```
Select Case args.FunctionType

    Case Is = SystemExtenderFunctionType.Unknown

    Case Is = SystemExtenderFunctionType.GetDesiredScaleSetCapacity
        Dim systemExtenderScaleSetResult As New SystemExtenderScaleSetResult
        systemExtenderScaleSetResult.Capacity = args.ScaleSetArgs.CurrentScaleSetCapacity

        If (args.ScaleSetArgs.ScaleSetMetricValues.AvgCPUUtilization > 50) Then
            systemExtenderScaleSetResult.Capacity = args.ScaleSetArgs.CurrentScaleSetCapacity + 1
        End If

        Return systemExtenderScaleSetResult

    Case Is = SystemExtenderFunctionType.GetDesiredElasticDatabasePoolCapacity
        Dim systemExtenderSQLServerElasticPoolResult As New SystemExtenderSQLServerElasticPoolResult
        systemExtenderSQLServerElasticPoolResult.AzureElasticPoolDTU = args.SQLServerElasticPoolArgs.DatabaseAndEPoolDTU.AzureElasticPoolDTU

        If (args.SQLServerElasticPoolArgs.AzureElasticPoolLevelMetricValues.DTUConsumptionPercent > 90)
            systemExtenderSQLServerElasticPoolResult.AzureElasticPoolDTU = 1600
        End If

        Return systemExtenderSQLServerElasticPoolResult

    Case Is = SystemExtenderFunctionType.GetDesiredExternalScaleSetCapacity

End Select
```

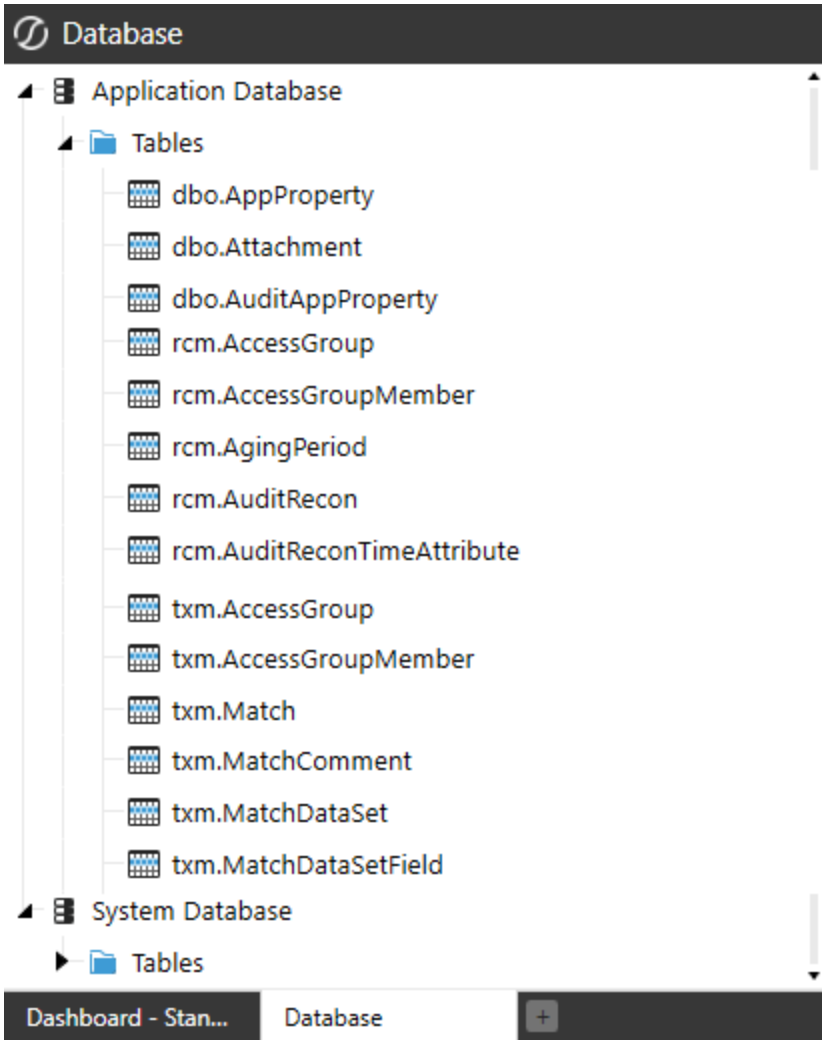
Database

The **Database** screen allows System Administrators to view database tables and provides tools for managing stored data and other information. To view the **Database** screen, go to **System > Tools > Database**.

Database Tables

Go to **Application Database > Tables** or **System Database > Tables** to view all tables in the Application Database or System Database. This view provides read-only access to the data tables that can be used for debugging tasks that do not affect the database or system logs.

Data tables that are imported through MarketPlace Solutions have their schema name added to the beginning of the database table name and replace dbo (for example, rcm.AccessGroup and txm.AccessGroup).



Tools

Database Tools allow System Administrators to manage the database.

Data Records

Go to **Application Database > Tools > Data Records** to view data for the entire system. Filter the data using the member filter.

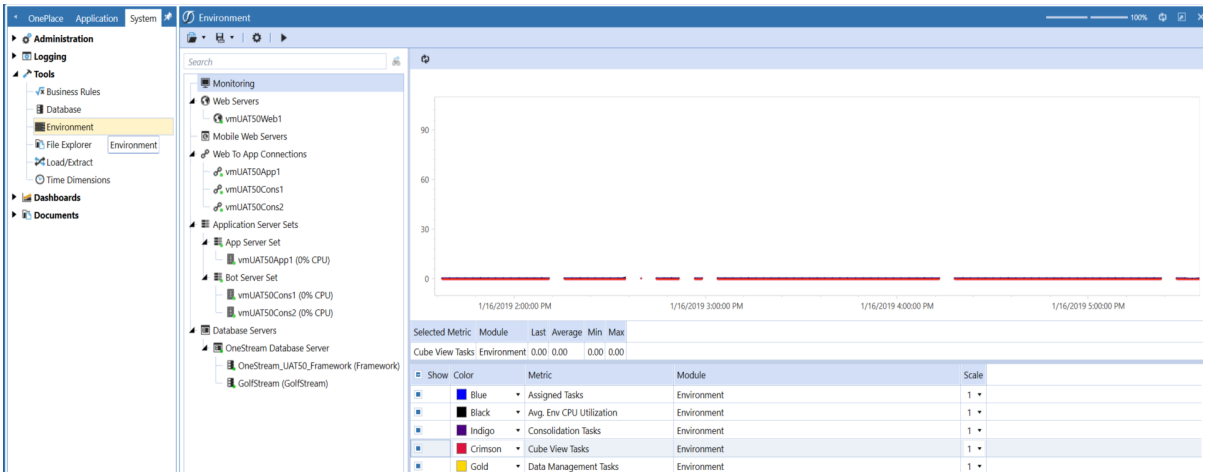
Environment

This section can be used to check the overall health of the environment, which contains Web Servers, Mobile Web Servers, Application Server Sets and Database Servers. This will check the connection status as well as the configuration.

The Environment page is designed to give both IT and power business users a way to manage and optimize their applications and the environment that is running under. Using the Environment page, the user can monitor the environment, isolate bottlenecks, look at properties and configuration changes, and scale in/out application servers and database resources if needed. They can also customize what data to collect in log files, save collection metrics files, and replay collected performance data in many ways.

To access the Environment page, select **System > Tools > Environment**.

> **NOTE:** The Environment page is only accessible via the OneStream Windows App.



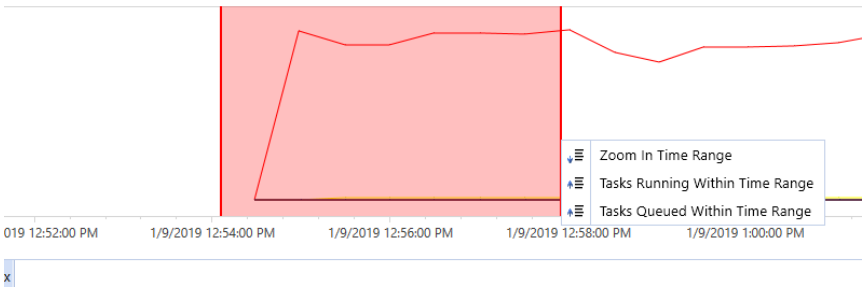
Monitoring Environments

Instead of logging onto the server to collect metrics, use the Monitoring page to access real time Key Performance Indicators (KPIs) across an environment. Click **System** > **Tools** > **Environment** > **Monitoring** to:

- Access user activity and interactive KPI graphs.
- Track all system components that affect stability and performance.

You can perform these tasks:

- **Open**: Access a metrics and configuration setting file from the File System or a local folder.
- **Save As**: Save metrics and configurations locally or in the File System.
- **Settings**: Specify these types of KPI metrics to monitor and the monitor frequency:
  - Environment
  - Application Servers
  - Database Servers
  - Server Sets
- Zoom into part of the chart to see running or queued activities.



- **Refresh Automatically Using a Timer**: Retrieve metrics based on the Play Update Frequency interval setting.

Specifying General Settings

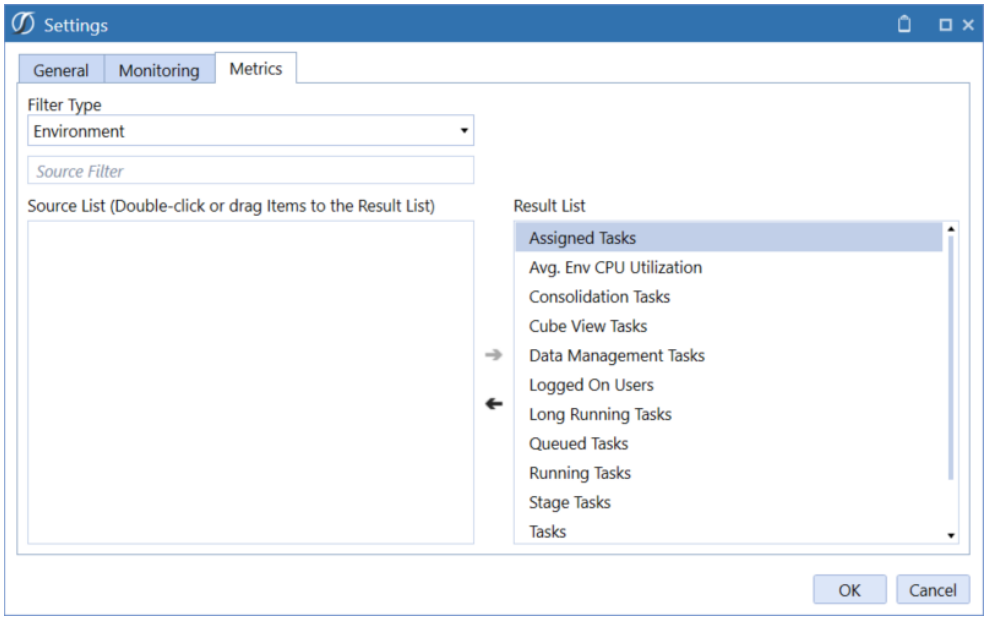
- **General Play Update Frequency (seconds):** How often to update Performance charts.
- **Metric and Task Time Range**: Indicate how much historical data to retrieve and depict on the Performance Chart. Applying a time range can help identify the cause of a server event or issue.
- **Y-Axis**: If Auto Range box is selected, the system sets the range for the Min and Max values on the Y-Axis. If cleared, you can set the Min and Max ranges on the Y-Axis.
- **Secondary Y-Axis**: Displays series of a different range of values, or different arguments (or values) in the same chart. Can be used when the numbers in a chart vary widely from data series to data series, or when there are mixed types of data.

Specifying Monitoring Settings

- **Monitoring(Server filter)**: Identify specific application or database servers to evaluate.
- **Filter Type**: Specify the type of servers for which to collect metrics.
- **Source Filter**: Use to filter the Source List.
- **Source List**: Displays the servers that meet your criteria.
- **Result List**: Displays selected **Source List** items for which to collect metrics.

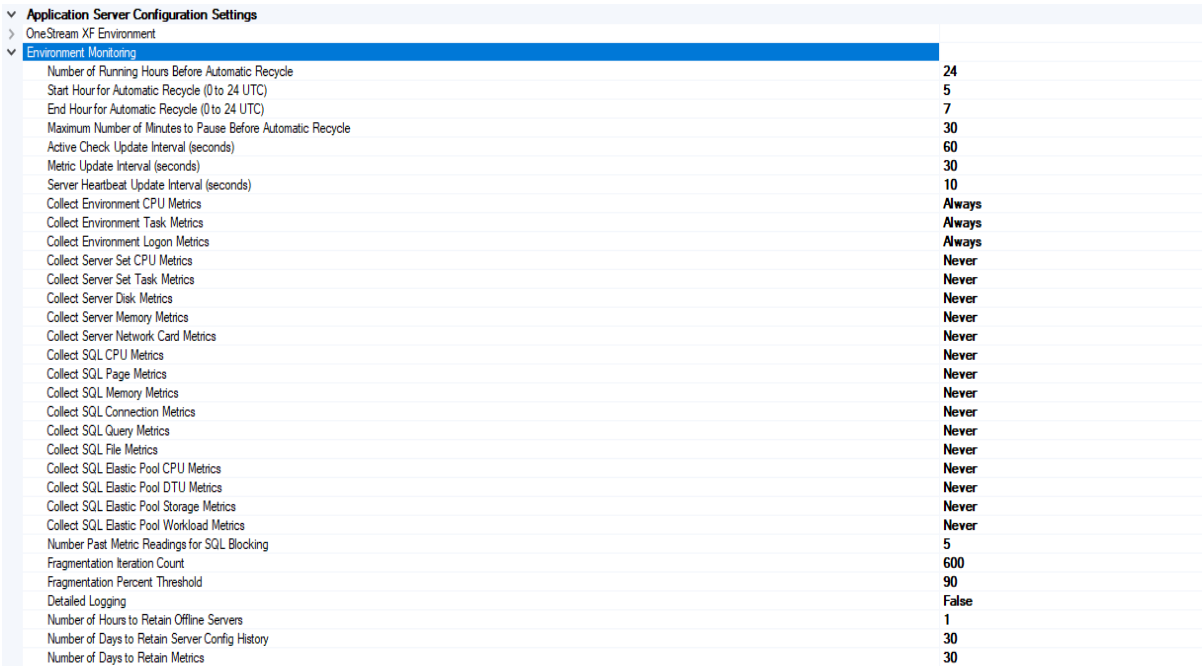
Specifying Metrics Settings

Metrics are collected based on the Metric Update Interval (seconds) setting in the Application Server Configuration utility. To minimize database access and maximize performance, some metrics are collected on every iteration, but some will skip iterations based on count settings defined for a metric.



- **Filter Type**: Refine the types of application servers, database servers or server sets for which to collect metrics.
- **Source Filter**: Use to filter the Source List.
- **Source List**: Displays the list of filtered application servers, database servers or server sets.
- **Result List**: Displays the items for which to collect metrics.

These settings shown below in the Application Server Configuration File in Environment Monitoring, determine which metrics are collected and how often.



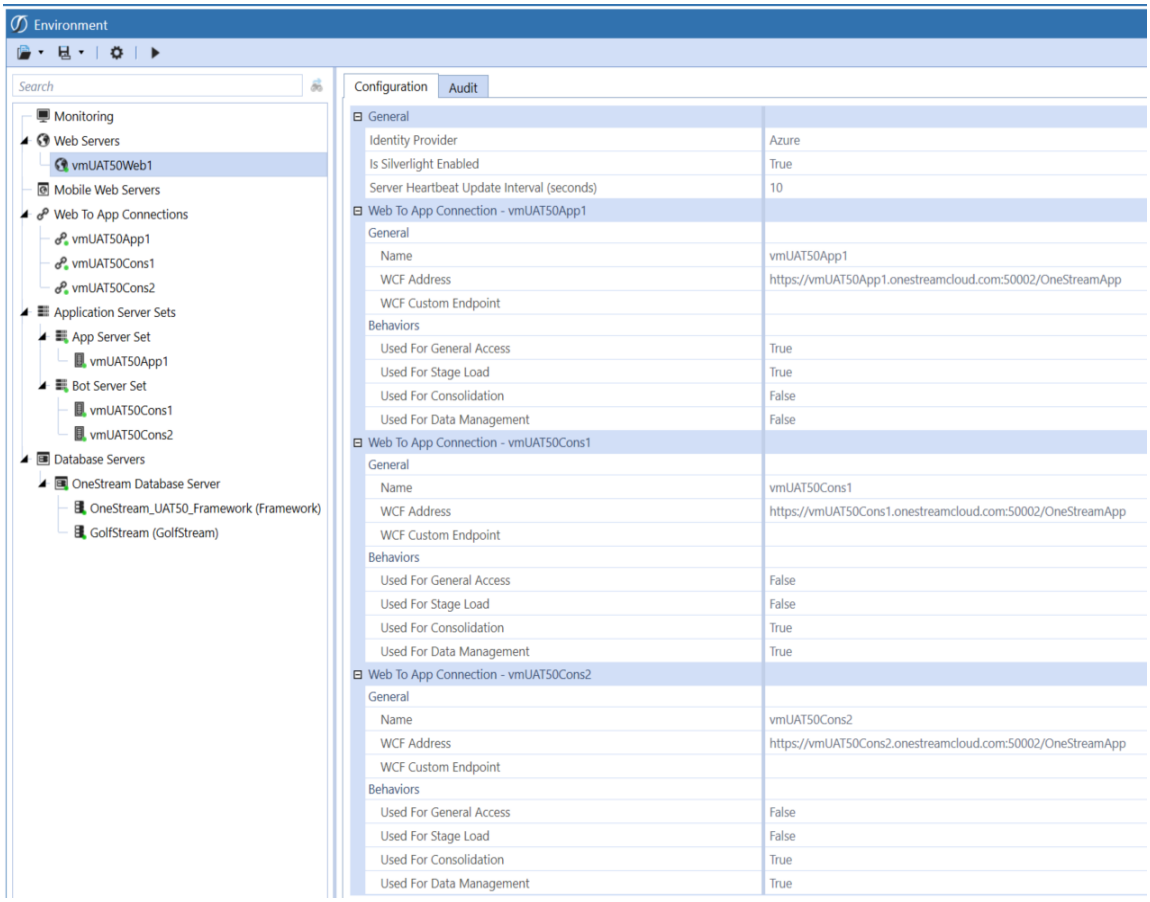
Application Server Configuration Settings	
OneStream XF Environment	
Environment Monitoring	
Number of Running Hours Before Automatic Recycle	24
Start Hour for Automatic Recycle (0 to 24 UTC)	5
End Hour for Automatic Recycle (0 to 24 UTC)	7
Maximum Number of Minutes to Pause Before Automatic Recycle	30
Active Check Update Interval (seconds)	60
Metric Update Interval (seconds)	30
Server Heartbeat Update Interval (seconds)	10
Collect Environment CPU Metrics	Always
Collect Environment Task Metrics	Always
Collect Environment Logon Metrics	Always
Collect Server Set CPU Metrics	Never
Collect Server Set Task Metrics	Never
Collect Server Disk Metrics	Never
Collect Server Memory Metrics	Never
Collect Server Network Card Metrics	Never
Collect SQL CPU Metrics	Never
Collect SQL Page Metrics	Never
Collect SQL Memory Metrics	Never
Collect SQL Connection Metrics	Never
Collect SQL Query Metrics	Never
Collect SQL File Metrics	Never
Collect SQL Elastic Pool CPU Metrics	Never
Collect SQL Elastic Pool DTU Metrics	Never
Collect SQL Elastic Pool Storage Metrics	Never
Collect SQL Elastic Pool Workload Metrics	Never
Number Past Metric Readings for SQL Blocking	5
Fragmentation Iteration Count	600
Fragmentation Percent Threshold	90
Detailed Logging	False
Number of Hours to Retain Offline Servers	1
Number of Days to Retain Server Config History	30
Number of Days to Retain Metrics	30

Web Servers

This section lists all the web servers that are participating in that environment. Each web server will display its configuration and its audited setting.

Configuration

Sample Web Server configuration settings:



Identity Provider

Displays single sign on identity provider if used.

Server Heartbeat Update Interval (seconds)

Used to specify how often each server updates its record that it is alive and responding to user input.

Name

This is the name of the server as it is defined in the web configuration file.

WCF Address

The full URL address of the server.

WCF Connection

This examines the status of the connection. Ok = connected

Used for General Access

This examines the web configuration file to see if it has this server configured as a general server. Settings are True or False.

Used for Stage Load

This examines the web configuration file to see if this server is configured as a Stage server. Settings are True or False.

Used for Consolidation

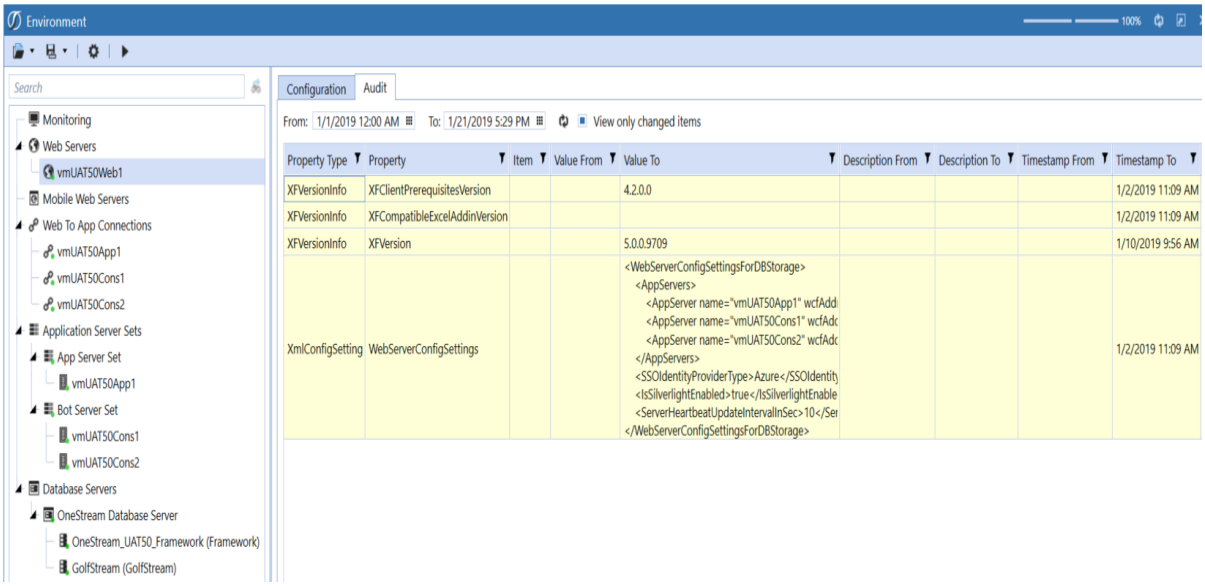
This examines the web configuration file to see if this server is configured as a consolidation server. Settings are True or False.

Used For Data Management

This examines the web configuration file to see if this server is configured as a Data Management server. Settings are True or False.

Audit

Sample Web Server audit setting:



Property Type

Type of server property in the hardware or software that was changed.

Property Name

The name of the property in the hardware or software that was changed.

Value From

Displays the original value of the property.

Value To

Displays the new value of the property.

Description From

Displays the original description of the setting if available.

Description To

Displays the new description of the setting if available.

Timestamp From

Displays the original date and time of the setting if available.

Timestamp To

Displays the new date and time of the setting.

User From

Displays the original user name if available.

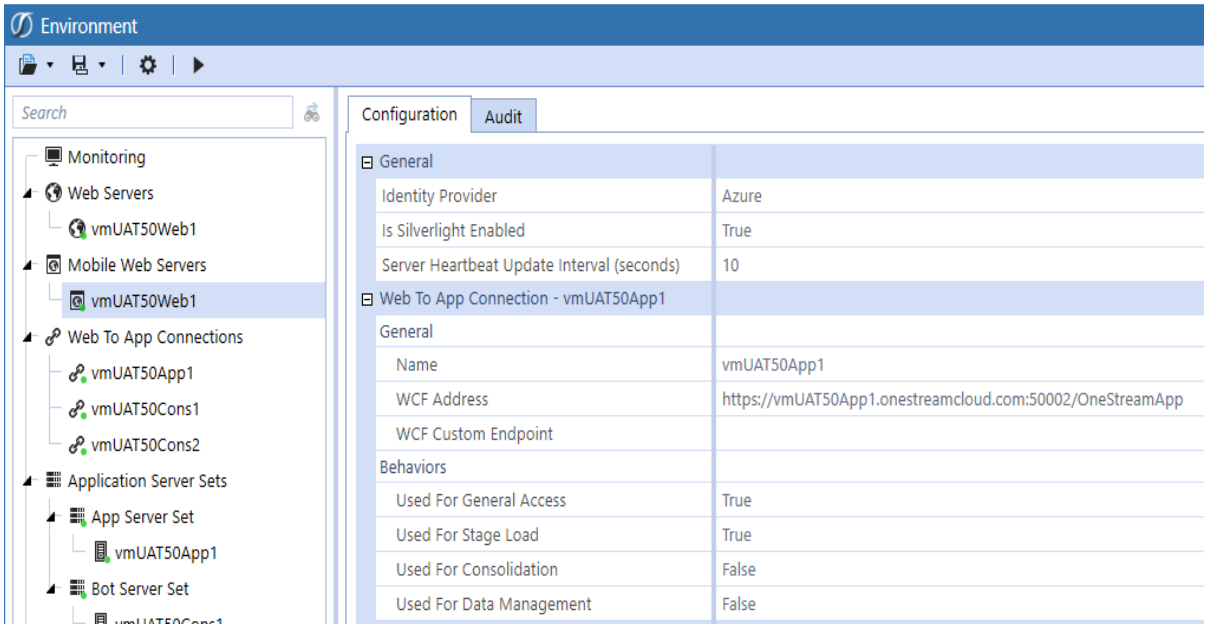
User To

Displays the user name of the person that made the change if available.

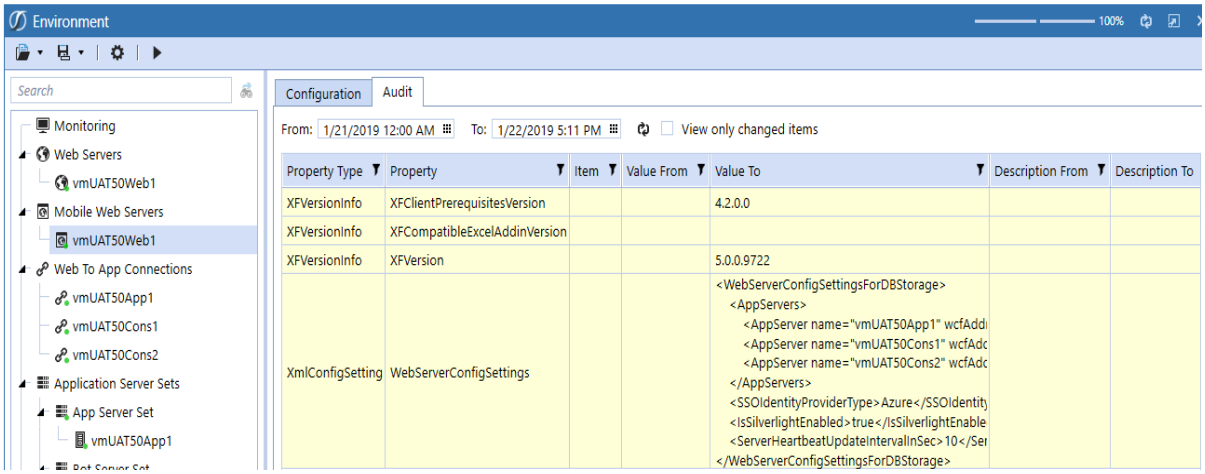
Mobile Web Servers

Similar to web server, the mobile web servers list all the web servers that are included in that environment. It pulls the information from the ServerHeartBeat table in the Framework database.

Sample Mobile Web Server Configuration



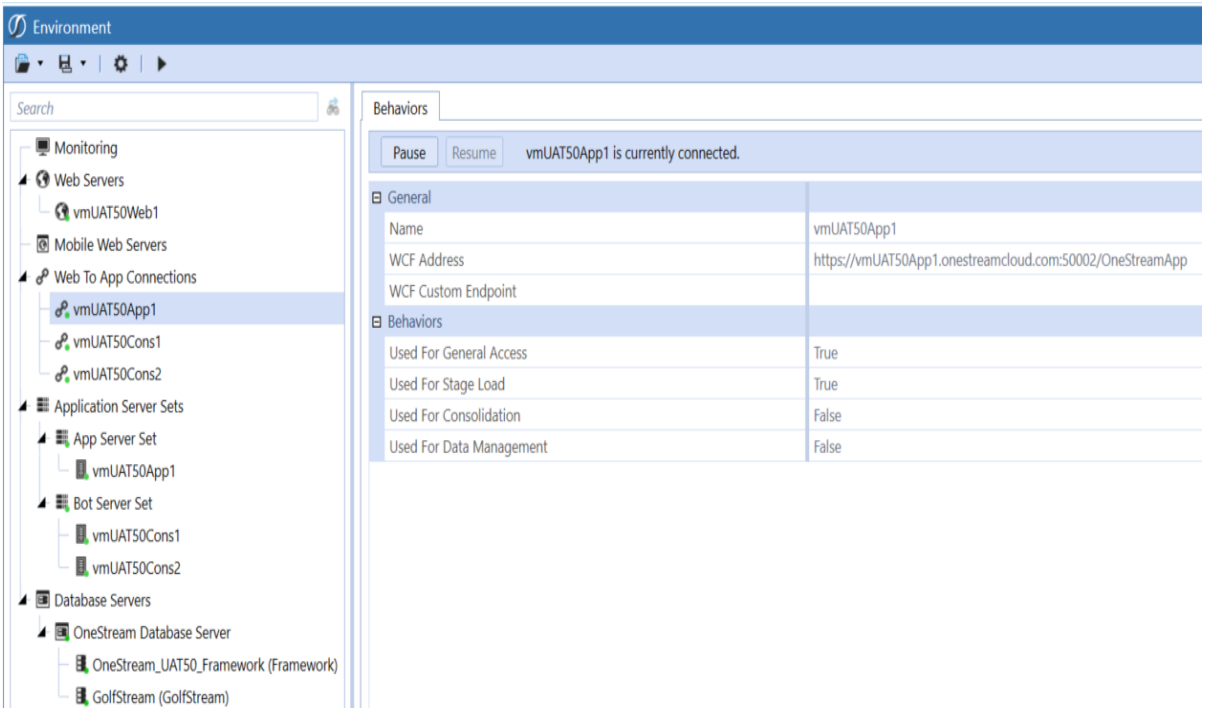
Sample Mobile Web Server Audit



WebTo App Server Connections

This section will list all the combined connections from the web server configuration files to all the application servers. From here the user can pause and then resume a specific connection to an application server or a load balancer (a load balancer could point to a multiple application server). See Installation and Configuration Guide.

Sample Web to App connection configuration:



Pause

Used to pause any request to a WCF Address connection. This connection could be either an Application Sever or a Load Balancer. This can be set in the web configuration file.

Resume

Used to Resume any request to a WCF Address connection. This connection could be either an Application Sever or a Load Balancer. This can be set in the web configuration file.

Name

This is the name of the server as it is defined in the web configuration file.

WCF Address

The full URL address of the server.

WCF Connection

This examines the status of the connection. Ok = connected

Used for General Access

This examines the web configuration file to see if it has this server configured as a general server. Settings are True or False.

Used for Stage Load

This examines the web configuration file to see if this server is configured as a Stage server. Settings are True or False.

Used for Consolidation

This examines the web configuration file to see if this server is configured as a consolidation server. Settings are True or False.

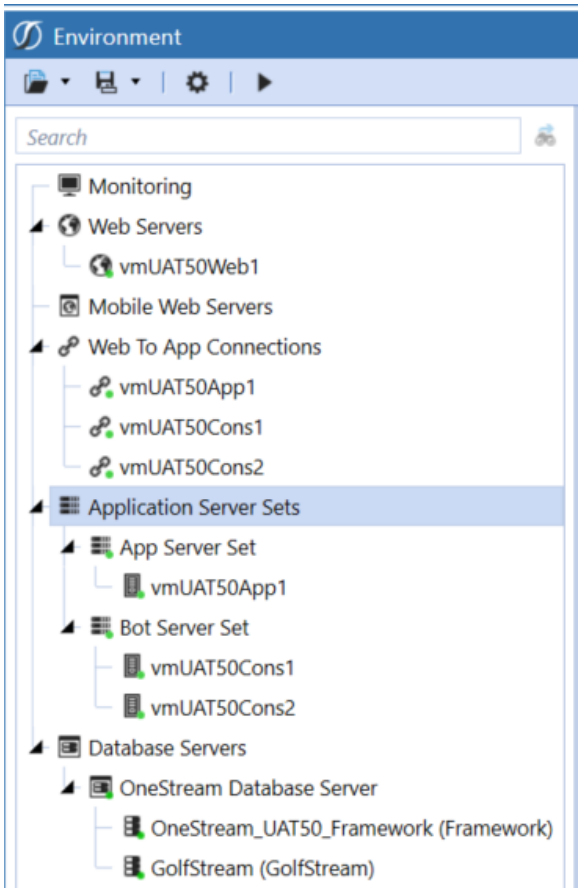
Used for Data Management

This examines the web configuration file to see if this server is configured as a Data Management server. Settings are True or False.

Working With Application Server Sets

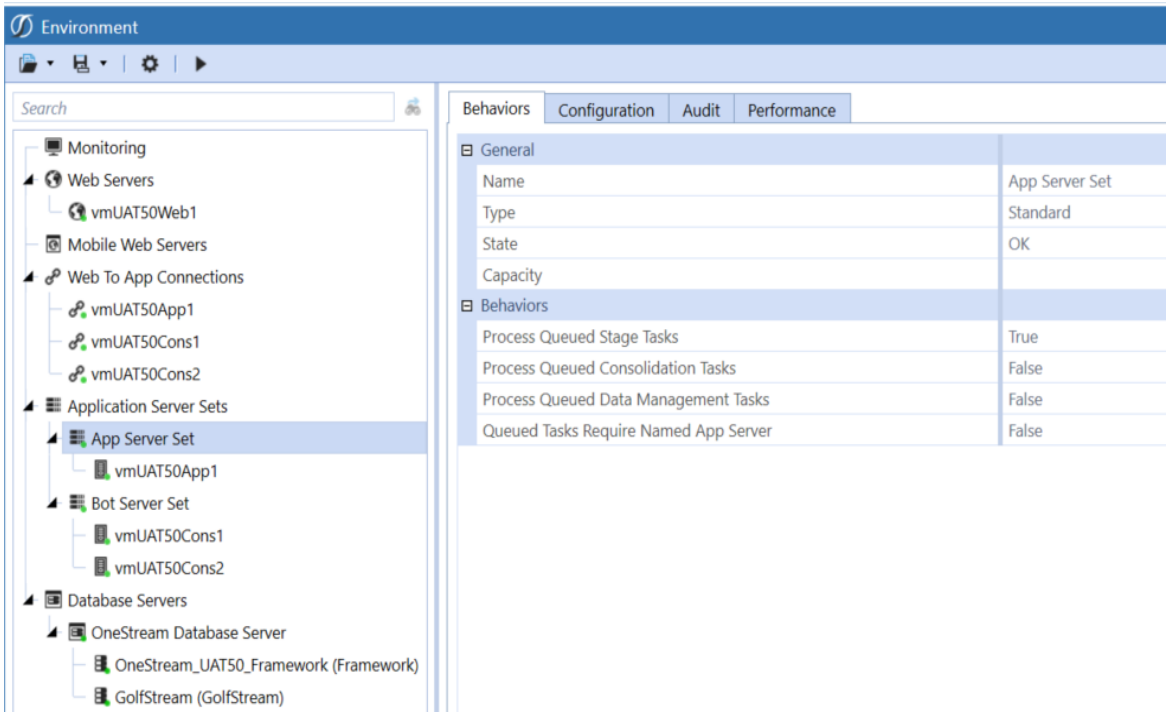
Click **Application Server Sets** to view the server sets participating in an environment. Sets display based on the Server Set configuration in the Application Server Configuration Utility. A red “X” indicates offline servers. You can access configurations for each sever set and server in a server set. If a Server is hosted on Azure, you can Scale Out/ and Scale In the Scale Set manually or using a System Business Rule.

Sample Application Server Set:



Accessing Server Sets Behaviors

Use the **Behavior** tab to view the capability of a Server Set. You can Scale Out (remove) and Scale In (add and configure) if the Scaling Type setting is set to Manual or ManualAndBusinessRule in the Server Configuration Utility file. This enables you to expand or contract server resources manually or automatically. For scaling with Azure, select Server Sets > Azure > Scaling Type. This feature will be available in a future release.

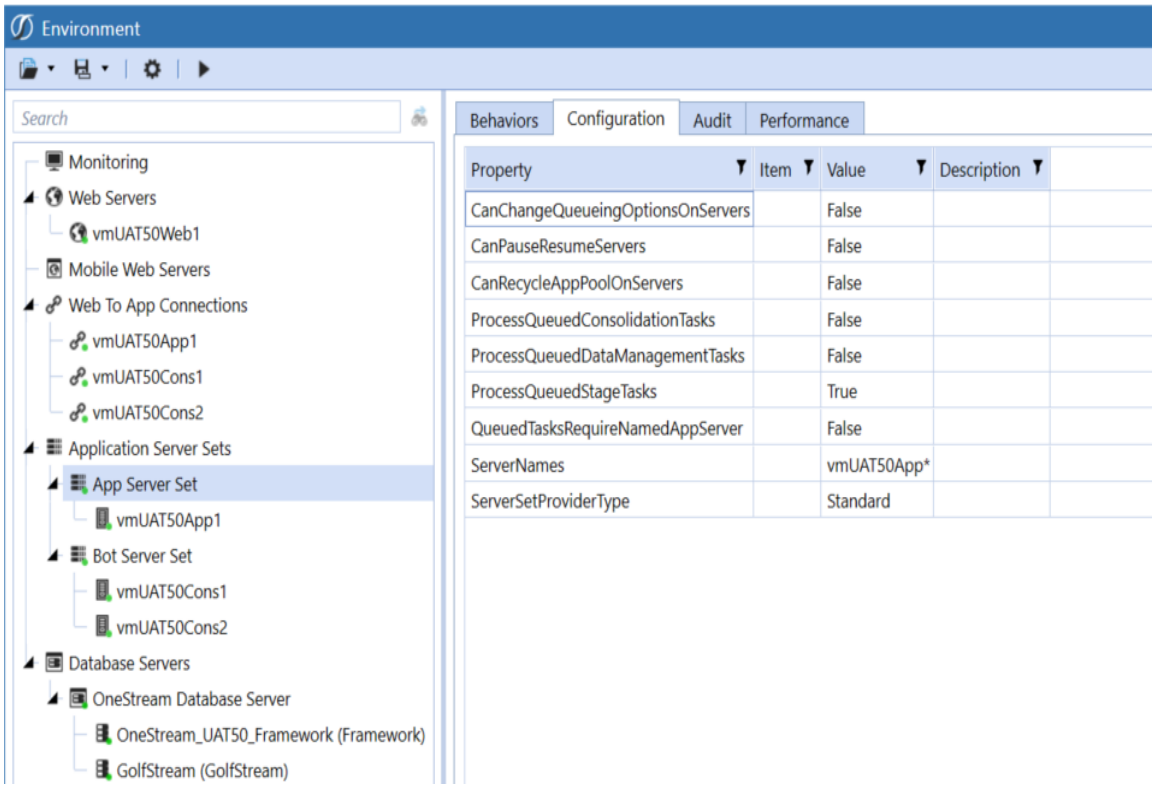


You can specify the following settings at the Server Set level using the Server Configuration Utility, but can override them at the individual Application Server level.

- **Process Queued Consolidation Tasks:** If true, the server can processes consolidation tasks.
- **Process Queued Data Management Tasks:** If true, the server can process data management tasks.
- **Process Queued Stage Tasks:** If true, the server can process stage tasks.
- **Queued Tasks Require Named Application Server:** If true, the server only runs assigned tasks.

Server Sets Configuration

Use the **Configuration** tab to view Server Set configurations defined with the Application Server Configuration Utility. For example:

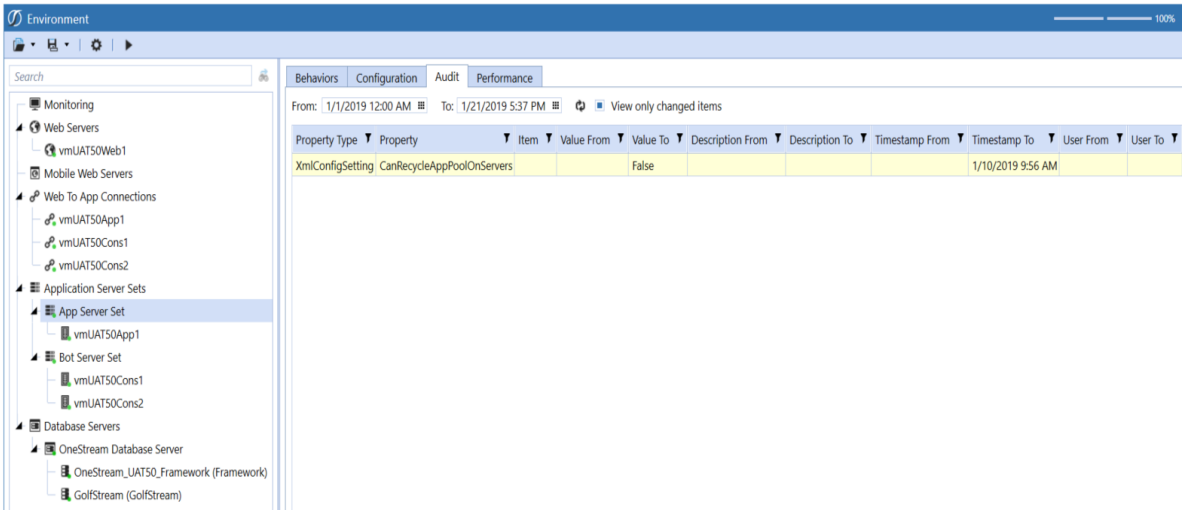


- **Azure Resource Group Name:** (Azure Scale Set Only) The resource group name for the set.
- **Azure Scale Set Name:** (Azure Scale Set Only) The scale set name in the resource group.
- **Can Change Queuing Options On Servers:** If true, Administrators can change the queuing behavior - impacting stage, consolidation and data management tasks - of specific servers.

- **Can Pause or Resume Servers:** If set to true, then the user can pause and/or resume the server from the Environment page via the Pause and Resume buttons.
- **Can Stop or Start Servers** (Azure Scale Set Only): If true, users can stop / restart the server on the Environment page.
- **Maximum | MinimumCapacity** (Azure Scale Set Only): A fail-safe setting to identify the maximum | minimum number of servers.
- **Process Queued Consolidation Tasks:** If true, the server can process consolidation tasks.
- **Process Queued <feature> Tasks:** If true, the server can process tasks such as data management and state tasks.
- **Queued Tasks Require Named Application Server:** If true, the server only runs its assigned tasks.
- **Server Names for Standard Server Sets (Supports *? Wildcards):** Specify server names if the standard server set type is used.
- **Server Set Provider Type:** Specify the external authentication provider used, such as Azure.
- **System Business Rule Name** (Azure Scale Set Only): The name of the extender business rule for Azure scale set and database scaling.

Using Server Sets Audit

The **Audit** tab identifies all property changes made in **XFAppServerConfig.xml**. Changes display in yellow. For example:

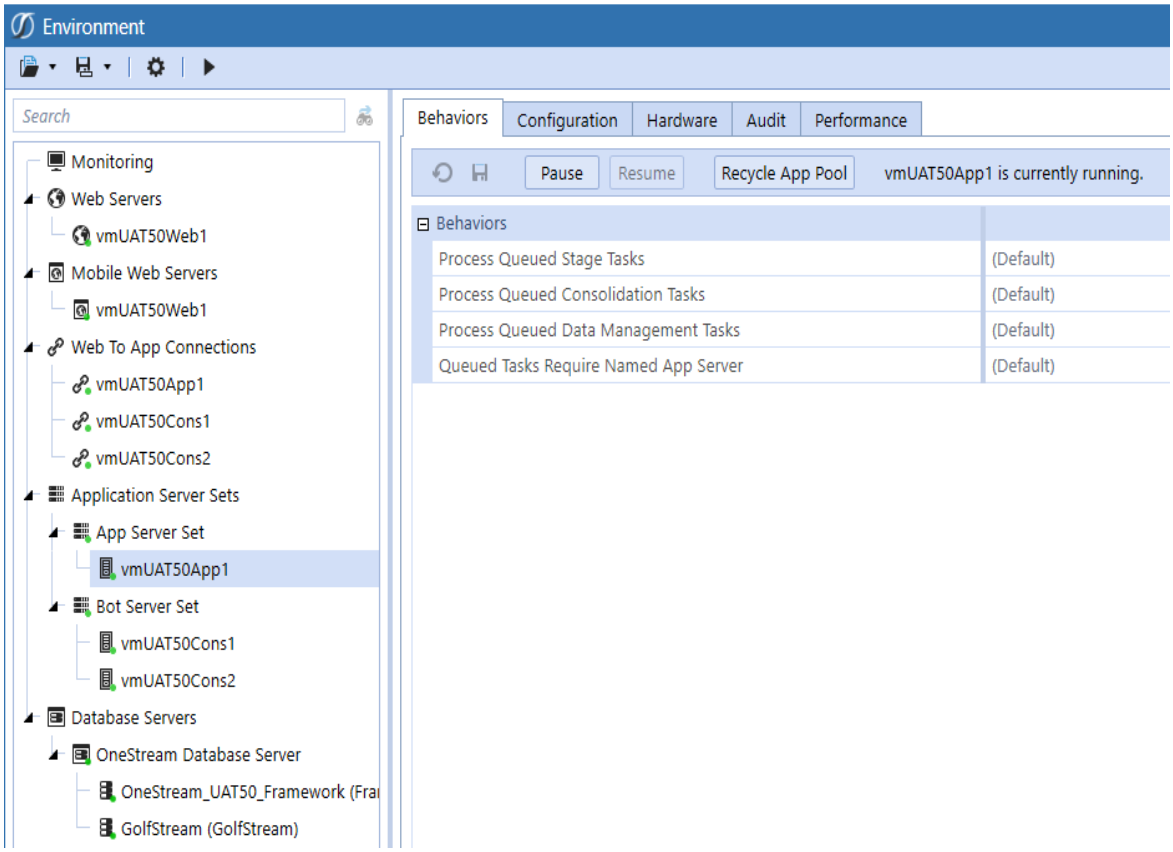


Monitoring Server Sets Performance

Use the **Performance** tab to view scale set and environment metrics, highlighting sections to drill down to time-specific records.

Application Server Behavior

When selected, the user can pause and resume that server, so that none of the long duration tasks can be run on that particular server. Also, IIS can be recycled by selecting the Recycle App Pool button. Whether these buttons appear or not is based on settings in the OneStream Server Configuration Utility.



Pause

Pause will stop the server from seeking more tasks to run from the queue but will let tasks that have already started, finish.

Resume

Server will resume accepting tasks from the queue.

Recycle App Pool

Used to recycle the specific server's application pool (reset IIS).

Stop - (Azure Scale Set Only)

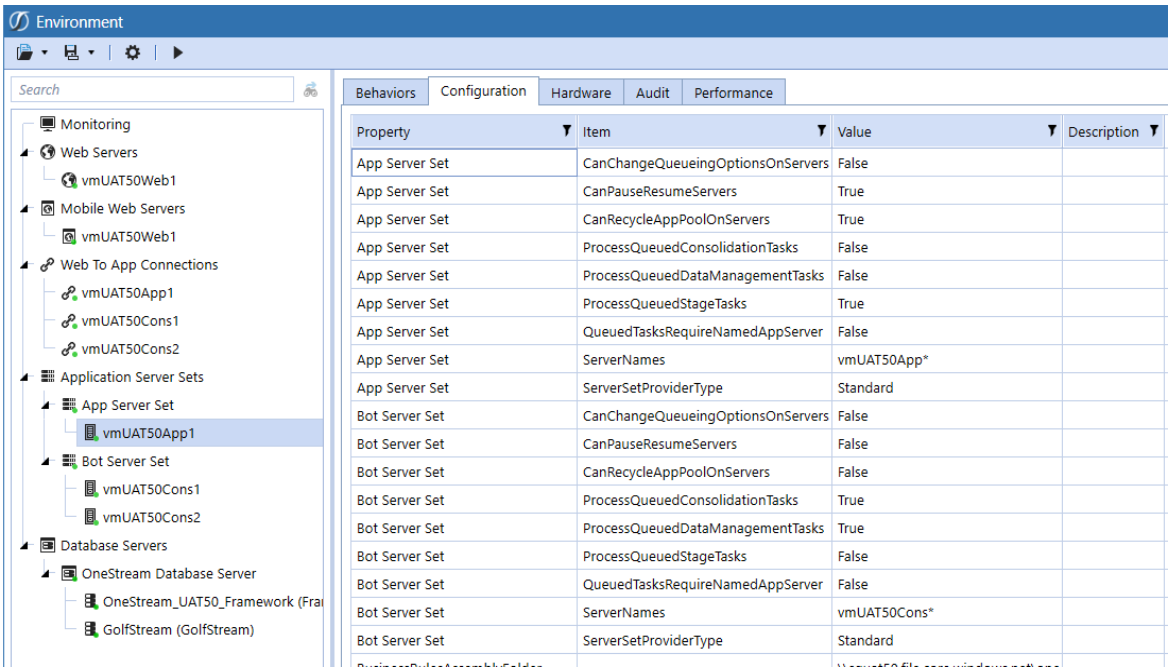
Stops the server, but while in this state will continue to incur Azure compute charges. The public and internal will be preserved.

Stop (Deallocate)- (Azure Scale Set Only)

Stopping this way will mean the cost of virtual machine will not be charged, but the public and internal IP will be deleted.

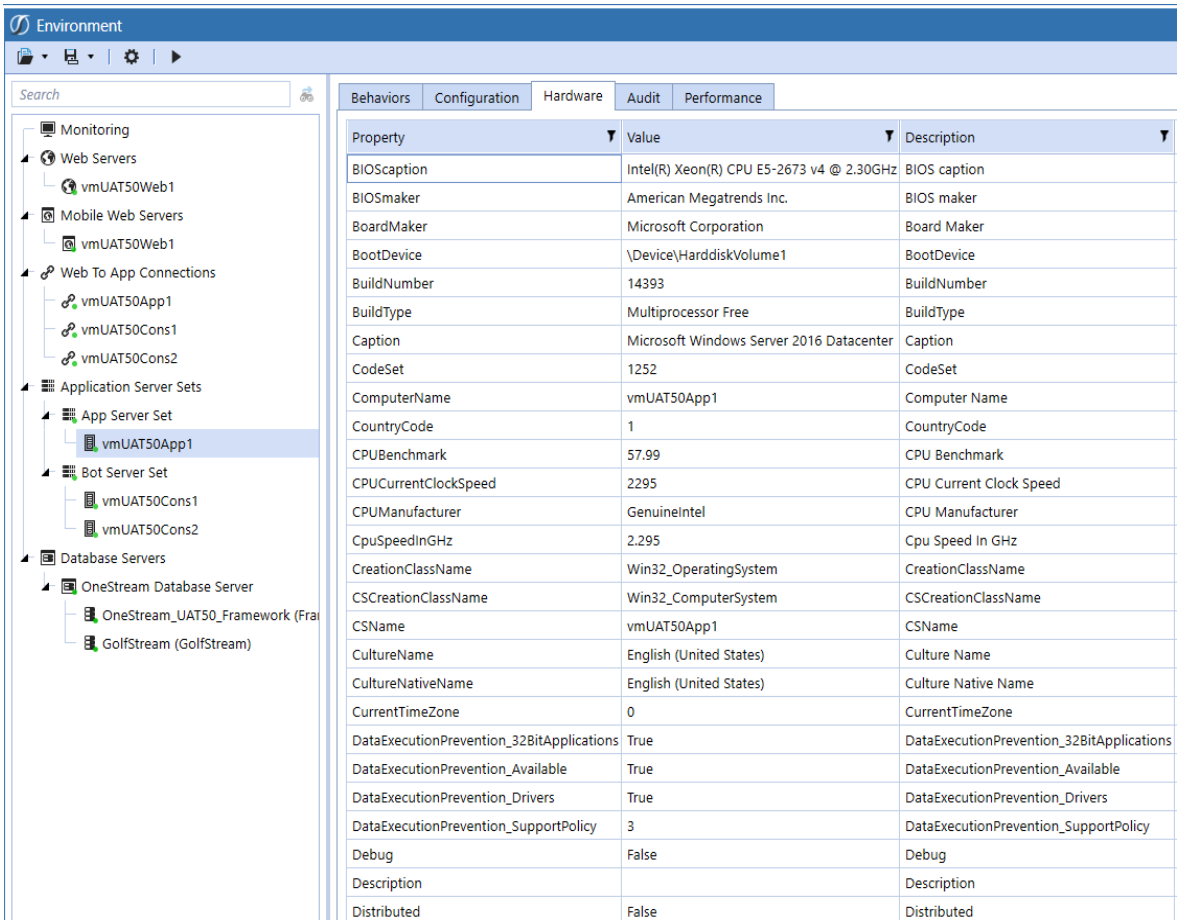
Application Server Configurations

This tab will display the server configurations from the OneStream Server Configuration Utility pertaining to that server.



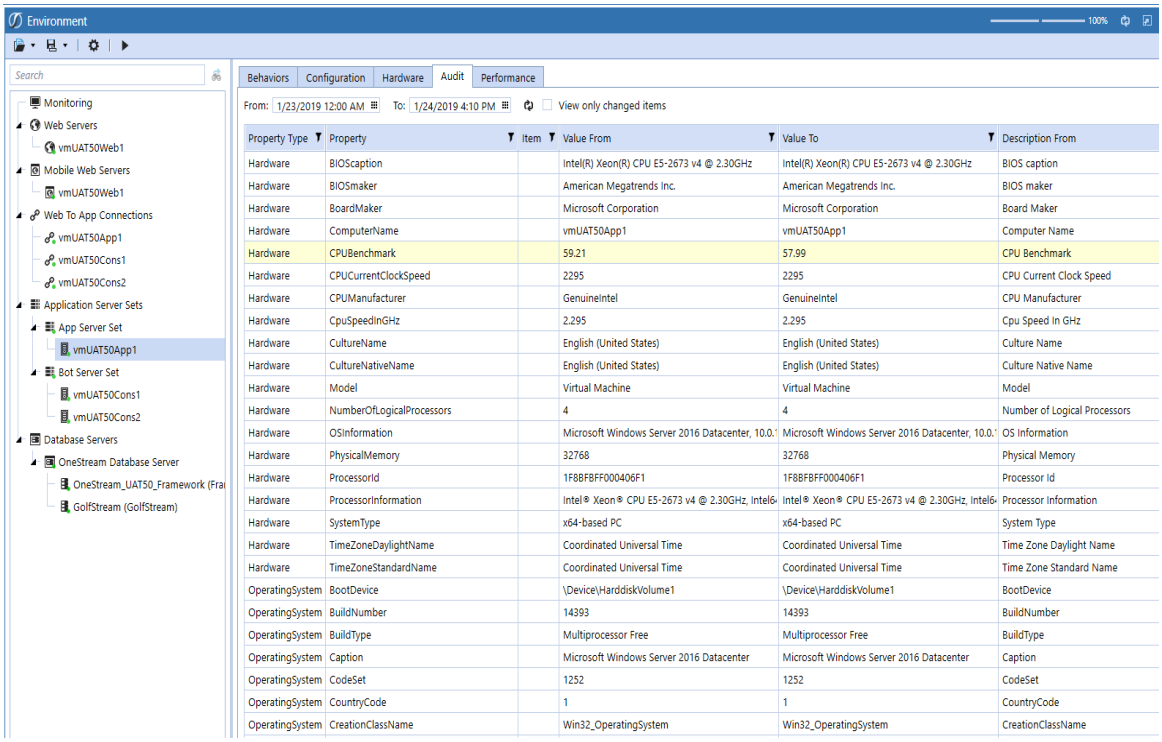
Application Server Hardware

This tab will show the machine hardware information based on settings in the OneStream Server Configuration Utility.



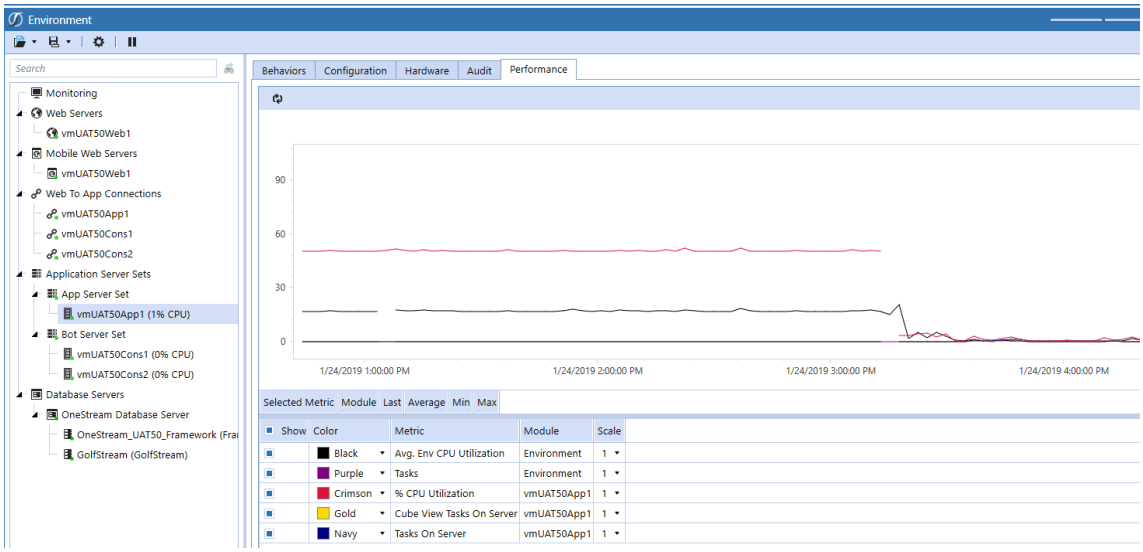
Application Server Audit

The audit tab will keep track and display any hardware and configuration changes.



Application Server Performance

This tab will display metric values pertaining to that server and the Environment.



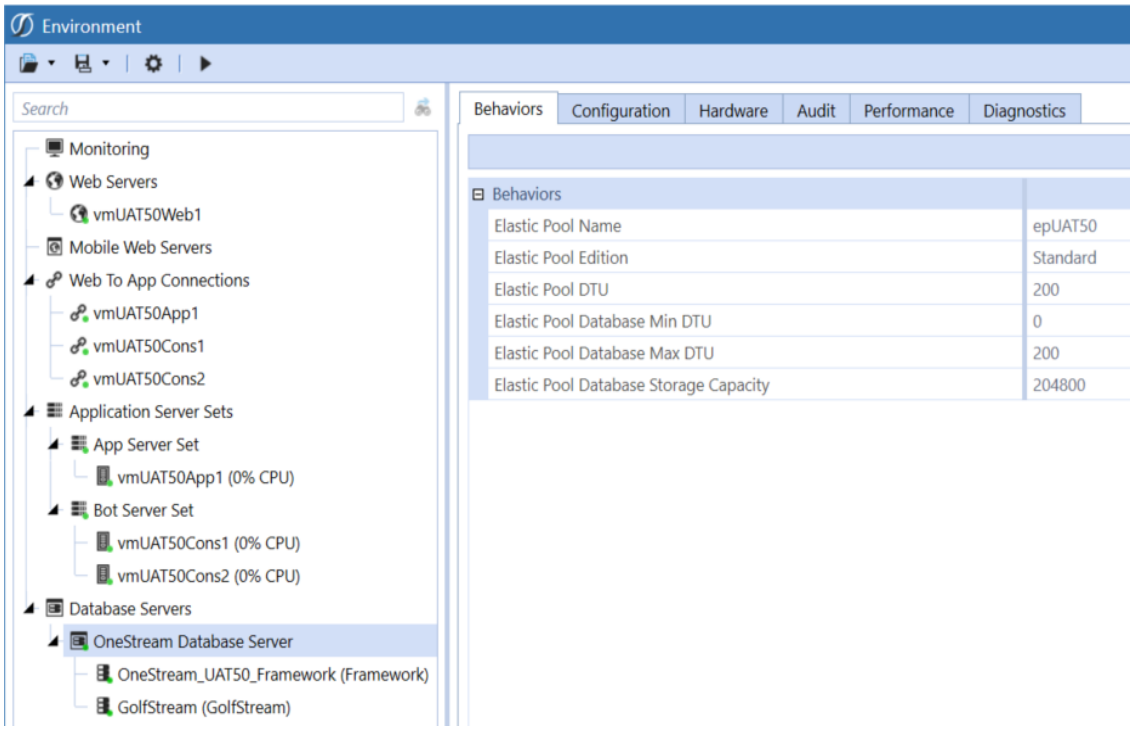
Database Servers (Connection Items)

The Database Servers section will list all the database connections based on settings in the OneStream Server Configuration Utility and all the databases that each connection is pointing to.

From the database connection items, the user can expand the SQL server elastic pool if one is configured, view hardware and server properties, view audit info, look at SQL server metrics, and run some diagnostic commands to track down performance issues.

Behaviors

Sample database connection Behavior tab:

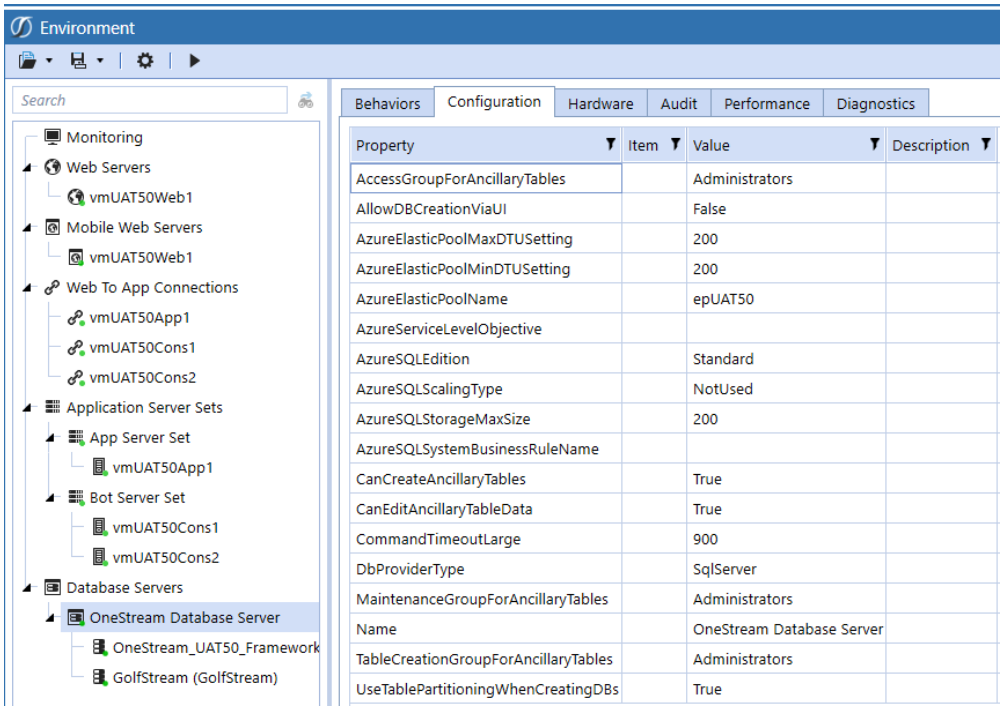


The behavior will show up if SQL Server Azure and Elastic Pool are configured and used. Using the Behavior tab, you can increase or decrease the resources available to an Elastic Pool based on resource needs. When rescaling Elastic Pool DTUs, database connections are briefly dropped. This is the same behavior as occurs when rescaling Elastic Pool DTUs for a single database (not in a pool).

Configuration

The configuration tab will display SQL server configuration properties.

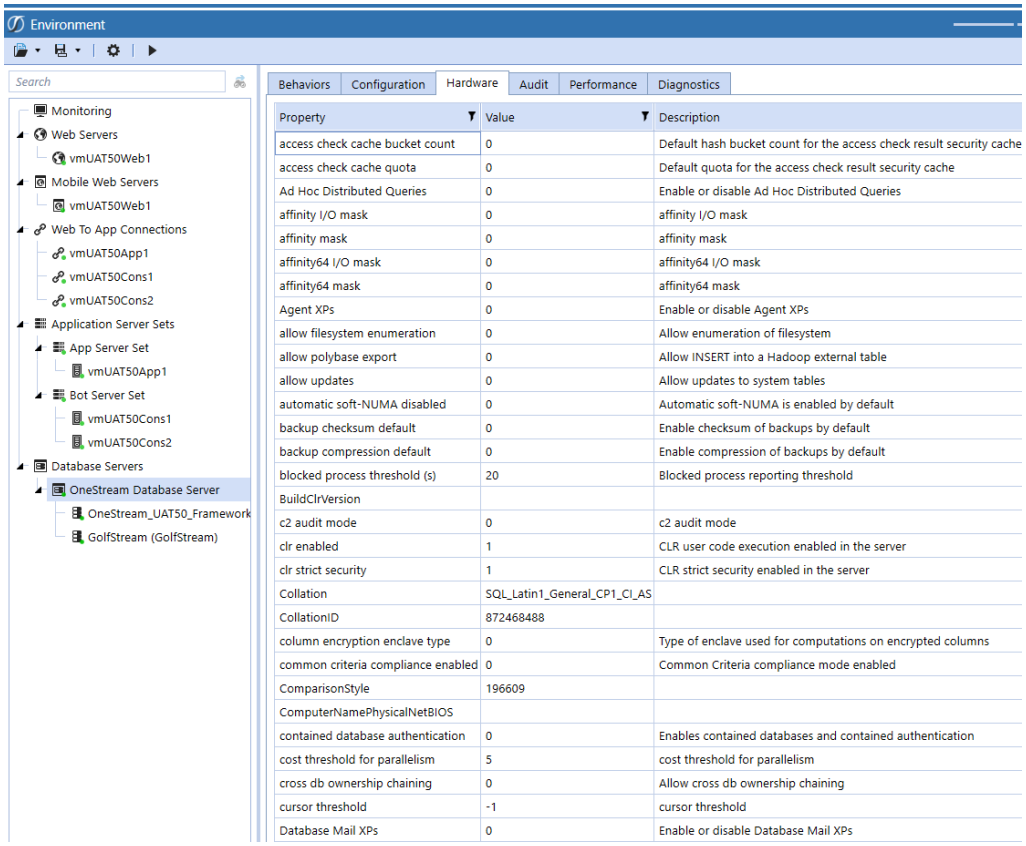
Sample configuration properties:



Hardware

The hardware tab will display hardware related information pertaining to SQL server.

Sample hardware properties:

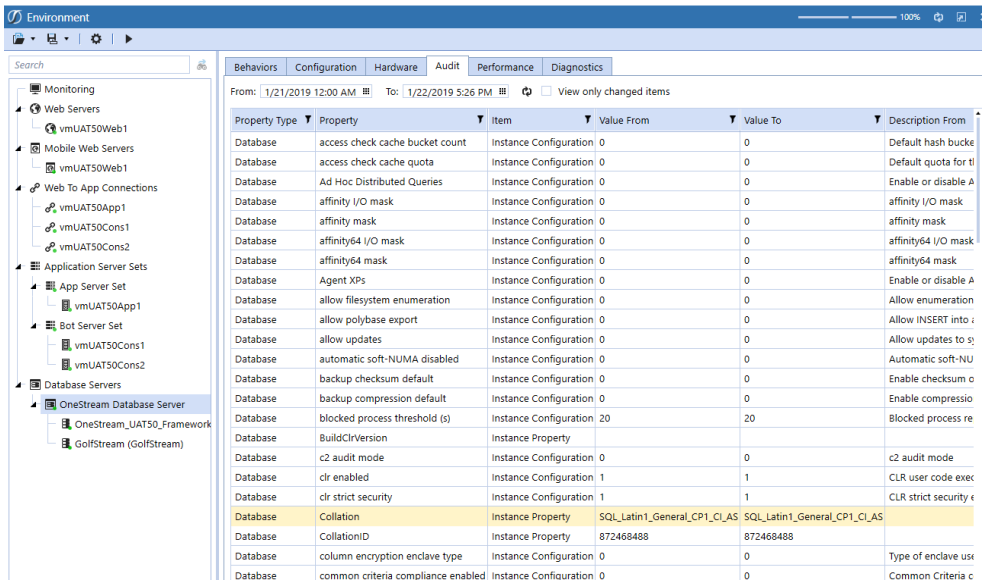


Property	Value	Description
access check cache bucket count	0	Default hash bucket count for the access check result security cache
access check cache quota	0	Default quota for the access check result security cache
Ad Hoc Distributed Queries	0	Enable or disable Ad Hoc Distributed Queries
affinity I/O mask	0	affinity I/O mask
affinity mask	0	affinity mask
affinity64 I/O mask	0	affinity64 I/O mask
affinity64 mask	0	affinity64 mask
Agent XPs	0	Enable or disable Agent XPs
allow filesystem enumeration	0	Allow enumeration of filesystem
allow polybase export	0	Allow INSERT into a Hadoop external table
allow updates	0	Allow updates to system tables
automatic soft-NUMA disabled	0	Automatic soft-NUMA is enabled by default
backup checksum default	0	Enable checksum of backups by default
backup compression default	0	Enable compression of backups by default
blocked process threshold (s)	20	Blocked process reporting threshold
BuildClrVersion		
c2 audit mode	0	c2 audit mode
clr enabled	1	CLR user code execution enabled in the server
clr strict security	1	CLR strict security enabled in the server
Collation	SQL_Latin1_General_CP1_CI_AS	
CollationID	872468488	
column encryption enclave type	0	Type of enclave used for computations on encrypted columns
common criteria compliance enabled	0	Common Criteria compliance mode enabled
ComparisonStyle	196609	
ComputerNamePhysicalNetBIOS		
contained database authentication	0	Enables contained databases and contained authentication
cost threshold for parallelism	5	cost threshold for parallelism
cross db ownership chaining	0	Allow cross db ownership chaining
cursor threshold	-1	cursor threshold
Database Mail XPs	0	Enable or disable Database Mail XPs

Audit

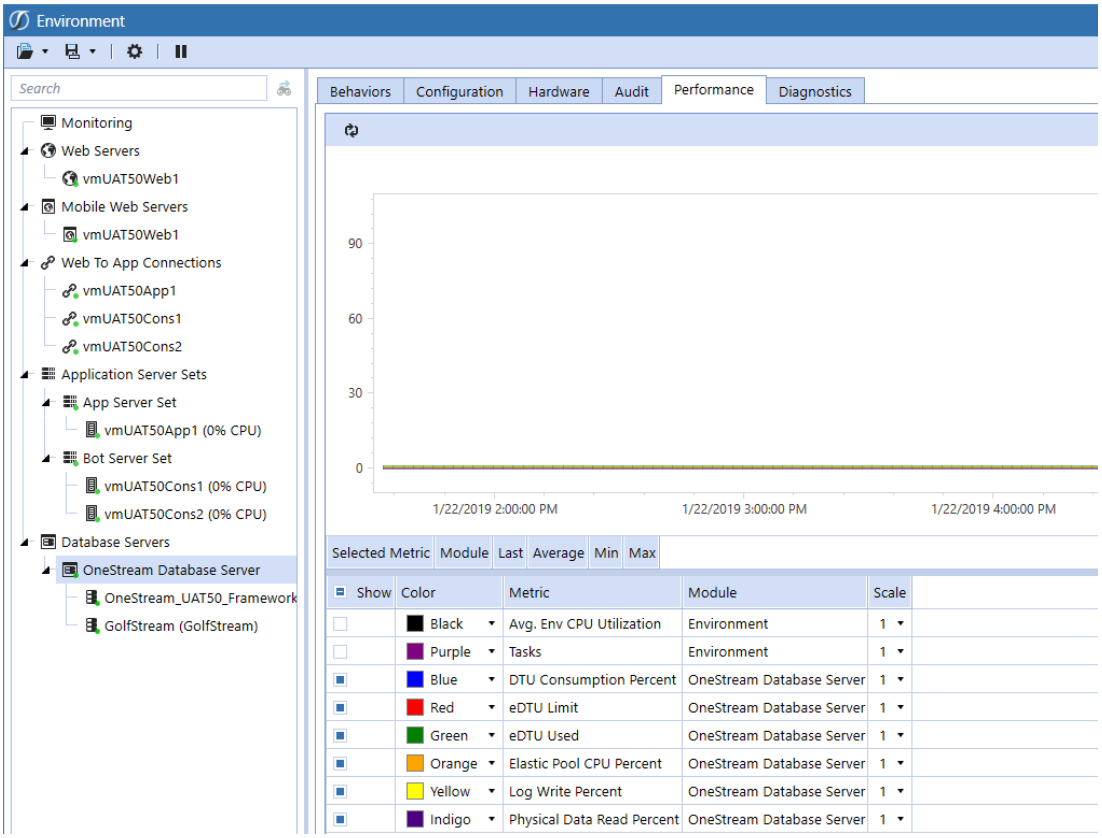
The audit tab will display any changes to the SQL Server properties.

Sample configuration audit report:



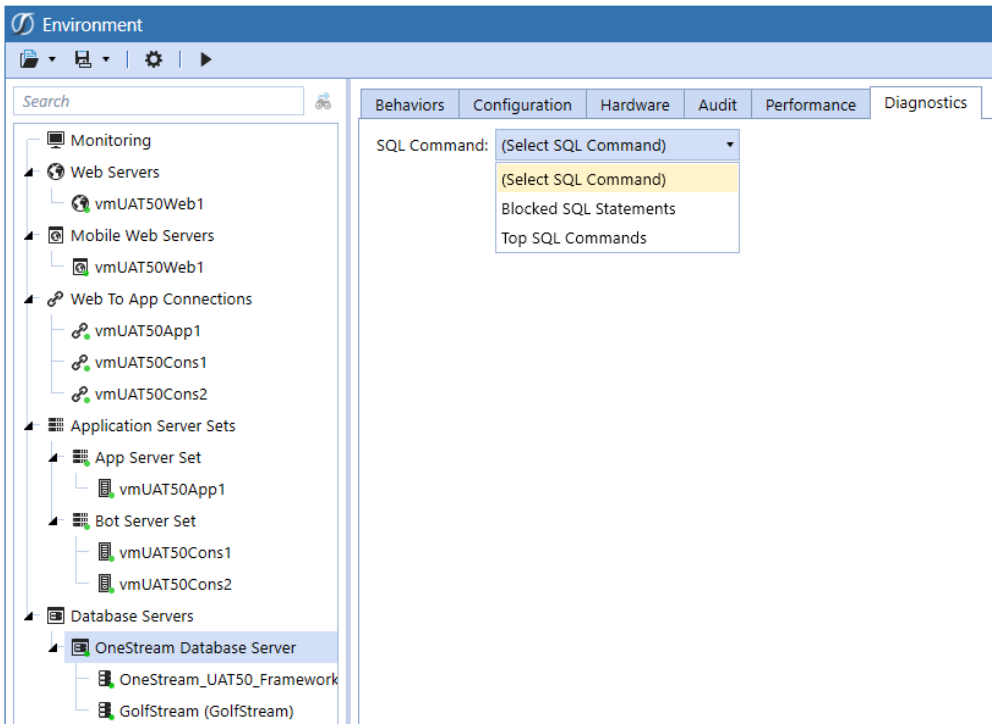
Performance

The Performance tab will display metrics pertaining to SQL Server:

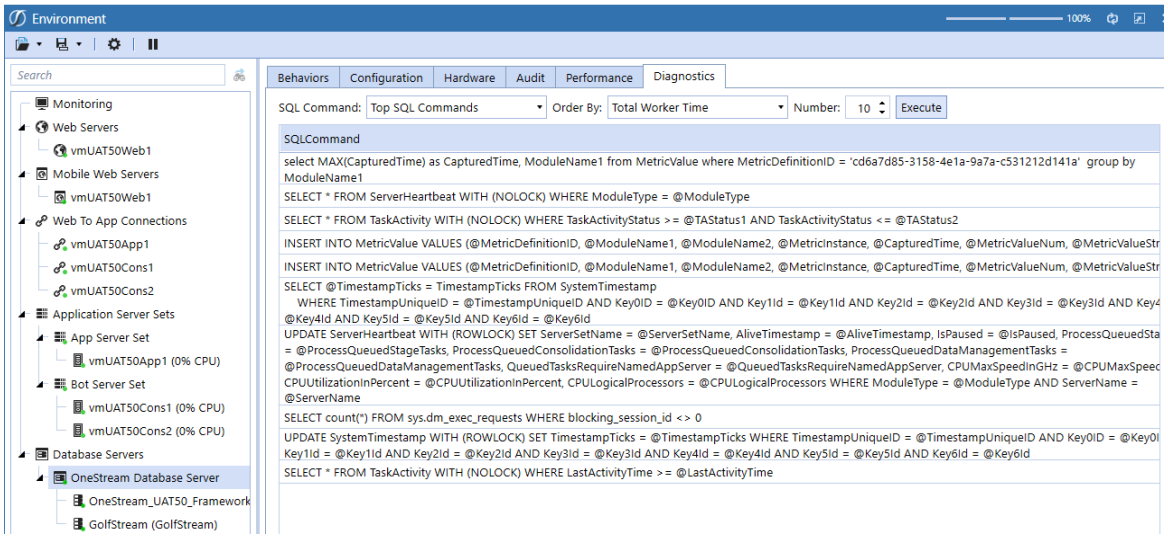


Diagnostics

The Diagnostics tab allows the user to run SQL diagnostic commands to determine performance issue on the database instance.



- SQL Deadlock information.
- The Deadlock SQL command will list out any deadlocks if any on the SQL server instance.
- Top SQL Commands will list out the top number of SQL using the order select by the user (Total Logical Reads, Total Logical Writes, Total Worker Time)

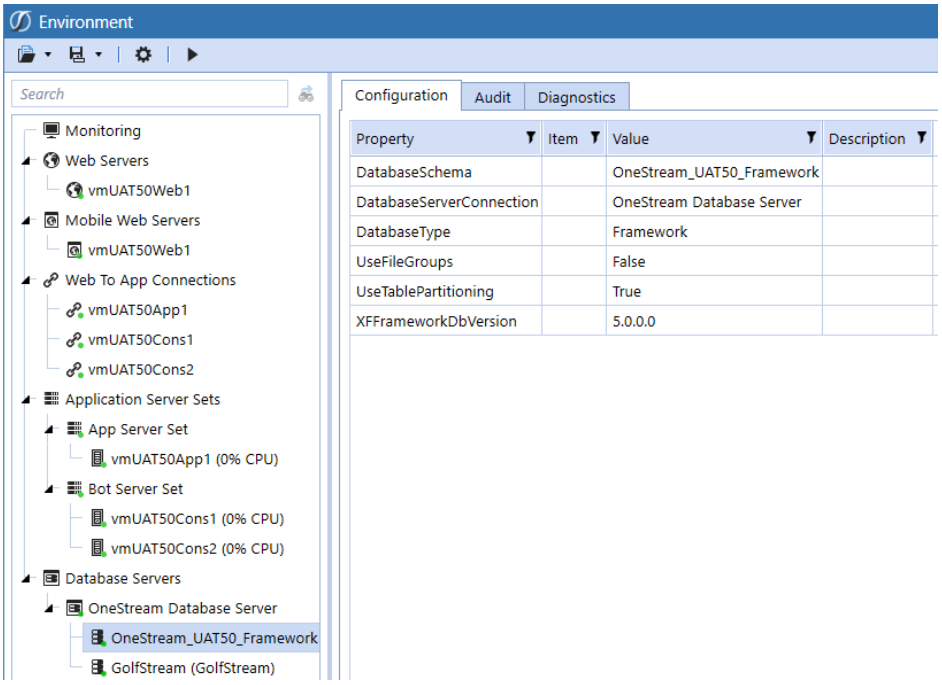


OneStream Database Servers (Schema Items)

This section will list all the database schemas that this connection is pointing to. Each schema has its own configurations, audit and diagnostic tabs.

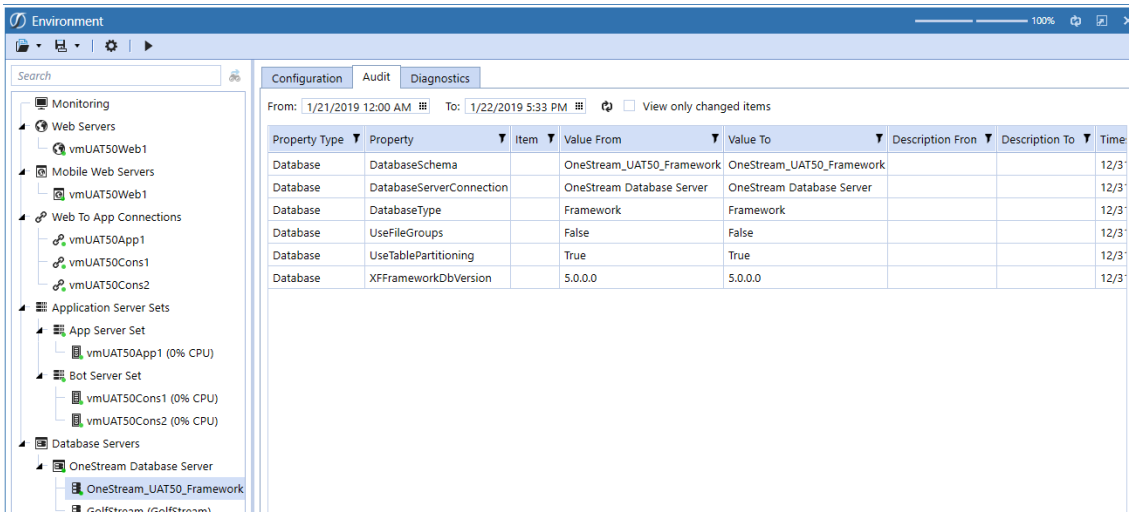
Configuration

The Configuration tab contains the application-specific information:



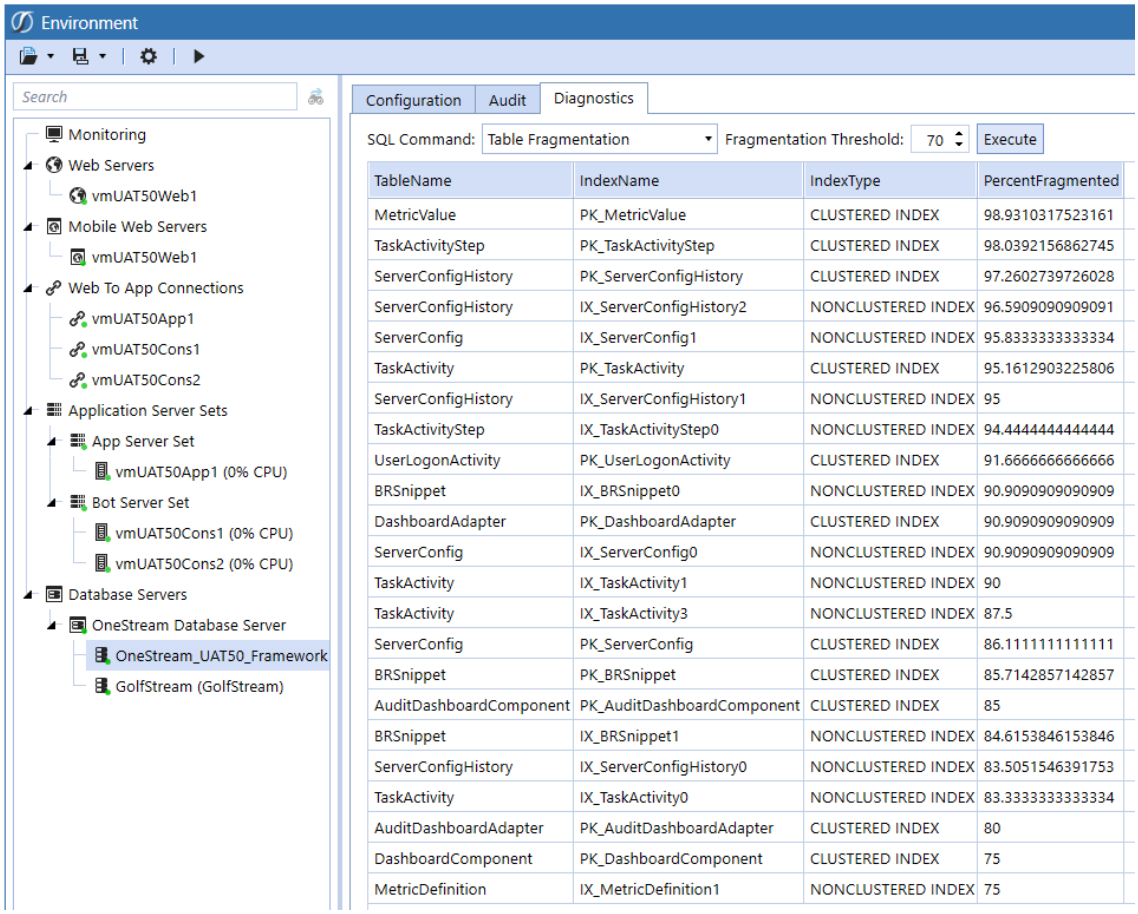
Audit

The audit tab will show if any of the application configuration has been changed.



Diagnostic

Diagnostics tab will show a report of current schema table fragmentation. Sample of table fragmentation on the framework DB showing all tables beyond 70% fragmented:



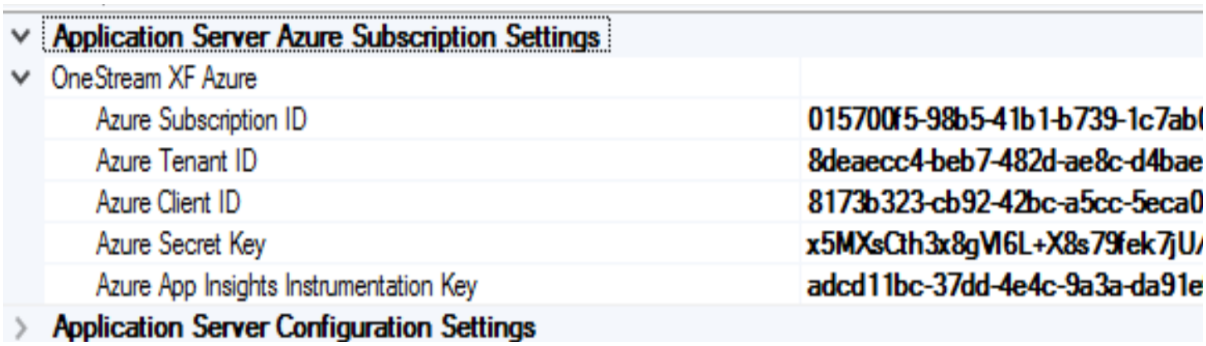
TableName	IndexName	IndexType	PercentFragmented
MetricValue	PK_MetricValue	CLUSTERED INDEX	98.9310317523161
TaskActivityStep	PK_TaskActivityStep	CLUSTERED INDEX	98.0392156862745
ServerConfigHistory	PK_ServerConfigHistory	CLUSTERED INDEX	97.2602739726028
ServerConfigHistory	IX_ServerConfigHistory2	NONCLUSTERED INDEX	96.5909090909091
ServerConfig	IX_ServerConfig1	NONCLUSTERED INDEX	95.8333333333334
TaskActivity	PK_TaskActivity	CLUSTERED INDEX	95.1612903225806
ServerConfigHistory	IX_ServerConfigHistory1	NONCLUSTERED INDEX	95
TaskActivityStep	IX_TaskActivityStep0	NONCLUSTERED INDEX	94.4444444444444
UserLogonActivity	PK_UserLogonActivity	CLUSTERED INDEX	91.6666666666666
BRSnippet	IX_BRSnippet0	NONCLUSTERED INDEX	90.9090909090909
DashboardAdapter	PK_DashboardAdapter	CLUSTERED INDEX	90.9090909090909
ServerConfig	IX_ServerConfig0	NONCLUSTERED INDEX	90.9090909090909
TaskActivity	IX_TaskActivity1	NONCLUSTERED INDEX	90
TaskActivity	IX_TaskActivity3	NONCLUSTERED INDEX	87.5
ServerConfig	PK_ServerConfig	CLUSTERED INDEX	86.1111111111111
BRSnippet	PK_BRSnippet	CLUSTERED INDEX	85.7142857142857
AuditDashboardComponent	PK_AuditDashboardComponent	CLUSTERED INDEX	85
BRSnippet	IX_BRSnippet1	NONCLUSTERED INDEX	84.6153846153846
ServerConfigHistory	IX_ServerConfigHistory0	NONCLUSTERED INDEX	83.5051546391753
TaskActivity	IX_TaskActivity0	NONCLUSTERED INDEX	83.3333333333334
AuditDashboardAdapter	PK_AuditDashboardAdapter	CLUSTERED INDEX	80
DashboardComponent	PK_DashboardComponent	CLUSTERED INDEX	75
MetricDefinition	IX_MetricDefinition1	NONCLUSTERED INDEX	75

Azure Configurations

(Azure Only or if Azure Elastic Pool Being Used)

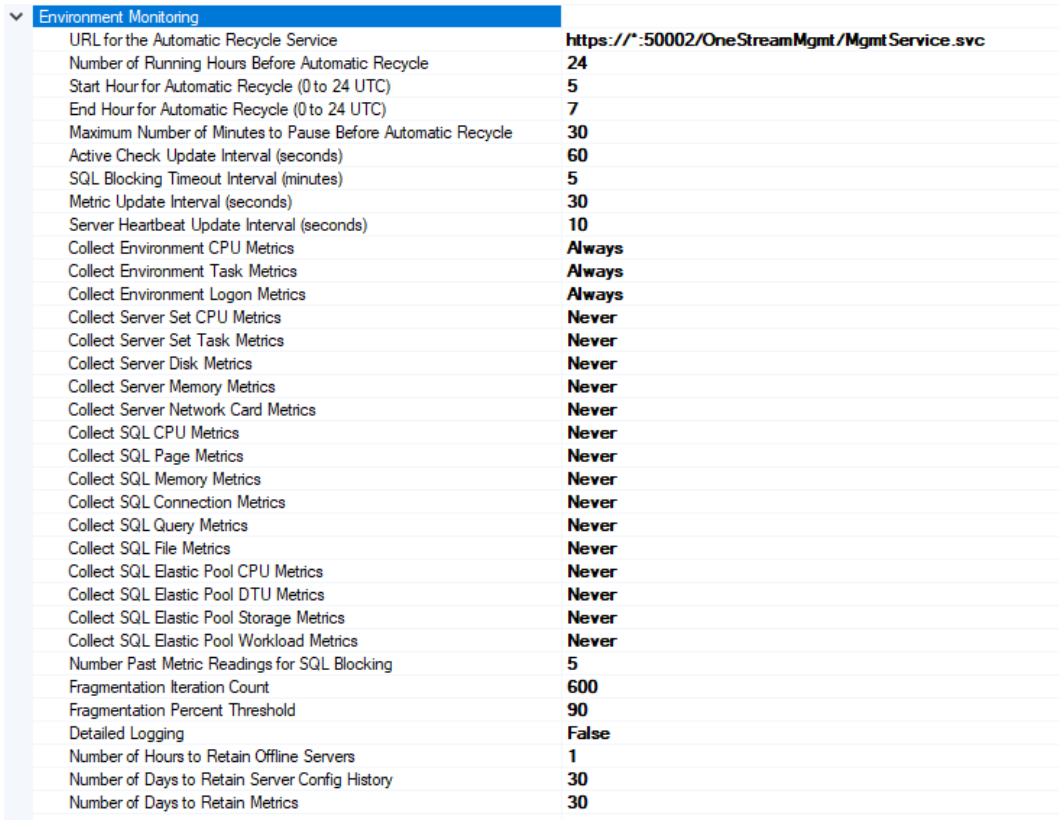
Azure Subscription Settings

The Azure Subscription Settings must be filled in, as they are used to login and retrieve Azure settings and data. This section will should be populated when using Azure Elastic Pool or if using Scale Sets (which is a feature that will available in a future release).



Environment Monitoring

The Environment Monitoring section configures how and how often metrics are collected. What is available for metrics or monitoring is based on configurations made in the OneStream Server Configuration Utility. See Installation and Configuration Guide.



Environment Monitoring	
URL for the Automatic Recycle Service	https://*:50002/OneStreamMgmt/MgmtService.svc
Number of Running Hours Before Automatic Recycle	24
Start Hour for Automatic Recycle (0 to 24 UTC)	5
End Hour for Automatic Recycle (0 to 24 UTC)	7
Maximum Number of Minutes to Pause Before Automatic Recycle	30
Active Check Update Interval (seconds)	60
SQL Blocking Timeout Interval (minutes)	5
Metric Update Interval (seconds)	30
Server Heartbeat Update Interval (seconds)	10
Collect Environment CPU Metrics	Always
Collect Environment Task Metrics	Always
Collect Environment Logon Metrics	Always
Collect Server Set CPU Metrics	Never
Collect Server Set Task Metrics	Never
Collect Server Disk Metrics	Never
Collect Server Memory Metrics	Never
Collect Server Network Card Metrics	Never
Collect SQL CPU Metrics	Never
Collect SQL Page Metrics	Never
Collect SQL Memory Metrics	Never
Collect SQL Connection Metrics	Never
Collect SQL Query Metrics	Never
Collect SQL File Metrics	Never
Collect SQL Elastic Pool CPU Metrics	Never
Collect SQL Elastic Pool DTU Metrics	Never
Collect SQL Elastic Pool Storage Metrics	Never
Collect SQL Elastic Pool Workload Metrics	Never
Number Past Metric Readings for SQL Blocking	5
Fragmentation Iteration Count	600
Fragmentation Percent Threshold	90
Detailed Logging	False
Number of Hours to Retain Offline Servers	1
Number of Days to Retain Server Config History	30
Number of Days to Retain Metrics	30

URL for the Automatic Recycle Service

Used to specify the address of the recycle management service. The protocol for the address should be set to however the service is deployed (https or http) and the port (default is 50002). The asterisk will force the service to use the fully qualified domain name of the executing server.

Number of Running Hours Before Automatic Recycle

Used to specify how often the app servers should be restarted. The default is 24.0. Use 0.0 to turn off auto recycling. Fractional numbers may be used in this setting.

Start Hour for Automatic Recycle (0 to 24 UTC)

Used to specify the time period within each day that the server can be recycled. This setting is only used if Number of Running Hours Before Automatic Recycle is set to 24.0. The default setting is 5. With that value, the application servers will be recycled between 5:00am UTC and what the setting for the End Hour for Automatic Recycle is set to.

End Hour for Automatic Recycle (0 to 24 UTC)

Used to specify the time period within each day that the server can be recycled. This setting is only used if Number of Running Hours Before Automatic Recycle is set to 24.0. The default setting is 7. With that value, the application servers will be recycled between 7:00am UTC and what the setting for the Start Hour for Automatic Recycle is set to.

Maximum Number of Minutes to Pause Before Automatic Recycle

Used to specify the maximum number of minutes that the server should pause its ability to run newly queued tasks before recycling. The default setting is 30. This setting is the time period that the server will use to allow previously started tasks to finish. If there are no previously running tasks on the application server, the setting is ignored.

Active Check Update Interval (seconds)

How often system checks for system monitoring (i.e.; deadlocks ...)

Metric Update Interval (seconds)

How often system checks for metric updates.

Server Heartbeat Update Interval (seconds)

How often system updates that this server is alive.

Collect Environment CPU Metrics

How often to collect environment CPU metrics.

Collect Environment Task Metrics

How often to collect environment task metrics (i.e.; running tasks, Queued Tasks …)

Collect Environment Login Metrics

How often to collect environment user login metrics.

Collect Server Set CPU Metrics

How often to collect server set CPU metrics.

Collect Server Set Task Metrics

How often to collect server set task metrics (i.e.; running tasks, Queued Tasks …).

Collect Server Disk Metrics

How often to collect server disk metrics (i.e.; Average Disk read/write per sec…).

Collect Server Memory Metrics

How often to collect server memory metrics (i.e.; Available MBytes…).

Collect Server Network Card Metrics

How often to collect server network card metrics.

Collect SQL CPU Metrics

How often to collect SQL server CPU metrics.

Collect SQL Page Metrics

How often to collect SQL server Page caching metrics (i.e.; Page Life Expectancy…).

Collect SQL Memory Metrics

How often to collect SQL server CPU metrics.

Collect SQL Connection Metrics

How often to collect SQL server connection metrics (i.e.; Number of connections…).

Collect SQL Query Metrics

How often to collect SQL server Query metrics (i.e.; Number of Deletes/Inserts…).

Collect SQL File Metrics

How often to collect SQL server File growth metrics.

Collect SQL Elastic Pool CPU Metrics

How often to collect SQL server Elastic Pool CPU metrics (i.e.; Number of connections…).

Collect SQL Elastic Pool DTU Metrics

How often to collect SQL server Elastic Pool DTU metrics (i.e.; Number of connections…).

Collect SQL Elastic Pool Storage Metrics

How often to collect SQL server Elastic Pool Storage metrics (i.e.; disk storage usage…).

Collect SQL Elastic Pool Workload Metrics

How often to collect SQL server Elastic Pool Workload metrics.

Number Past Metric Readings for SQL Blocking

Used to select the past number of metric values for blocking analysis.

Fragmentation Iteration Count

Used for fragmentation check, every 10 hours if the ActiveCheckUpdateIntervalInSec is set to 60.

Fragmentation Percent Threshold

Used for fragmentation threshold check in percent.

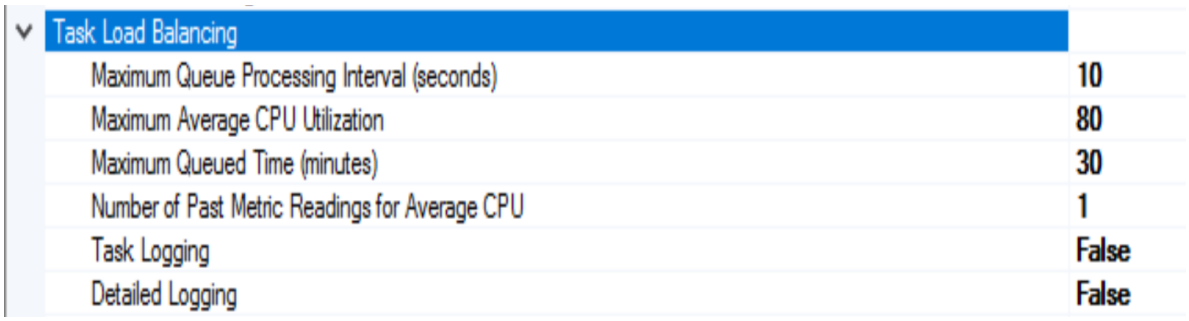
Detailed Logging

If true, then log whenever we enter and exit the metric collection and the Active System check.

Number Hours To Retain Offline Servers

Remove offline servers from the heartbeat table after certain number of hours.

Task Load Balancing



Maximum Queued Processing Interval (seconds)
How often the queuing thread checks for new tasks to be run.

Number Past Metric Reading For Analysis
How many metric readings to retrieve for analyzing server demand.

Maximum Queued Time (minutes)
Maximum time to wait for a task to run before it is cancelled.

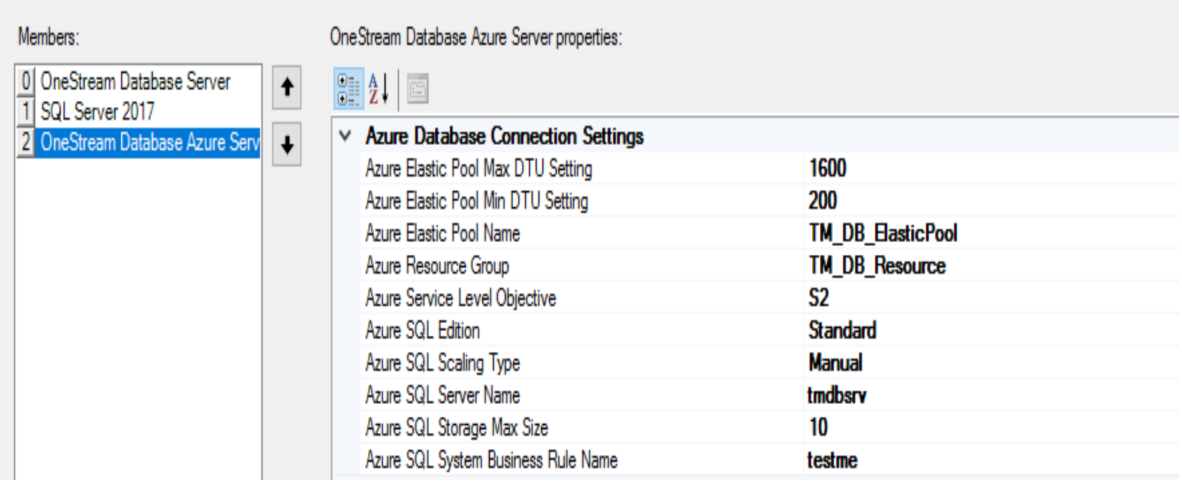
Maximum Average CPU Utilization
Maximum average CPU utilization before we determine that a server can't take a task.

Task Logging Only
If true will only logs picked up tasks.

Detailed Logging
If true, then log whenever we enter and exit the Task Load Balancing function and what was run.

Database Server Connection

The Azure section in the Database Connection Settings needs to be completed only if Azure database and/or Azure Elastic pool is used.



Sample Azure database connection settings:

Azure Elastic Pool Max DTU Setting

This is a fail-safe setting that the user can't set the DTU setting above this point.

Azure Elastic Pool Min DTU Setting

This is a fail-safe setting that the user can't set the DTU setting below this point.

Elastic Pool Name

The name of the elastic pool used with this database connection.

Azure Resource Group

The resource group name that the elastic pool is in.

Azure Service Level Objective

The service level used. This setting is used to create application on Azure.

Azure SQL Edition

The Azure SQL Server edition used.

Azure SQL Scaling Type

Manual, Business Rule or ManualAndBusiness Rule. The is used to set the type of scaling that is used to Scale Out or Scale In the SQL Server eDTUs. The default setting should be set to Not Used.

Azure SQL Server Name

The name of the SQL Server. This setting is used to create application on Azure.

Azure SQL Storage Max Size

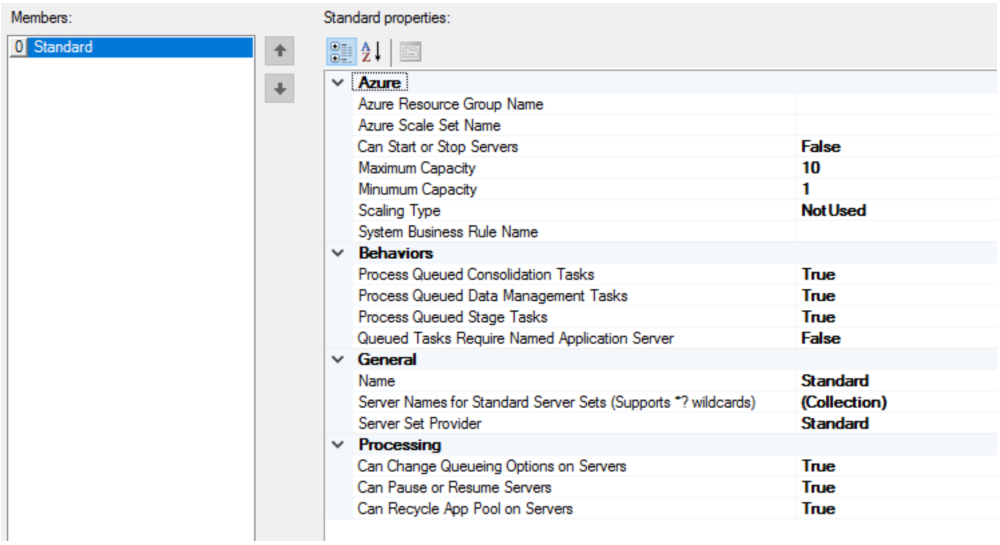
This is used to specify the database storage size when creating a database on AZURE.

Azure SQL System Business Rule Name

If SQL Scaling Type is set to BusinessRule, this setting must be set to a business rule that is used to scale in/out. The Environment metrics and the database metrics are passed to this rule to properly determine the eDTU scaling.

Azure Server Sets Settings

This setting is used to set up the used scale sets for the different application servers:



The **Azure** section is used to set the Azure resource group name and the scale set name as it exists on Azure. The Scaling options are used to set the scaling capacity and the Scaling Type (e.g. Manual or BusinessRule). The System Business Rule Name must be set if the Scaling Type is set to Business Rule.

The **Behaviors** section is used determine how this scale set is used. Whether it can be called by the web server or it is used as a processing server for consolidations and Stage Load.

The **General** section is used to determine whether this scale set is used and who is its provider (i.e.; Azure, External …).

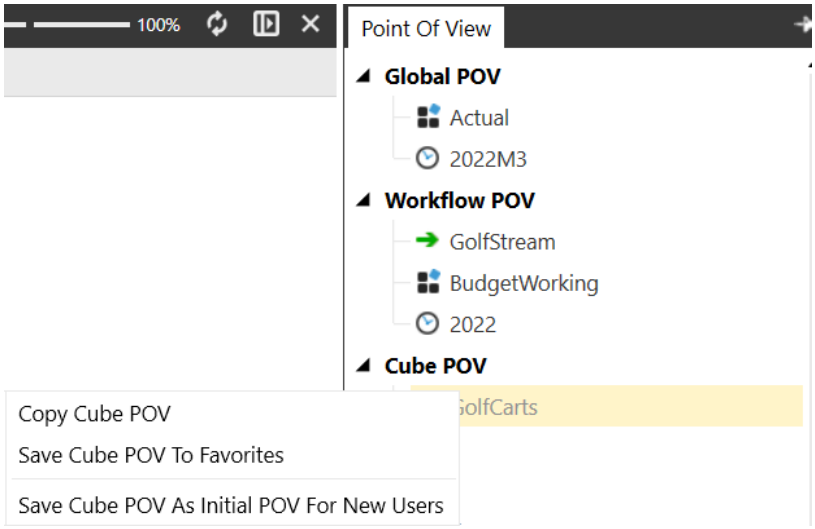
File Explorer

Documents and saved Point of View settings can be stored in the Application database.

Create a new folder in the File Explorer

Upload a specific file into any folder. If a user has Administrator rights or is in the ManageFileShare security role, he/she can upload and delete files in the Harvest folder.

A common Cube Point of View can be stored for continued reuse by multiple users. The POV can be stored for all users in the Public area or for the specific user in the User directory. To create a saved, named POV, right click the Cube POV node under Point of View within the Context Pane:

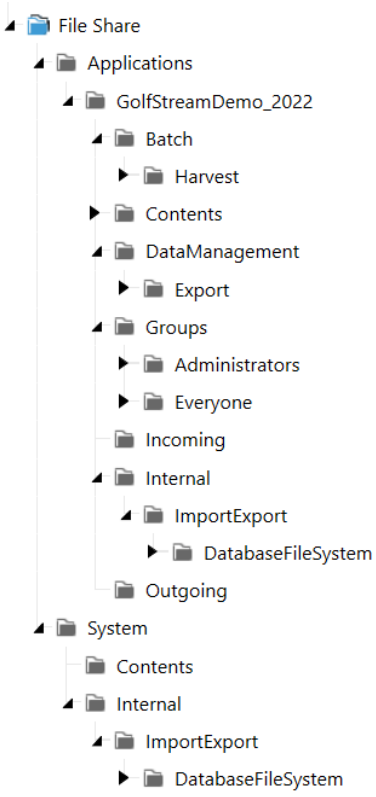


File Share

File Share is a self-service directory which supports file storage external from the OneStreamApplication and System Databases. Files stored in an Incoming folder are accessible only through that Workflow Profile.

To access File Share, click the **File Explorer** button in the toolbar, or navigate to **System >Tools** > **File Explorer**.

toolbar.



Folders

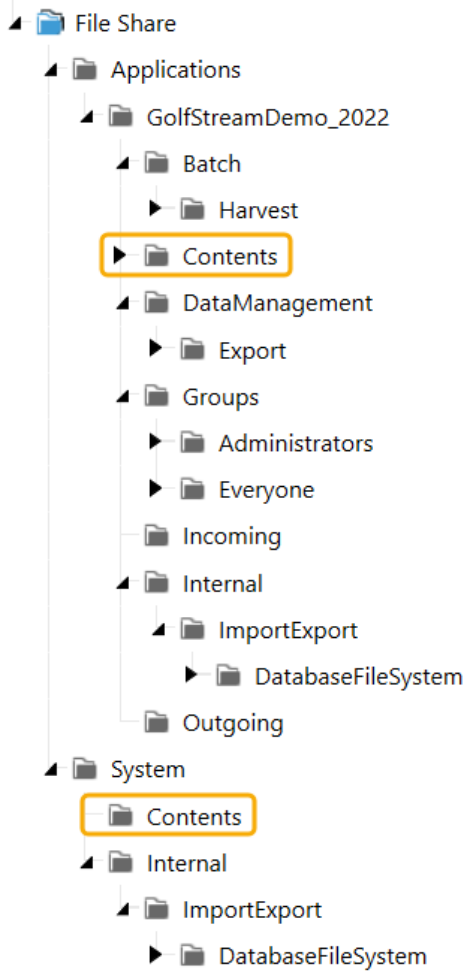
Contents Folders

Both the Application and System folders contain an auto-generated folder named Contents intended to store files larger than 300 MB. This folder is a secureable storage area that can be used in place of the File Explorer application database.

Permissions

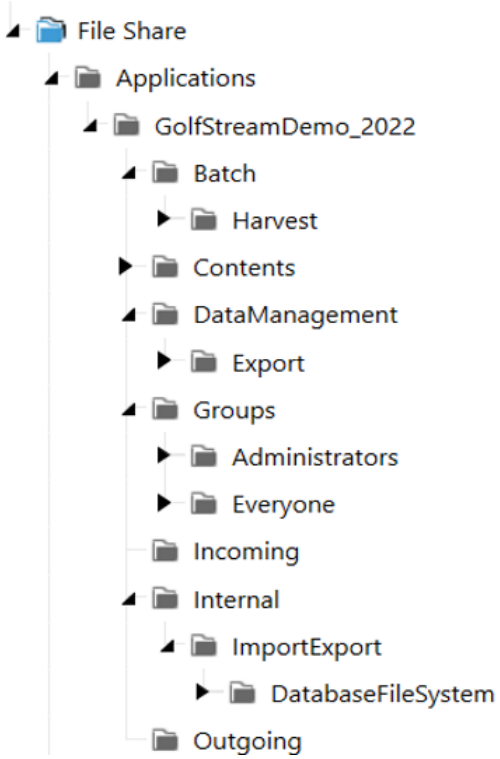
The Contents folders are managed and secured by System Security Roles. To set System Security Roles, navigate to **System** > **Security** > **System Security Roles**:

- Administrator and ManageFileShare Roles have full rights.
- Non-Administrators can be assigned rights to modify, fully access, or have limited access to the Content folders.



Application Folders

This folder has application specific storage.



Batch and Harvest

These folders are used for automation of Connector Data Loads. When the Harvest folder loads a file, it is automatically cleared and an archive folder with a copy of the loaded file is created. To add or delete files from the Harvest folder, you must have Administration rights or have the ManageFileShare security role.

> **NOTE:** The archive folder and the files within it can be cleared using Data Import Schedule Manager.

Content

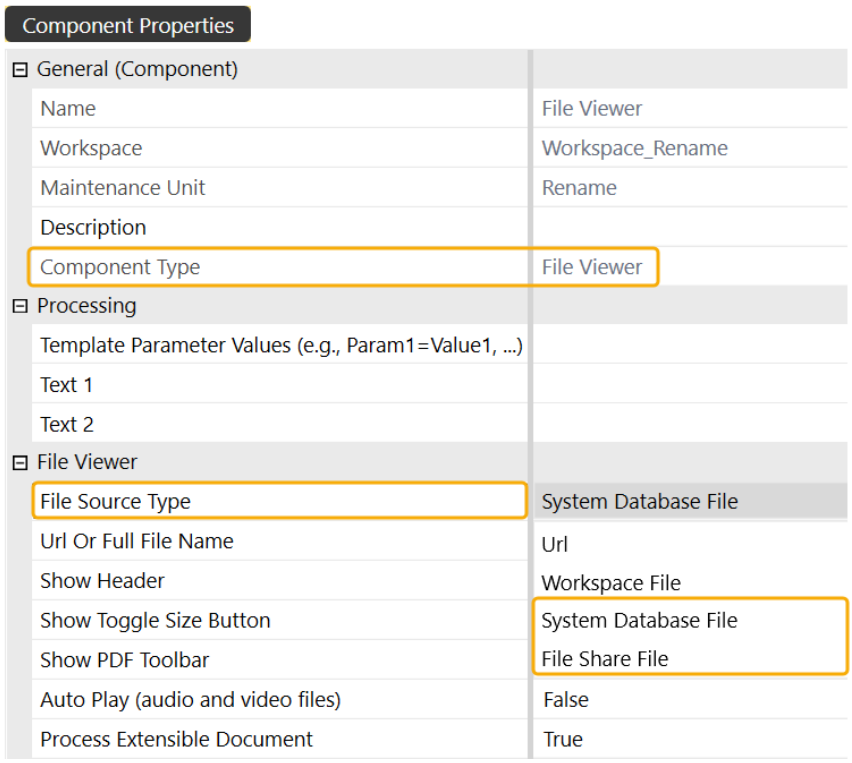
See Content Folders.

Data Management

This is the default folder for the data management export process.

Groups

This folder is used for the File Viewer Component to allow viewing by Security. The file is placed in one of the folders under \\MyServerName\OneStreamShare\Applications\MyApplicationName\Groups and then placed into a folder with the same name as a User Group. The user must have permissions to that User Group to view the file. See File Viewer.



Component Properties	
⊟ General (Component)	
Name	File Viewer
Workspace	Workspace_Rename
Maintenance Unit	Rename
Description	
Component Type	File Viewer
⊟ Processing	
Template Parameter Values (e.g., Param1=Value1, ...)	
Text 1	
Text 2	
⊟ File Viewer	
File Source Type	System Database File
Url Or Full File Name	Url
Show Header	Workspace File
Show Toggle Size Button	System Database File
Show PDF Toolbar	File Share File
Auto Play (audio and video files)	False
Process Extensible Document	True

Incoming

This folder is used as a location to place source files for import into Stage.

It is also used for the File Viewer Component. During an Import Load step, click **Load and Transform** > **Select Server File**. Click the ellipses to display source files placed in the Import folder. See File Viewer.

Internal

This folder stores the contents of the Application Database files or System Database files.

These icons enable users to export this content to File Share.

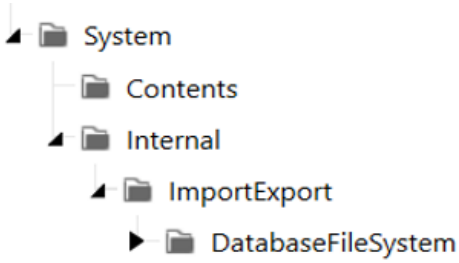
> **NOTE:** To access the File Share export and import options, the user must be granted rights to the SystemPane role and the FileExplorerPage role. When granted these rights, go to **System** > **Tools** > **File Explorer** and the icons will display in the toolbar.

Outgoing

This folder is available for use in customized processes that require a defined folder location.

System Folders

This folder has system specific storage.



Content

See Content Folders.

Internal

This folder exports from System Database.

Uploads

By default, when this flag is set to **True**, authorized users can upload or edit files or folders in the File Share using the OneStream File Explorer (required to use Content folder). When set to **False**, no users will have upload, edit file, or edit folder access to the File Share using the OneStream File Explorer. However, normal access rules will apply for browsing files and folders in the File System section.

Users will receive a Security error when attempting to write or edit the files and folders while attempting to use API or non File Explorer methods.

Supported File Sizes

The OneStream File Explorer can be used as an application interface to the File Explorer's Content folders. The supported file size varies by interface.

Windows Application

- Uploads (Applications and System): Up to 300 MB
- Uploads (Content): Up to 2 GB
- Downloads (Application and System): Up to 2 GB
- Downloads (Content): Up to 2 GB

Whitelist File Extensions

Whitelist File Extensions provide a method of identifying which kinds of documents, according to their file extensions, are allowed to be saved into the OneStream application File Explorer. Whitelisting extensions helps to alleviate the risk of uploading malicious file types into the application.

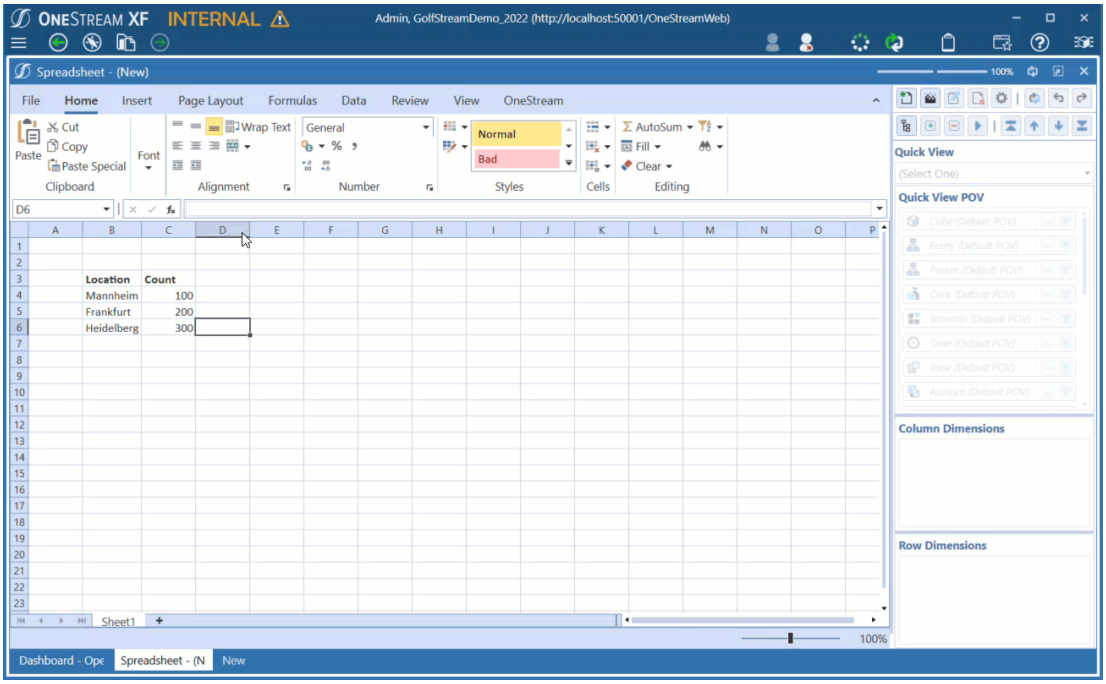
Overview

OneStream lets you save documents or upload documents from your device into the application. These documents are stored in one of three file storage root folders:

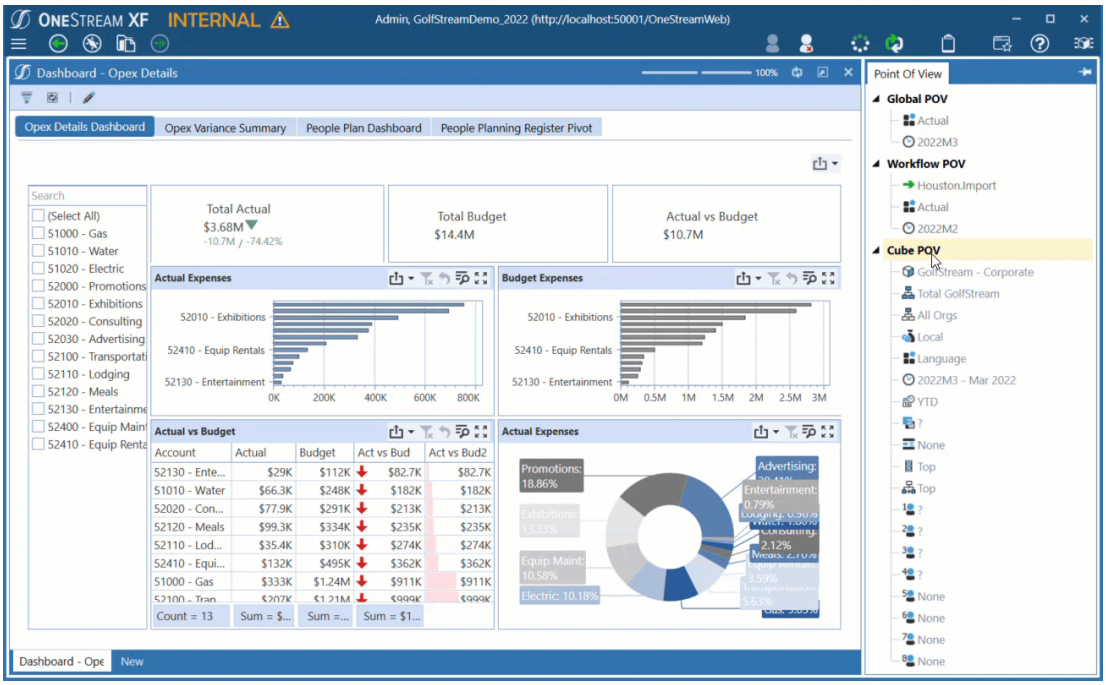
- Application Database: Displays stored documents for the current application only.
- System Database: Displays stored documents for the entire system without affecting the current application.
- File Share: This is a self-service directory that temporarily stores files before they are imported into the Application or System Database.

File Explorer is the product tool inside the Windows Application that lets you browse saved documents or upload new ones.

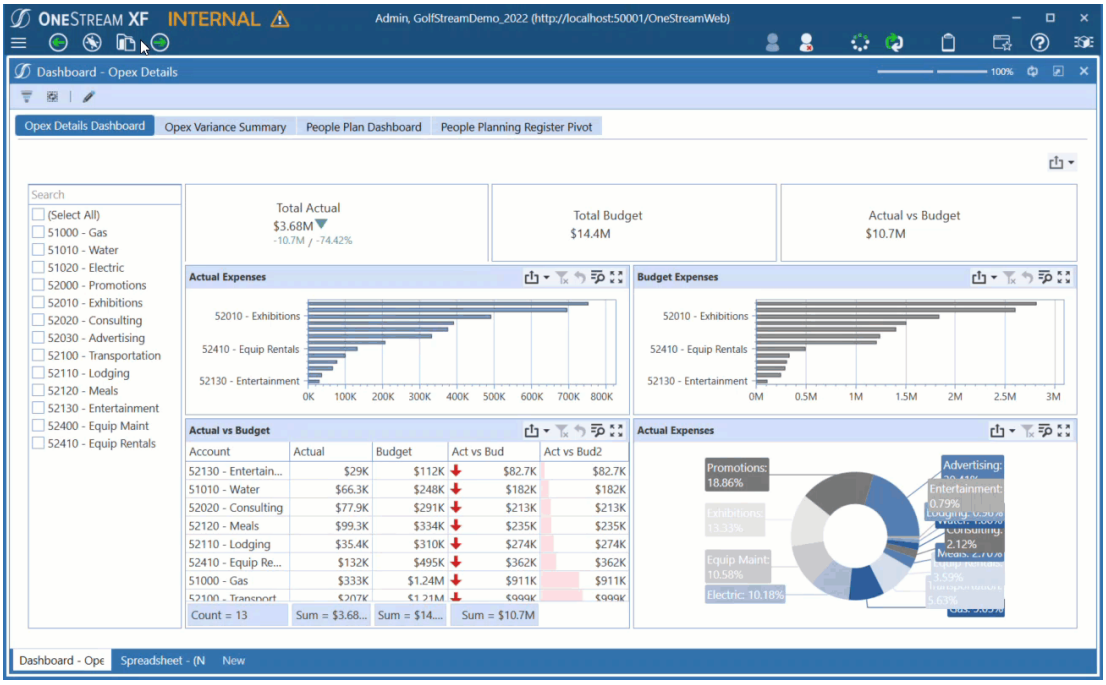
For example, a spreadsheet can be saved to any of the three root folders. A Cube POV can be saved to either the application or system folders.



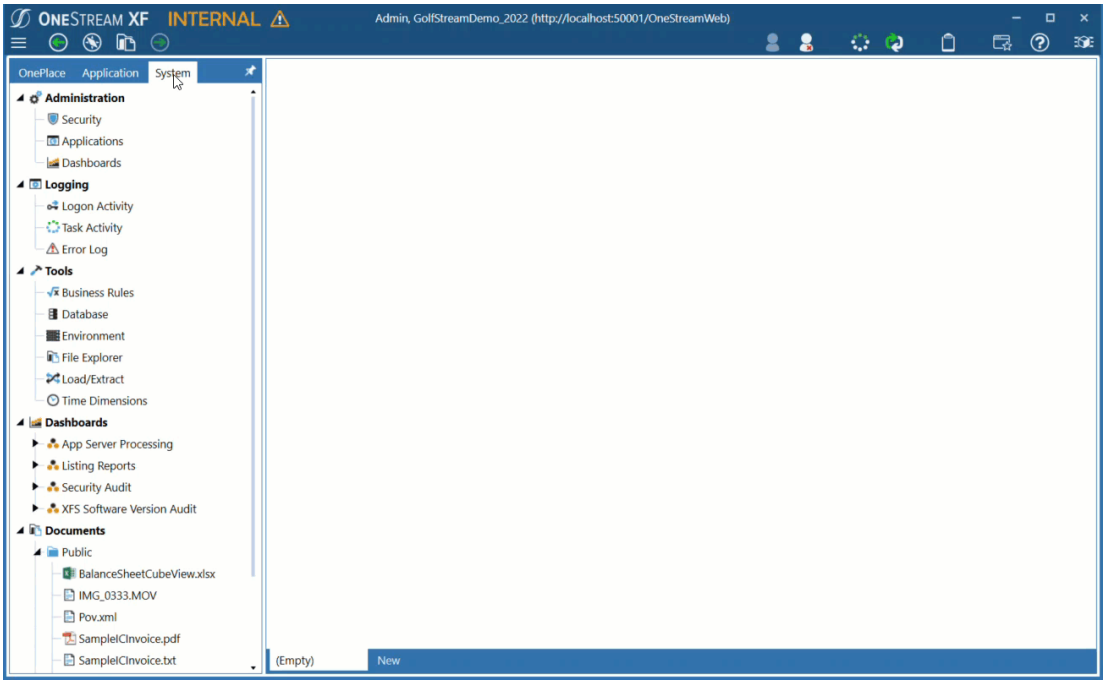
A Cube POV can be saved to either the application or system folders.



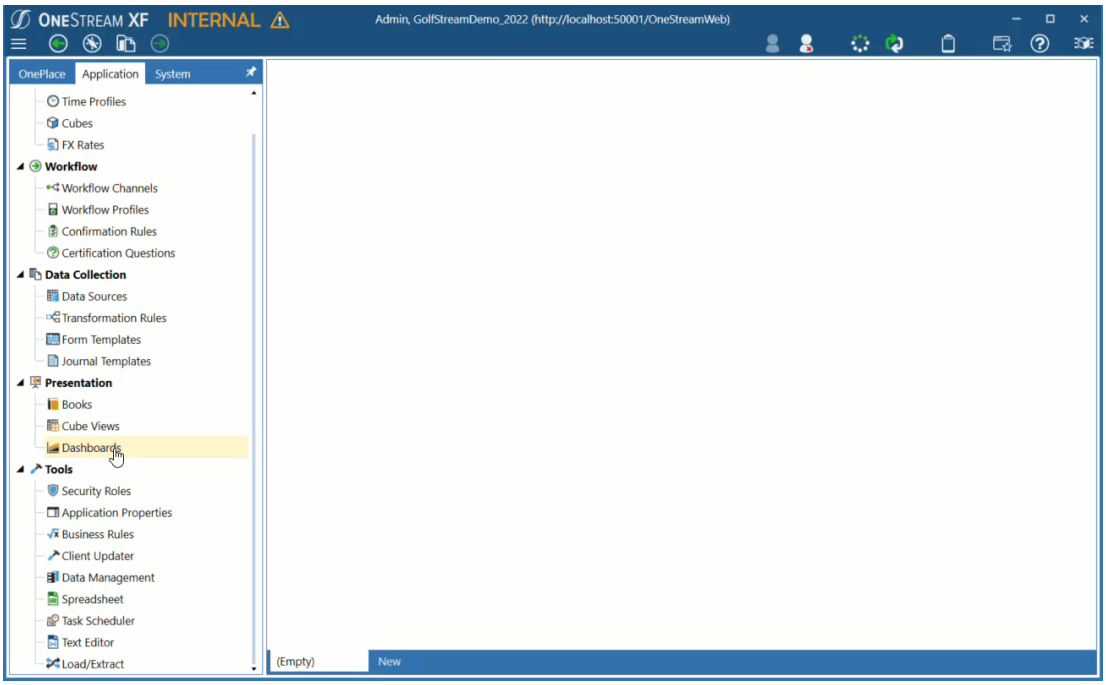
File Explorer is available from the main toolbar.



File Explorer is also available on the System tab.



The Dashboard component “File Viewer” also launches the File Explorer component.



Finally, there is a BRAPI used by OneStream Solutions or custom-developed dashboards to upload documents.

Whitelist Files Extensions Prerequisites

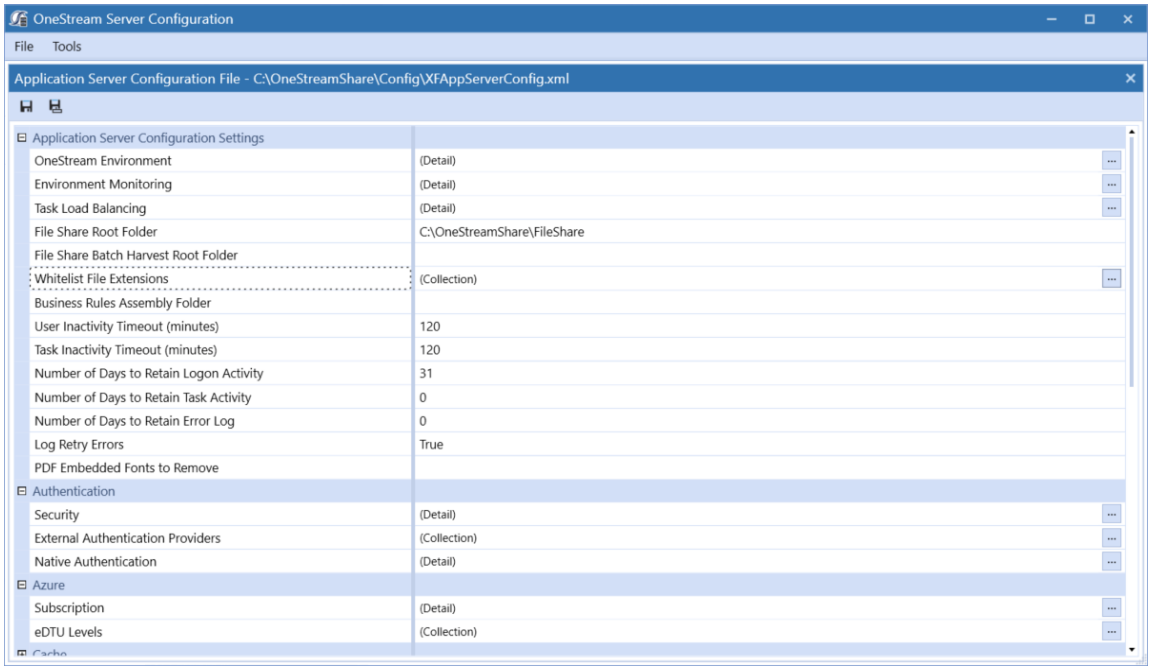
Standard File Explorer setup and security.

Define the List of File Extensions

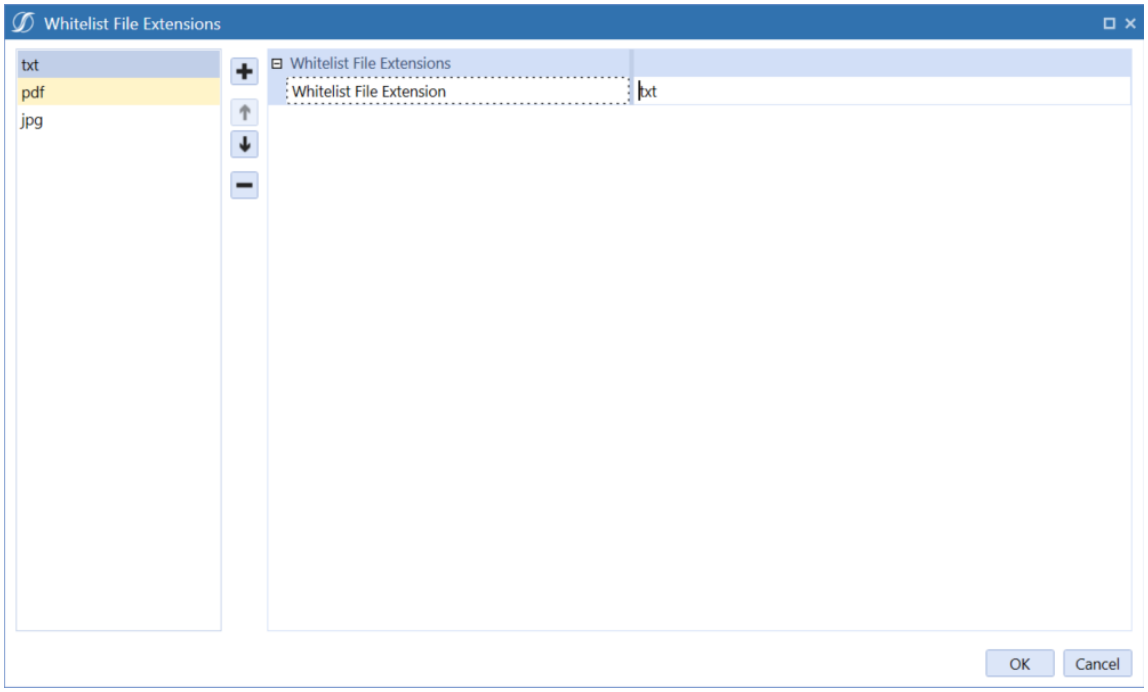
Whilelisted file extensions must be specified in the OneStream Application Server Configuration file.

To enable the whitelist and define allowed file extensions:

1. Go to **Start** > **OneStream Software** > **OneStream Server Configuration Utility**, right-click and select **Run as Administrator**.
2. On the File menu, select **Open Application Server Configuration File**.
3. Browse to the location of your XFAppServerConfigFile (typically located at C:\OneStreamShare\Config), select the application config file, and click **Open**.
4. In the Application Server Configuration Settings section, locate Whitelist File Extensions and click the ellipses (**...**) on the far-right of the field.



5. In the dialog box, click the plus sign (**+**) then type the whitelisted file extension. For example, .txt.



6. Continue adding other whitelisted file extensions.
7. When you are finished, click **OK**.
8. Click **Save** to save the changes to the Application Server Configuration File.
9. Restart Internet Information Service (IIS).

> **NOTE:** Cloud customers should contact support to have this configuration change made for them.

Also, if you enter any of the following characters, they will be automatically removed. For example, typing .txt (with a period) becomes txt.

/	^	=	'

(empty space)	&	\	[
!	*	?	]
@	(	<	.
#	)	>	
$	–	~	
%	+	`	

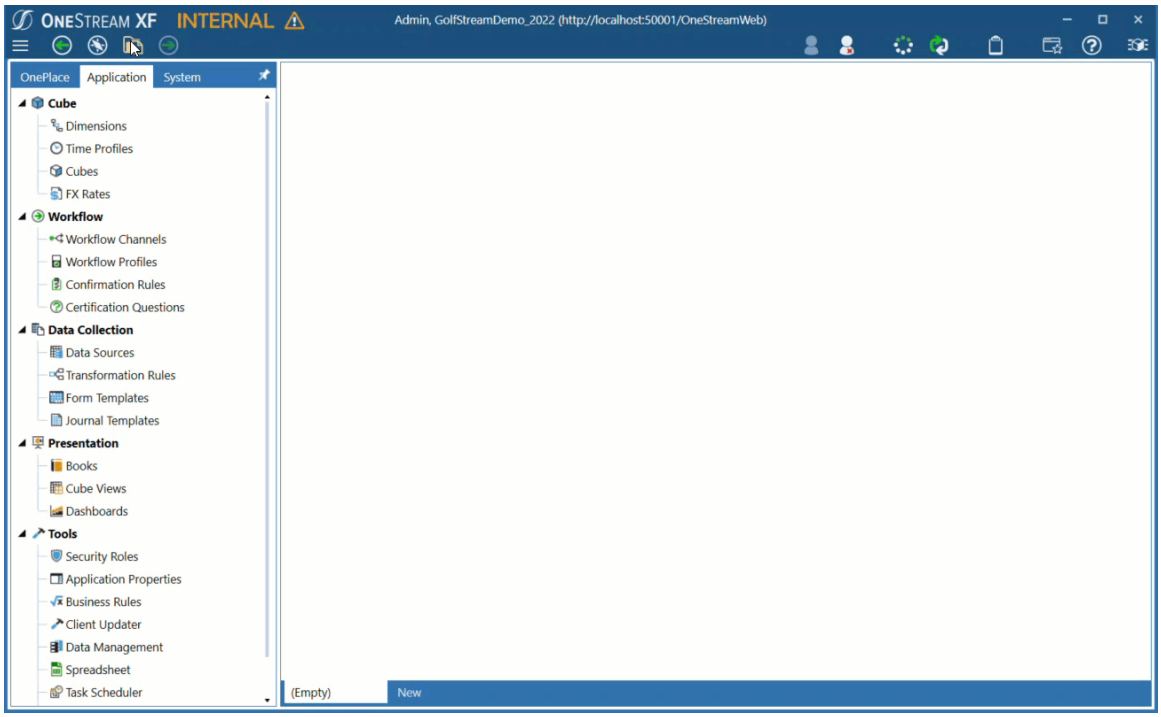
Add Documents using File Explorer

To add documents using File Explorer:

1. Navigate to any File Explorer component inside the Windows Application.
2. Select **Upload File**.
3. Browse to the file you would like to upload and click **OK**.

   If the extension of the file is whitelisted, it will be uploaded. If not, you will receive an error

message.



Troubleshooting

After a whitelist is defined, new files whose extension is not in the whitelist will be restricted from uploading.

If you upload a file that contains an exploit, malware, or a malicious script or macro, other users can download the file and infect their machine.

Conclusion

Whitelisting a file extension provides a means of allowing only certain types of documents to be saved in the File Explorer. This helps to alleviate the possibility of introducing malicious file types into the system.

Application Database/System Database

The Application Database displays stored documents for the current Application only. The System Database displays stored documents for the entire system without affecting the current application.

Documents

Public
Files available to everyone with access.

Security Access
Enables a user to access an object and read the content.

Maintenance (Group)
Enables a user to view an object and create, modify and delete objects in Groups. If in a Maintenance Group, a user does not need to be in the Access Group. The Maintenance Group also controls user profile contents.

Users
Private documents only available to named users.

Internal
Files used internally that cannot be modified.

Load/Extract System Artifacts

You can import and export sections of the System using an XML Format.

> **TIP:** Only System Administrators have access to this portion of the tool.

Extract

Choose an item from the drop-down list, click the Extract icon to start the extract process, then name the output file.

Load

After browsing to the file, click the Load icon to initiate the process.

Extract and Edit

This option is available for all extracts and allows the ability for the end user to edit the XML file as needed.

The following items can be extracted:

Security
This covers System Roles, Users, Security Groups, and Exclusion Groups which can be found under the System Tab |Administration| Security. The screen will display Extract Unique IDs. If this box is checked, the unique IDs OneStream assigns to each user will be extracted with the security. When moving security changes from one OneStream environment to another, trying to load the security with the unique IDs into the destination environment can result in an error if some of the records already exist. If this is the case, uncheck this box and extract without the unique IDs. Items to Extract also displays and by default All is chosen. Choose specific items in each section or turn the sections on or off.

System Dashboards

This has multiple sections such as Maintenance Units and Profiles. Under Maintenance Units, the Groups, Dashboard Components, Dashboard Adapters, and Dashboard Parameters all go together. System Dashboards can be found under the System Tab| Administration| Dashboards. The screen will display Items To Extract and by default All is chosen. Choose specific items in each section or turn the sections on or off.

Error Log

This covers the Error Log found under System Tab| Logging| Error Log. The screen will display an Extract All Items check box along with a Start Time and End Time. If the check box is notchecked, choose the Start and End Time by clicking . Click on the extract button and define where the file will be saved.

Task Activity

This covers the Task Activity found under System Tab > Logging > Task Activity. The screen will display an Extract All Items check box along with a Start Time and End Time. If the check box is not checked, choose the Start and End Time by clicking . Click on the extract button and define where the file will be saved.

Logon Activity

This covers the Logon Activity found under System Tab| Logging| Logon Activity. The screen will display an Extract All Items check box along with a Start Time and End Time. If the check box is not checked, choose the Start and End Time by clicking . Click on the extract button and define where the file will be saved.

Time Dimensions

Applications can have a monthly or weekly Time Dimension. Time Dimension Types determine if an application uses a weekly, monthly or 12/13 period frequency, and the type of calendar (such as 445 or 454). In a new application, use a custom Time Dimension Type so users can specify the number of months in a quarter and the number of weeks in a month.

After you create a Time Dimension, an XML file is generated. New applications use this file to implement the desired Time Dimension. If a Time Dimension Type is not assigned to a new application database, the default Standard Type is used.

> **NOTE:** Applications created prior to 4.1.0 can convert a monthly application to weekly application using the Database Configuration Utility. Contact Support for assistance.

Time Dimension Types

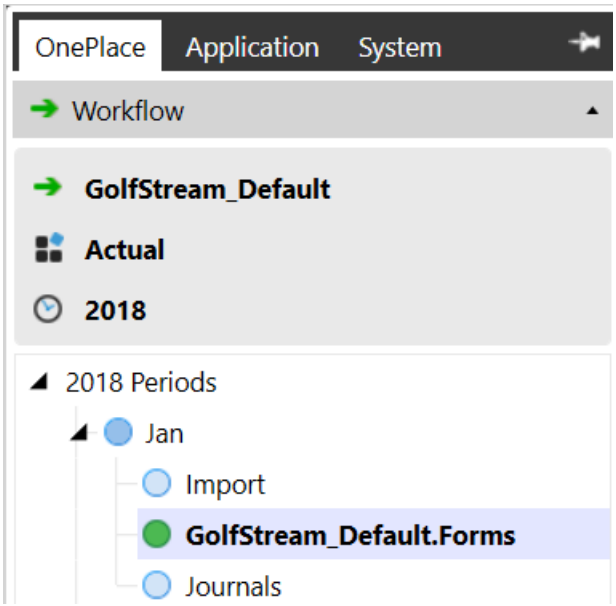
- **Time Dimension Type Standard**: Creates a Monthly Time Dimension and stores data by month in the data tables. Applications created prior to version 4.1.0 use this type.
- **StandardUsingBinaryData**: Creates a Monthly Time Dimension and stores data in a binary data table. Use this type if you may need to later convert an application to a Weekly Time Dimension.
- **M12_3333_W52_445**: Creates a 12 Month, 4 quarter, 52 Week, 445 calendar Time Dimension.
- **M12_3333_W52_454**: Creates a 12 Month, 4 quarter, 52 Week, 454 calendar Time Dimension.
- **M12_3333_W52_544**: Creates a 12 Month, 4 quarter, 52 Week, 544 calendar Time Dimension.
- **M12_3333_W53_445**: Creates a 12 Month, 4 quarter, 53 Week, 445 calendar Time Dimension.
- **M12_3333_W53_454**: Creates a 12 Month, 4 quarter, 53 Week, 454 calendar Time Dimension.

- **M12_3333_W53_544**: Creates a 12 Month, 4 quarter, 53 Week, 544 calendar Time Dimension.
- **Custom**: Creates a Time Dimension you can use to specify the months in a quarter and the in a month. This is only available for new applications.
- **Use Weeks**: If true, you can define a Custom Weekly Time Dimension. If False, you can create a Custom Monthly Time Dimension.
- **Vary Settings By Year**: If True, you can specify the number of weeks in a month for each year. If False, you can apply the number of weeks per month to each year.
- **M1- M16 Number of Weeks**: Specify the number of weeks per month. This coincides with the number of months per quarter, so any additional weeks required are enabled. If Vary Settings By Year property is True, select a year and customize the number of weeks in that year.

Using OnePlace Workflow

The top of this slider configures the Workflow View for data loading: Workflow Profile, Scenario, and Year. Clicking on any of the three items displays a pop-up window to select Workflow Profile, Scenario, and Year.

A Workflow Unit is an individual period within the selected year and Scenario combination for a particular Workflow Profile. Twelve Workflow Units or periods are always displayed for selection, but only one can be selected and active at a given time. Each Workflow Origin step available must be completed as well as the collective data confirmation and certification process for a single Workflow process to be finalized.



> **TIP:** The available tabs will be determined by the security level.

Right-Click Options

Right-click a Workflow month, Workflow Input Type, or Dependent Status cell to display the following options. Not all options are available for every object.

Status and Assigned Entities

Displays the Workflow Status of each Origin process (Import, Forms, and Journals). See Dependent Status under Certify for more details on this feature.

Audit Workflow Process

Right-click any Workflow channel. This provides an audit for every Workflow task's process including the date and time of the process, the user performing the process, how long it took for the process to complete, and any errors that occurred during the process. It also provides audit history through Lock History > Workflow Lock/Unlock.

Lock/Lock Descendants

This will lock a particular Workflow period, or Origin process.

Unlock/Unlock Descendants

This will unlock a particular Workflow period, or Origin process.

Edit Transformation Rules

This will navigate to the Transformation Rules screen and allow a user to fix any Transformation Rule errors that occur during the Workflow process.

Clear All 'Import' Data From Cube

This will clear all imported data from the Cube, but the data still remains in the Stage.

Clear All 'Forms' Data From Cube

This will clear all Forms data from the Cube.

Corporate Certification Management

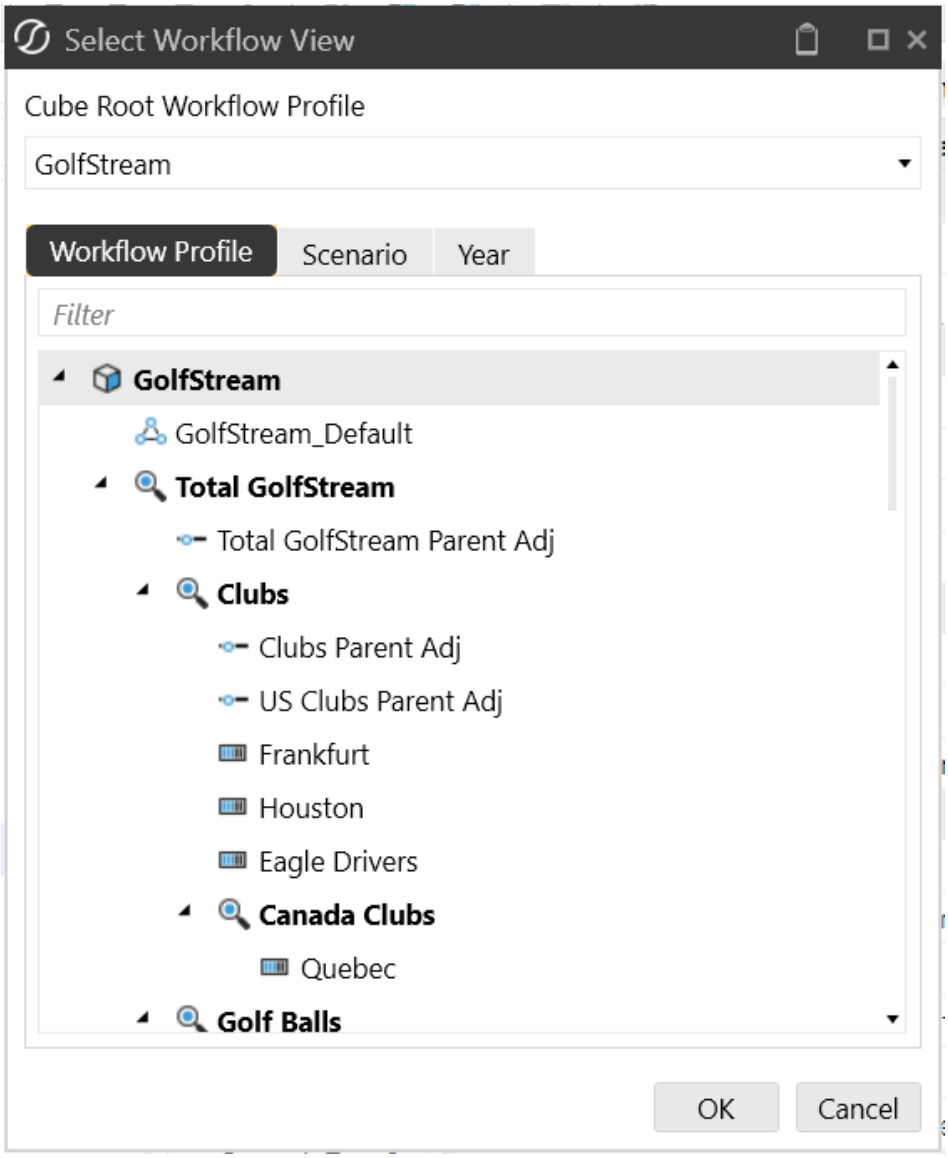
This allows a user to unlock and uncertify ancestors, or lock and certify descendants.

Corporate Data Control

This allows a user to preserve data if changes need to be made and restore the preserved data if necessary.

Workflow View Selection

A description can be added to a Workflow Profile during its design. Descriptions can only be added to Workflow Profiles using the Default Scenario Type. Scenario Type specific descriptions cannot be used.



Workflow Tasks

This simplifies the data collection, consolidation, and certification process. Importing data, entering a form, making a journal entry, and signing off on reports is completed through the Workflow process.

Import

This process step gives the end user the ability to import data into the system. This can be through a defined Data Source or a Data Connector. First click on the Import Workflow Origin located under the active month, then click Import in the task bar.

After a user clicks this icon, it will prompt him/her to either search for a file on the drive or initiate the Data Connector. The system will then import the data into the Stage engine. The file will be parsed into a clean tabular format with information on the Amounts, the Source ID, and each Dimension. Once the data is loaded successfully, the Import task will change from blue to green.

Load Method

Upon clicking Load and Transform, a dialog will appear with four Load Options:

Replace

This will clear all data for the previous file that correlates with the specific Source ID and replace it with the new file's data. This can be done even if the previous data has already been loaded into the Cube. Once the file is re-loaded, the user will need to complete all Workflow Tasks and load the new data into the Cube.

Replace (All Time)

Replaces all Workflow Units in the selected Workflow View (if multi-period). Forces a replace of all time values in a multi-period workflow view

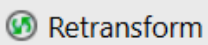
Replace Background (All Time, All Source IDs)

Replaces all Workflow Units in the selected Workflow View and all Source ID's in a background thread while the new file parse or connector execution is running. The delete is being performed while parse is being performed.

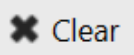
> **NOTE:** This load method always must always be used to delete ALL Source IDs. If the workflow uses multiple Source IDs for partial replacement during a load, this method cannot be used.

Append

This is used when additional rows are added to a source file and need to be loaded into the Stage. This will not change any of the data already loaded for the source file, it will only add rows that were not included in the previous file load.



This is used after data has already been loaded, but some changes have been made and the calculations need to be repeated.



This will clear all loaded data from the Stage.

View Last Source File Processed for Current Workflow Profile

View Last Log File Processed for Current Workflow Profile

When this icon displays, the Global POV is enforced and data loading is limited to the Global POV. The Enforce Global POV setting is controlled under Application|Tools|Application Properties.

When this icon displays, data cannot be imported to time periods prior to the current Workflow year. The Allow Loads Before Workflow View Year setting is controlled under Application|Tools|Application Properties.

When this icon displays, data cannot be imported to time periods after the current Workflow year. The Allow Loads After Workflow View Year setting is controlled under Application|Tools|Application Properties.

Right-Click Options

View Source Document

This will open the source file imported into the application.

View Processing Log

This will open a processing log and provide information on when and how the source file was imported.

View Transformation Rules

This provides all mapping rules for the specific intersection

Drill Back

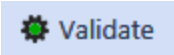
Using a SQL Connector allows a user to drill back to a source system and show detailed records from a document, PDF, website, within the application This option is only available if data was loaded using a Connector Data Source. For more details on this feature, see Drill Back in Data Collection Guides.

Export

This will export the data into an Excel XML< CSV, Text, or HTML file

Validate

After the data is loaded, the user will then validate the map and intersections. During the Validate step two specific actions happen. First, OneStream will check to make sure each piece of data has a map. Next, OneStream will check to make sure the combination of Dimensions can be loaded into the Cube based on the constraints defined in the system (e.g. using Intercompany). Click the green Validate task in the taskbar.



Click this icon to validate the data. If there are no errors, the Validate task will change from blue to green. If there are any errors, Validate will turn red and the errors will need to be fixed before proceeding.

Transformation Errors

The errors will be listed by the Dimension with the error, such as a new account that was added, but not mapped. The user will be able to see the Source Value and the (Unassigned) Target Value. The system will already know the Dimension and pull up the possible targets. On the right, the user will be able to use the search box and click the funnel/filter button to find the correct account. The default will be a One-To-One Transformation Rule. For more information on these rules, see Transformation Rules in Data Collection.

Save any changes made and then click the Retransform icon, this will automatically re-validate the data. If there is an intersection validation error, this will now appear. If there are no intersection errors, the Validate Workflow step will change from blue to green.

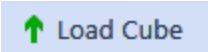
Intersection Errors

Intersection Errors mean something is not correct with the entire intersection of data. For example, a Customer Dimension may be mapped to a Salary Grade Account. This does not make sense and the system will identify this prior to the load. To fix this error, click on the bad intersection and drill down to see the GL/ERP Account; (see Drill Down in Using OnePlace Cube Views for more details). Right-click to View Transformation Rules and from there, investigate each rule for each Dimension in order to find which one does not make sense. Each Dimension will be on the left, and each Target Value Column needs to be investigated. To fix a Transformation Rule here, select the Target Value to correct, right-click and select Edit Rule. Choose the appropriate Target Value and fix the intersection error. Click Save and close the next two windows. Click the Validate icon again and if it was fixed correctly, the Validate task will change from blue to green.

See Right-Click Options under the Import Task for details on the right-click functions.

Load

This simply loads the cleansed data from the Stage to the Analytic Engine. Click the Load task in the taskbar.



Next, click the Load Cube icon. This now loads the data into the Consolidation Engine and completes the Import process.

A Task Progress dialog will appear:

A green check will appear next to the Import Origin Workflow Step.

See Right-Click Options under the Import Task for details on the right-click functions.

Pre-Process

The Pre-Process task is the same as the normal Process task. This can be added as a first step in the Form channel process. Please refer to the Process description further down in this section for more details.

Input Forms

Once in this step, there will be two options under Workflow Forms: Required and Optional. The required Forms will have to be completed before the end user can move on from this step. Click on the specific Form, and manually enter the required data.

Open Excel Form

A user can export a Form derived from a Cube View to Excel, complete it in Excel, and then submit the data back to the application.

Attachments

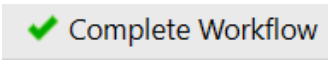
This allows the user to attach supplemental files to specific Forms.

Import Form Cells Using Excel or Comma Separated Values File

Click this to load Form data via Excel template or CSV template. See Loading Form Data in Data Collection Guides for more details on using these templates.

Import Cell Details Using Excel or Comma Separated Values File

Click this to load Cell Details via Excel template or CSV template. See Loading Cell Detail in Data Collection Guides for more details on using these templates.



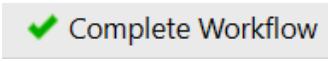
Once a Form is completed, click Save and then the Complete Form icon.

> **NOTE:** A drop-down time period menu will display for a form with a defined time filter containing multiple periods. If a Form does not have a time filter defined, this will be a standard button.

This will re-open a Form and clear the data. Once the Form is updated, Complete Form must be clicked again.

> **NOTE:** A drop-down time period menu will display for a form with a defined time filter containing multiple periods. If a Form does not have a time filter defined, this will be a standard button.

Checking this box enables all Forms to be completed/reverted in one step.



When each Form is completed, click Complete Workflow and a green check will appear for the Forms Origin Workflow Step.

This will re-open the Workflow. To make changes to the Form, Revert Form must also be clicked. Make any necessary changes to the Forms and click Complete Form/Workflow again.

This will run a consolidation/translation/calculation for a specific Form.

Show Report

This shows the Form in a Cube View Report.

Export to Excel

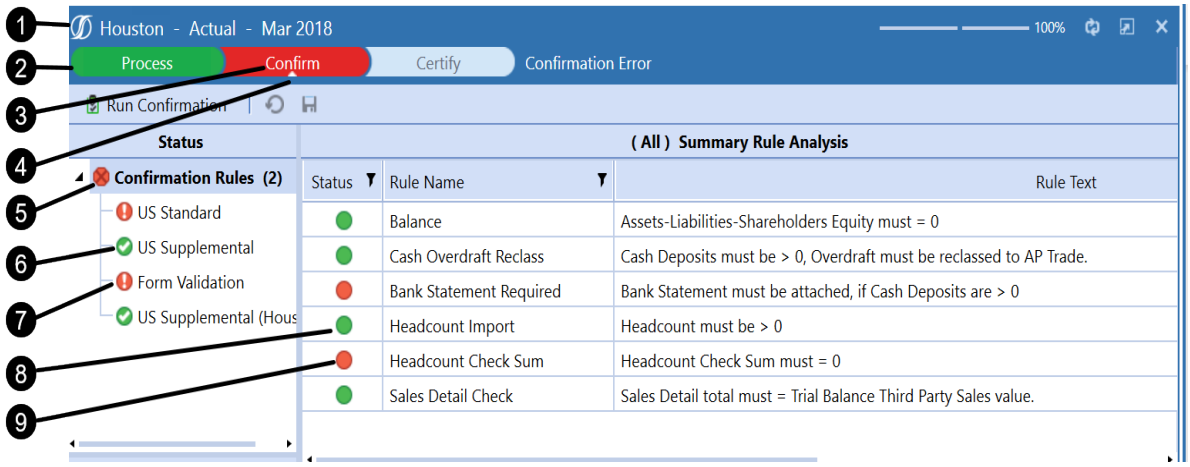
This exports a copy of the Form into Excel.

Cell Detail

Cell Detail can be entered on a Cube View used for a data entry form, or on a Cube View or Quick View in Excel. Cell Detail is available on any writeable O#Forms or O#BeforeAdj Member.

See Right-Click Options in Using OnePlace Cube Views for details on the right-click options in Forms.

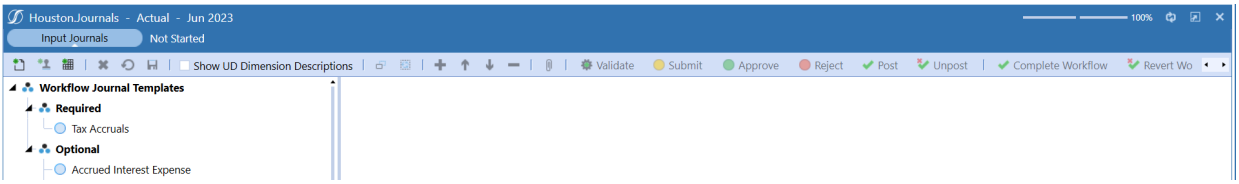
Workflow Icons



1. OneStream logo is white.
2. Workflow Status bar steps have curved edges.
3. Failed task text is white.
4. Up-arrow indicator for selected task.
5. Stop sign has an X in the middle, and a flat design style.
6. Confirmation Rules; Pass = green circle white checkmark, flat design style.
7. Confirmation Rules; Fail = red circle white exclamation mark, flat design style.
8. Status; last step completed green circle.
9. Status; last step error red circle.

Journals Toolbar

The Journals toolbar contains many tools for working with journals. This section describes each tool in detail.



Input Journals

Once in this step, there will be two options under Workflow Journal Templates: Required and Optional. The required Journals will have to be completed before you can move on from this step.

Create Journal

Click this to create a Journal if a template is not available. If this button is disabled, it is because the workflow profile is configured to only allow you to create journals based on existing templates.

Create Journal Using Template

Use this to create a Journal using the selected template. Manually enter the required Journal data and Click Save.

Create Journal Using Excel or Comma Separated Values File

Click this to load a journal entry via Excel template or CSV template. See Loading Journal Data in Data Collection Guides for more details on using these templates.

Copy Selected Journals

Copy existing journals into other time periods and scenarios. This is useful when you are creating similar journals, managing recurring journals, or copying Actual data into budget or what-if scenarios.

1. Select the journals you want to copy and click **Copy Selected Journals**. You can select journals in any status but when copied, they will be set to Working status in the new time period.
2. Click the drop-down arrow in the Target Scenario field.
3. Make a selection. The selection applies to all journals in that single-batch copy.
4. For each journal, select one or more time periods.
5. Type a new journal name and description.
6. You can include attachments by placing a check in the Include Attachments checkbox.

Administrators and members of the Journal Process Group can copy journals into a specified Adjustment Workflow Profile.

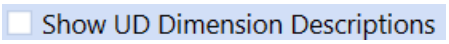
> **NOTE:** After journals are copied, WfProfile_WfUnit_Scenario_TimePeriod_Index is appended to the journal name.

Delete Selected Item

Delete one or more journals simultaneously. Journals in Working or Rejected status can be deleted. This saves time when you need to remove multiple journal entries quickly and efficiently. Select the boxes next to the journals you want to remove then click this button and confirm.

Reapply Template Settings to Journal

This will clear the data from the selected Journal and return it to the original template.



Show UD Descriptions

Check this checkbox to display descriptions for the eight UD Dimension Types. They will display next to the Dimension Type in the Journal POV and the line item headers. Descriptions are defined in the Application Properties, and can provide clarity on the purpose of the dimensions.

> **NOTE:** This feature applies to the Journal page only. Reports, BRApis and Excel/CSV templates related to journals remain unchanged.

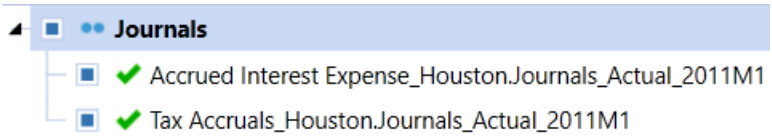
Export

Export journals from within the journals page. This is useful when you are more comfortable using a spreadsheet tool to edit journal data. You can export data from one area in OneStream and import it into another area or application.

1. Click **Export** to open the **Select Journals to Export** dialog. Workflow profile and scenario are locked.
2. Type a **Time Filter** using the T# syntax. For example, T#2023M1, or for multiple time periods, T#2023M1,T#2023M2.
3. Type a **Journal Status**. Select a single journal status or select **All** to include all statuses.
4. Type a **Journal Name**. This optional step will find all journals that contain the text you type.
5. Click **OK** to generate an XLSX file with journals that match the selected criteria.
6. In the **File Explorer** dialog, browse to where you want to save the XLSX file. Data from all journals that match the selection criteria is saved.

Journal Entry Checkboxes

Select multiple journal entries in the same state (e.g., completed, posted, approved, etc.) in order to Submit, Post, Approve, or Quick Post them at the same time. Select multiple journal entries in either the Working or Rejected status to delete them at the same time.



Validate

You can ensure that dimension members selected in the journal are valid cube intersections based on constraints configured in the dimension library. If validation errors exist, they will display in a Validation pane allowing you to view them as you make edits to the journal. This helps you save time by validating entries earlier in the process.

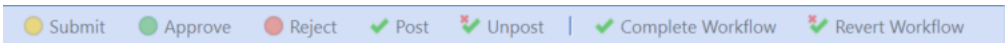
You must have the Journal Processing security role to use this feature and dimension member constraints must first be configured in the dimension library.

To use this feature:

1. Open a journal that is in Working status.
2. Click **Validate**.

If invalid data is found, the Validation pane will display messages that detail each entry error.

Once a Journal is completed, select Quick Post or Post. When all required Journals and any optional Journals are finished, click Complete Workflow and a green check will appear for the Journals Workflow Step.



Depending on the security configuration, there are multiple options for Journals. If full security is in place, the end user will be able to create and submit. The approver will approve or reject, then the end user can post and complete the Workflow.

Journal Line Items

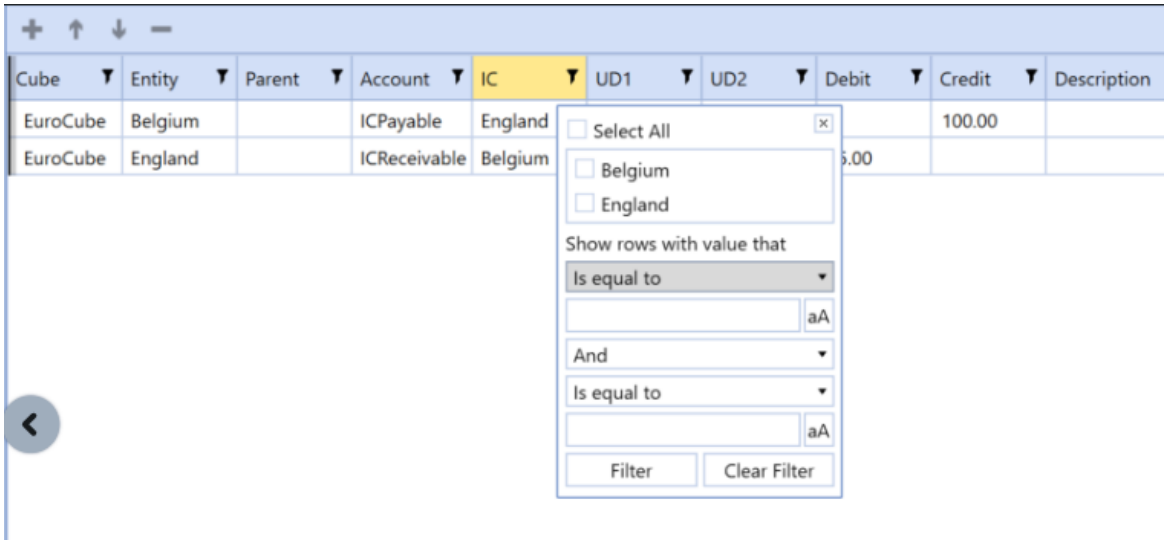
Journal data is entered into line items. You can add line items to reflect the Dimensions they are adjusting. Any dimensions not in line items must be configured in the Journal POV.

To select a dimension member in journal line items:

1. Double-click the Entity, Account, Flow IC, or UD1-UD8 field.
2. If you know the member name, type it into the text field. Repeat for all fields as needed.
3. If you do not know the member name, use the Select Member dialog to locate the member.

> **NOTE:** If the dimension members have a description, the members are displayed in the journal line items. For each dimension, you can set a preference to hide or show the description in journal line items. These preferences are saved on your local machine in OneStream user settings.

You can filter journal line items by clicking the filter icon in the column headings. A drop-down selection box appears where you can choose from various filtering options.



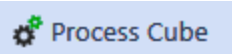
This is useful when you are working with journals that contain a large amount of line items, making it easier to locate a specific subset of line items.

There are four line item actions buttons that sit just above the line item entries that aid in editing the journal line items:

- **Add**: Creates a new journal line item. If an existing line is already selected, Add copies the selection. If no line is selected, the new line item is blank.
- **Up**: Moves the selected item up in the list.
- **Down**: Moves the selected item down in the list.
- **Remove**: Deletes the selected line item.

Process

Once in this step, click Process Cube in order to process the loaded data.



This icon performs No Calculate and all the standard Calculate, Translate, and Consolidate options using Calculation Definitions. Each added line item can be filtered by entity for reviewer level processes. Once the Cube is processed, the Process task will change from blue to green and move to the Confirm task. No Calculate can be used to allow the assignment of Data Management Sequences.

Confirm

This step runs the Confirmation Rules defined for this particular Workflow. This will immediately inform users if they have passed or failed the data quality rules. The two types of statuses for this step are Warning or Error. Warning means the user is outside of the threshold, but it will not stop the process. Error means the user is outside of the threshold and this will stop the process and turn this step to red. If anything has failed, revisit one or many of the previous steps to make sure the data is accurate and complete. Once the data has passed all quality rules, the Confirm task will change from blue to green and move to the Certify task.

Certify

This is typically the final step in the Workflow process. This certifies and completes the phase of the Workflow.

Some questions may need to be answered regarding the processes. Click each set of questions under the Questionnaires area. Answer the questions by clicking in the response cell and selecting the correct answer. Comments can also be added in order to explain the answers. The status will be displayed on the right. When this is completed, click on the Set Questionnaire Status icon and select Completed and then OK.

Set Certification Status

After each group of questions is completed, the Set Certification Status icon will be enabled. Click this and select Certify in order to certify the data as complete and accurate. This will give the final green check for the month being processed and the data can now be trusted as complete and accurate by any stakeholder that is analyzing this information.

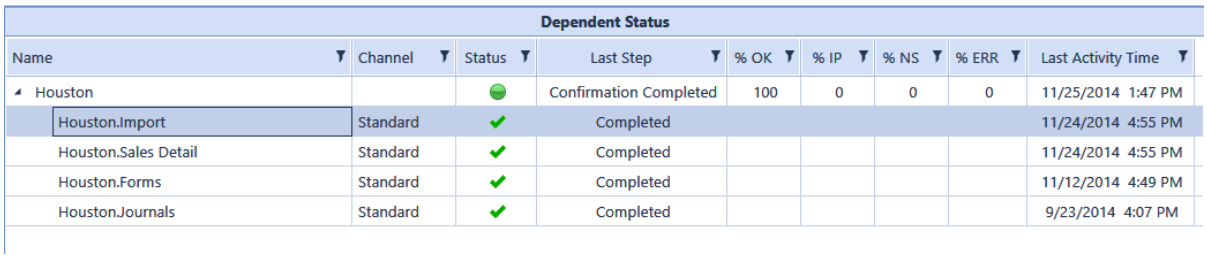
Quick Certification

This is a one click option to expedite the Certify process. No questions need to be answered. This will give the final green check for the month being processed and the data can now be trusted as complete and accurate by any stakeholder that is analyzing this information.

This data can now be used for Consolidations by users or managers responsible for this workflow. They can look at data as it moves up and perform their own top side adjustments, confirmations, and certifications at as many levels as is appropriate for the organization.

Dependent Status

Click on Dependent Status to see the status of each required Workflow task to ensure they are all completed. This will display the Workflow Profile name and all input types, the Workflow Channel, the status of each input type, the last step completed for each input type, the percentage of each step that is OK, In Process, Not Started, and steps with errors, and a record of when the last activity took place for each step. See Right-Click Options for details on these right-click functions.



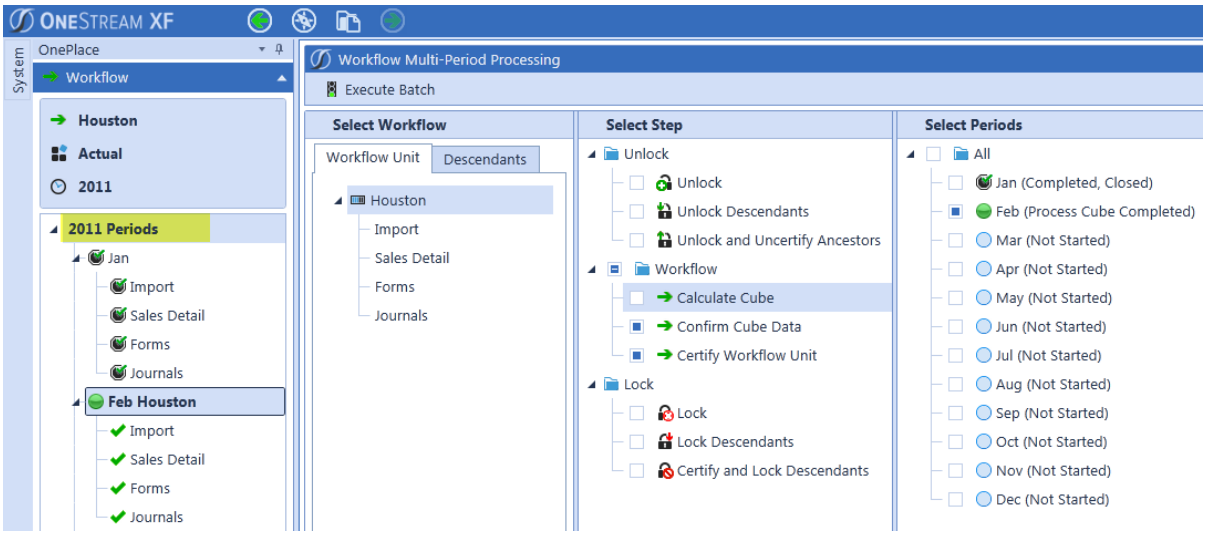
Dependent Status

Name	Channel	Status	Last Step	% OK	% IP	% NS	% ERR	Last Activity Time
Houston			Confirmation Completed	100	0	0	0	11/25/2014 1:47 PM
Houston.Import	Standard	✔	Completed					11/24/2014 4:55 PM
Houston.Sales Detail	Standard	✔	Completed					11/24/2014 4:55 PM
Houston.Forms	Standard	✔	Completed					11/12/2014 4:49 PM
Houston.Journals	Standard	✔	Completed					9/23/2014 4:07 PM

For more details on Workflow, see Workflow in Workflow.

Multi-Period Processing

Click on the Year in the Navigation Pane to enact Multi-Period Processing options:



From here, perform multiple workflow tasks for one to many time periods, as shown above.

Analysis Pane

In each workflow profile for a defined period there will be an Analysis Pane under the Status Pane. For example: under Process, Confirm, and Certify tasks.

Cube Views and Dashboards for Analysis

This is where data can be viewed and analyzed in cube views and dashboards. If there is a grid in any dashboard or cube view, these are available for further drill down or annotation by right clicking the cell. See Using OnePlace Cube Views and Using OnePlace Dashboards for more details.

To view the Cube Calculation Status, click on Cube Views| Calculation Status to show a data grid presenting the Calculation Status of the current active Workflow POV. This will be available at the total Monthly (not an Origin process) Review, Process, or Certify tasks. See Calculation Status in About the Financial Model for more details on this feature.

Intercompany Matching

IC Matching which will show any Intercompany discrepancies. If the button is red in the status column, click the item to see the details. Each Intercompany counterparty will be visible along with counterparties with an Intercompany variance. By clicking on the counterparty, details including the Reporting Currency, Entity Currency, and Partner Currency will be visible.

Select the Intercompany Partner to see the status of the Intercompany issue and any annotations the partner may have made. Select the Difference row to see both parties' statuses and annotations.

A red triangle indicator displays in the cell when data is dynamically derived. This generally occurs when the intercompany matching report performs currency translations.

Right click on the Entity row to perform the following functions:

Set IC Transaction Status

This allows the user to update the status by selecting Not Started, Loaded, Adjusting, Disputed, Finalized. The user may also include comments for the counterparty to see.

Show Partner Workflow Status

This allows the user to see the Workflow status of the counterparty.

Show/Hide Dimension Details

This allows the user to see the Dimension details for the Intercompany accounts.

Drill Down

This will allow the user to drill down on the Dimensions in order to get more information about the Intercompany data.

See Drill Down in Using OnePlace Cube Views for more details on this feature.

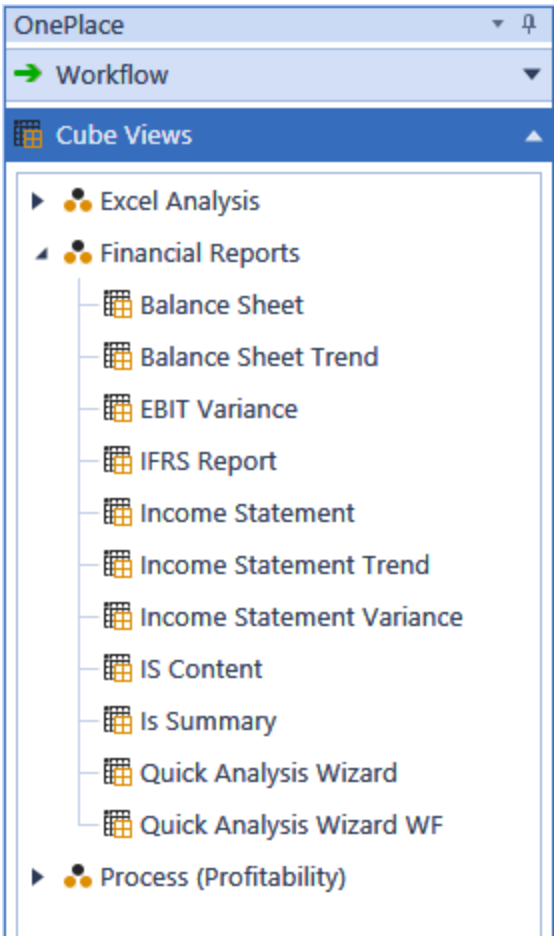
See Intercompany Eliminations in About the Financial Model for more details on Intercompany.

Using OnePlace Cube Views

These Cube Views are complete and organized reports created specifically for the company's data in the Application section. A Cube View is used to query Cube data and present it to the user in variety of ways. They can be made read-only, used for editing data, and can be used as the Data Source for several different display mechanisms.

In order to copy Cube View cells from a Data Explorer Grid to an Excel spreadsheet, click CTRL, select the cells desired, and then click CTRL-C. Navigate to an Excel spreadsheet, select a cell, and click CTRL-V, this will paste the cells into Excel. This can also be done from an Excel spreadsheet into a Data Explorer Grid.

While viewing these reports, users can right click on any cell in order to learn more about any given number or piece of data. For more information on creating Cube Views, see Cube Views in Presenting Data With Books, Cube Views and Dashboards. For information on how to use Cube Views, or advanced uses, see Cube Views in Presenting Data With Extensible Documents.



Toolbar

Consolidate

This consolidates the data in the Cube View

Translate

This translates the data in the Cube View

Calculate

This performs a calculation on the Cube View data

Row Suppression

Use Default Suppression

This suppresses the data based on the Cube View's row suppression settings.

Suppress Rows

This suppresses any data rows with zeroes, no data, etc. regardless of the Cube View suppression settings.

Unsuppress Rows

This removes any data suppression set via the Cube View suppression settings and displays all data rows including those with no data, zeroes, etc.

> **NOTE:** The actions above vary based on the user's security settings and restriction properties set on the Cube View.

Show Report

This opens the Cube View in a polished, formatted report

Export to Excel

This will open the fully formatted Cube View in Excel.

Users can open multiple excel exports using a Cube View without being prompted to rename or save the file. A version number will change with each export in sequence.

Select Parameters

This allows the user to select new Parameters and view the Cube View data differently.

> **NOTE:** This is based on the Parameters set for this Cube View. This feature will not apply to all Cube Views

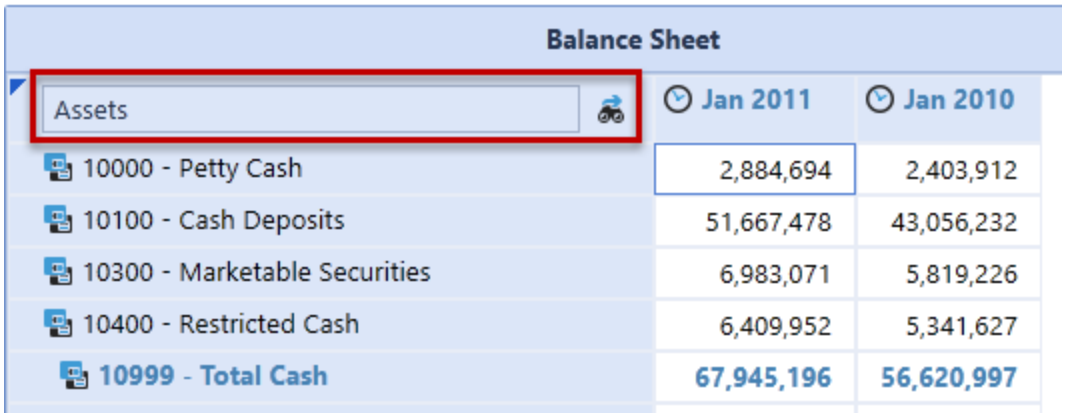
Edit Cube View

This opens the Cube View Application page where changes can be made to the Cube View design.

> **NOTE:** This varies based on the user's security settings.

Find Next Row

When viewing a Cube View with a large number of rows, use the search filter at the top of the screen in order to navigate to the desired row. Type a keyword into the search filter and click the Find Next Row icon in order to navigate to the preferred row. Continue clicking the icon in order to navigate to any row with the keyword included in the row header name. This also works when rows are collapsed.



Balance Sheet		
Assets	Jan 2011	Jan 2010
10000 - Petty Cash	2,884,694	2,403,912
10100 - Cash Deposits	51,667,478	43,056,232
10300 - Marketable Securities	6,983,071	5,819,226
10400 - Restricted Cash	6,409,952	5,341,627
10999 - Total Cash	67,945,196	56,620,997

Copy or Paste Data to Multiple Cells

Use <Ctrl+C> or <Ctrl+V> to copy and paste values into Cube View data cells. Select the desired data cell, click <Ctrl+C> to copy the cell value and then select another cell to paste <Ctrl+V> the value. Hold down <Ctrl> to select multiple cells and paste the value into the selected cells simultaneously.

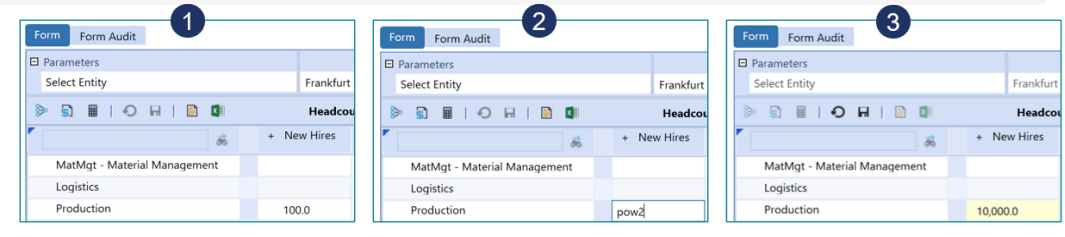
Shortcuts, Scaling, and HotKeys

The Cube View Data Explorer allows the use of shortcuts, scaling, and hotkeys.

Shortcuts

Use Shortcuts to calculate amounts within a cell. Click once in a cell, type in a shortcut, then click enter to run the calculation.

Example: Type "add2k" to add 2,000 to the cell value. Type "in10" to increase the cell value by 10%. Type "pow2" to calculate to the power of 2.



Shortcuts	
add	Add a number to the last value

Shortcuts	
sub	Subtract a number from last value
div	Divide last value of a number
mul	Multiply last value of a number
increase, in, gr	Increase last value by a percentage
decrease, de	Decrease last value by a percentage
percent, per	Calculate a percentage of last value
power, pow	Calculate a power of last value

Scaling

When entering data in the Data Explorer, use scaling to quickly enter values.

> **Example:** Type "1m" for a cell value of "1,000,000"; "1.5b" for a cell value of "1,250,000,000"

Scaling	
k	Scale value by 1,000
m	Scale value by 1,000,000

Scaling	
b	Scale value by 1,000,000,000
%	Divide by 100 (when using with a P number format on Cube Views)

HotKeys

Use HotKeys to manage changes while entering data into the Data Explorer.

> **Example:** Press CTRL+S to save changes.

HotKeys	
CTRL + S	Save

Right-Click Options

Expand/Collapse

If nested rows are used in a Cube View, right-click on any row header in order to collapse or expand the selected row content. This feature works with the RowExpansionMode property in the Cube View designer and controls how users view Cube Views in the data grid. See Rows and Columns under Cube Views in Presenting Data With Books, Cube Views and Dashboards for more details.

Calculate/Translate/Consolidate

Cube Views can be set to enable processing of the Cube View data. This includes Calculate, Translate and Consolidate. There are also option to force these calculations and log the activity in the Task Activity. Each of these options can be enabled/disabled on an individual Cube View. For more details on these options, see Launching a Consolidation in About the Financial Model.

Spreading

Spreading can be done from a Cube View while viewing it in the Data Explorer grid, in the Spreadsheet feature and Excel Add-in. User experience may differ slightly across these three user interfaces and more functions may be available in the Spreadsheet feature than others. This functionality provides the ability to spread values over selected cells.

> **NOTE:** The Spreading Dialog can be left open while entering data. If the dialog is left open while users are spreading values on the grid, it will update to the respective spreading behavior.

Spreading Type

Fill

This fills each selected data cell with the value in the Amount to Spread property.

Clear Data

This clears all data within the selected cells.

Factor

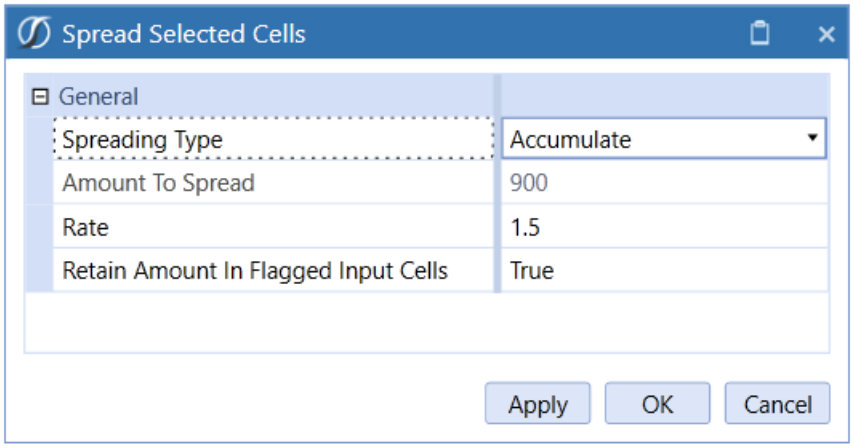
This takes the selected cell's value and multiplies it by the rate specified.

Accumulate

This takes the first selected cell's value and multiplies it by the rate specified. It then takes that value, multiplies it by the specified rate and places it in the second cell selected, and does this for all selected cells. For example, four cells are selected and the first cell has a value of 900.

Apr 2011	May 2011	Jun 2011	Jul 2011
909.00	**14.00**	**622.50**	**33.75**
900.00	0.00	0.00	0.00

The Accumulate Spreading is setup as follows with a rate of 1.5:



When the spreading is applied the outcome is as follows:

Apr 2011	May 2011	Jun 2011	Jul 2011
1,359.00	**2,039.00**	**3,660.00**	**4,590.00**
1,350.00	2,025.00	3,037.50	4,556.25

Each cell's value is a factor of the previous cell amount.

Even Distribution

This takes the Amount to Spread and distributes it evenly across the selected cells.

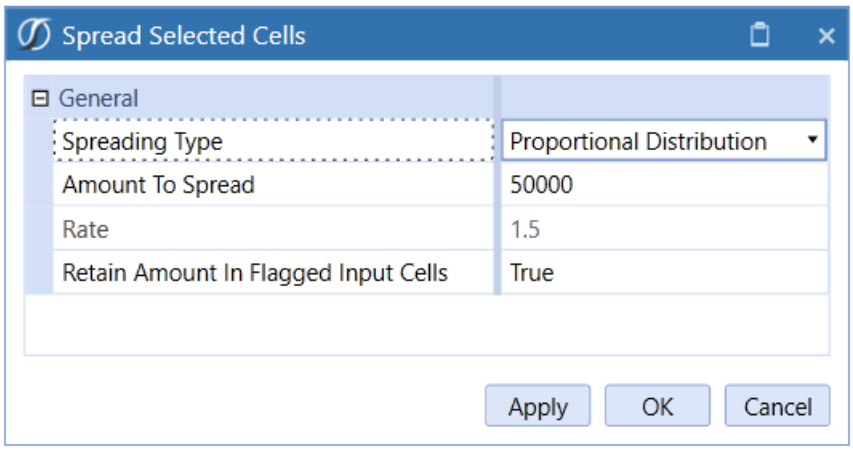
Proportional Distribution

This takes the selected cell's value, multiplies it by the specified Amount to Spread, and then divides it by the total sum of all selected cells. If all the cells have a zero value, the Amount to Spread will behave like an Even Distribution.

In the example below, four cells are selected:

Apr 2011	May 2011	Jun 2011	Jul 2011
1,359.00	**2,039.00**	**3,660.00**	**4,590.00**
1,350.00	2,025.00	3,037.50	4,556.25

A proportional amount of 50,000 is applied to the cells.



Result

Apr 2011	May 2011	Jun 2011	Jul 2011
6,162.85	**9,244.77**	**14,468.65**	**20,802.98**
6,153.85	9,230.77	13,846.15	20,769.23

445 Distribution

This takes the Amount to Spread and distributes it with a weight of 4 to the first two selected cells and a weight of 5 to the third cell.

454 Distribution

This takes the Amount to Spread and distributes it with a weight of 4 to the first selected cell, a weight of 5 to the second cell and a weight of 4 to the third.

544 Distribution

This takes the Amount to Spread and distributes it with a weight of 5 to the first selected cell and a weight of 4 to the second and third cells.

Spreading Properties

Amount to Spread

Specify the value to spread over the selected cells. The value defaults to the last cell selected. The way the amount in this field spreads varies by Spreading Type.

Rate (Factor and Accumulate Spreading Types Only)

Enter a rate to multiply by a cell value.

Retain Amount in Flagged Input Cells

Users can flag specific cells in order to retain the data within the cell. If this property is set to True, spreading will not apply to the selected flagged cells.

Include Flagged Read only Cells in Totals

Set this to True to include locked base-level cell values when calculating spreading totals. True is the default.

Flag Selected Cells

Flag cells so the original amount in the cell is retained during the spreading process.

Clear Flags

Select this to clear any flagged cells.

Apply

Enter spreading values in the dialog and select Apply to perform spreading without closing the dialog. The dialog closes upon clicking OK or Cancel.

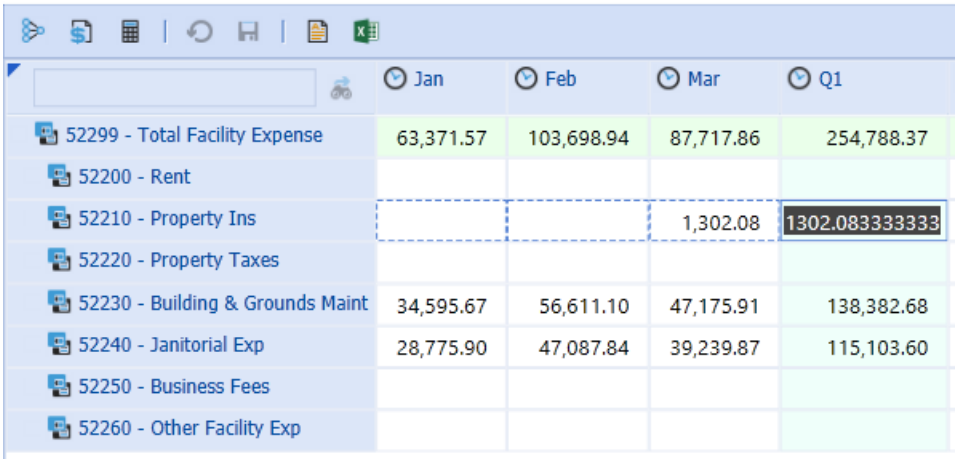
Spread Selected Cells

In order to select multiple data cells, hold the Control Key (Ctrl) on the keyboard, and click on all cells that apply. Spreading occurs across rows first and then down the columns. Once spreading has been performed using the Spreading Dialog, users can type values on specific Members without having to launch the dialog again. This spreading function applies the settings from the last time the Spreading Dialog was used, so all Spreading Types and flagged cells can be utilized. Even Distribution is the default Spreading Type if there is nothing currently set in the Spreading Dialog.

Users can select multiple Cube View data cells and type over the primary cell in order to apply spreading. If the primary selected cell is a Parent Member, a value can still be typed over it if multiple cells are selected. Any locked base-level data cells are automatically flagged when applying spreading.

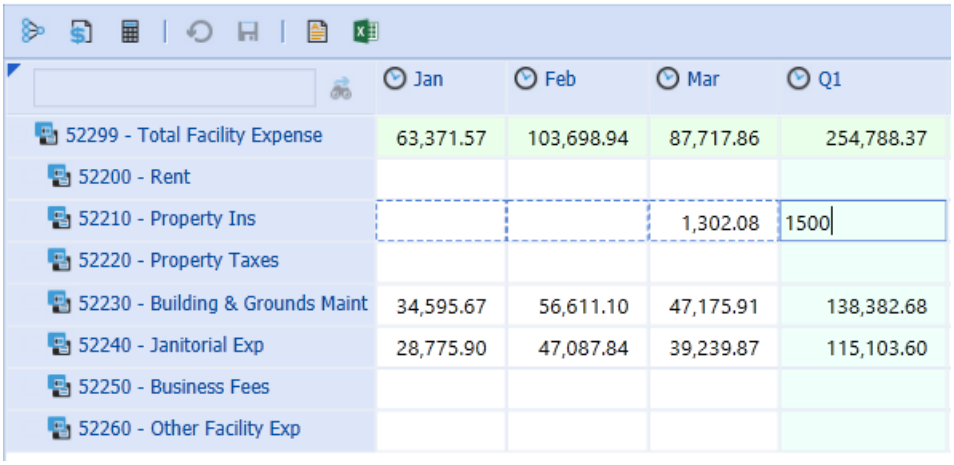
When spreading across multiple time periods on one data row, users can select the Parent Time Member and automatically apply spreading to all its Base Time Members without having to manually select them.

For example, double-click on the Q1 Parent Member and the periods associated with Q1 will be selected.



	Jan	Feb	Mar	Q1
52299 - Total Facility Expense	63,371.57	103,698.94	87,717.86	254,788.37
52200 - Rent				
52210 - Property Ins			1,302.08	1302.083333333
52220 - Property Taxes				
52230 - Building & Grounds Maint	34,595.67	56,611.10	47,175.91	138,382.68
52240 - Janitorial Exp	28,775.90	47,087.84	39,239.87	115,103.60
52250 - Business Fees				
52260 - Other Facility Exp				

Enter a value to spread on the Q1 data cell.



	Jan	Feb	Mar	Q1
52299 - Total Facility Expense	63,371.57	103,698.94	87,717.86	254,788.37
52200 - Rent				
52210 - Property Ins			1,302.08	1500
52220 - Property Taxes				
52230 - Building & Grounds Maint	34,595.67	56,611.10	47,175.91	138,382.68
52240 - Janitorial Exp	28,775.90	47,087.84	39,239.87	115,103.60
52250 - Business Fees				
52260 - Other Facility Exp				

Click <Enter> and it will automatically apply to the periods associated with the Quarter.

	Jan	Feb	Mar	Q1
52299 - Total Facility Expense	63,371.57	103,698.94	87,717.86	254,788.37
52200 - Rent				
52210 - Property Ins	500.00	500.00	500.00	1,302.08
52220 - Property Taxes				
52230 - Building & Grounds Maint	34,595.67	56,611.10	47,175.91	138,382.68
52240 - Janitorial Exp	28,775.90	47,087.84	39,239.87	115,103.60
52250 - Business Fees				
52260 - Other Facility Exp				

Allocation

Form data Allocations can be completed on a Cube View in the Data Explorer Grid, a Cube View in Excel, and Quick Views.

Allocation Type

Clear Data

This clears all Form data for the specified Destination POV.

> **NOTE:** Users can clear data for Dimension Members that are not displayed on the Form.

For property definitions, see Even Distribution.

Even Distribution

This provides an even distribution across the Destination Members.

Source POV

The Source POV determines the source intersection and applies the value from this intersection to the allocation. The Source POV defaults to the last cell selected. Users can also select a data cell and drag and drop the cell's POV into this property in order to update it. Any Members not included in the Source POV will default to the user's Cube POV.

Source Amount or Calculation Script

Use this property to override the Source POV value. Drag and drop a value from the Cube View grid, specify a source amount (e.g., 1000), or use the Source POV to specify a different source amount (e.g., A#SourcePOV*0.90).

Destination POV

The Destination POV determines the intersection where the allocation will take place. A Member for each Dimension must be specified for the allocation to take place in the correct intersection. Users can select a data cell and drag and drop a POV into this field in order to display all Dimensions. If the Destination POV is similar to the Source POV, specify as many Dimensions as necessary and the remaining Members will come from the Source POV, or leave this field blank and all POV Members will be based on the Source POV. Any Members not included in the Source or Destination POV will default to the user's Cube POV.

Dimension Type/Dimension Type 2

Allocations can be applied to Dimension Members not included in the Destination POV such as several periods or Accounts. These Members then override the Destination POV. Specify the Dimension in this property. For the Clear Data and Advanced Allocation Types, two Dimensions can be specified.

Member Filter/Member Filter 2

Specify the Dimension Members to which the allocation will apply. The Members specified in this field override the Member in the Destination POV script. For the Clear Data and Advanced Allocation Types, two Dimension Member Filters can be specified.

Save Zeroes as Not Data

Set this to True in order to suppress zeroes when saving allocation data. The default setting is True.

445 Distribution

A 445 Distribution takes the source amount and applies a weight of 4 to the first two specified destination intersections and then a weight of 5 to the third intersection. This applies the allocation across rows first and then moves down the column.

454 Distribution

A 454 Allocation takes the source amount and applies a weight of 4 to the first destination intersection, a weight of 5 to the second and then a weight of 4 to the third intersection. This applies the allocation across rows first and then moves down the column.

544 Distribution

A 544 Allocation takes the source amount and applies a weight of 5 to the first destination intersection and then a weight of 4 to the second and third intersections. This applies the allocation across rows first and then moves down the column.

For Source and Destination property definitions, see Even Distribution above.

Weighted Distribution

This applies a weighted value to each specified Destination Member. The weights are determined in a Weight Calculation Script which uses the specified Dimension intersections' cell values.

For Source and Destination property definitions, see Even Distribution above.

Weight Calculation Script

The weight calculation script calculates the value by determining

```
|SourceAmount| * (|Weight| / |TotalWeight|)
```

It then applies the weighted values to the intersections specified in the Destination POV/Member Filter combination.

These are system Substitution Variables and are determined by the following:

|Weight|

This is the weight value applied to each Member in the Destination POV.

|TotalWeight|

This multiplies the weight by number of allocated intersections. For example, if there is a weight of 5 being allocated across 12 intersections, the |TotalWeight| would be 60.

Identify specific Dimension Members separated by a colon in order to determine the |Weight| and |TotalWeight|. Any Dimensions not specified in this field will come from the Destination POV.

Users can drag and drop a POV from a data cell in order to jumpstart the calculation script. In order to apply the Members specified in the Member Filter property, delete that particular Dimension in the script.

Example

Destination POV

Cb#Houston:E#[Houston Heights]:C#USD:S#Actual:T#2011M5:V#Periodic:A#56000:F#None:O#Forms:I#None:U1#None:U2#None:U3#None:U4#None:U5#[IFRS Adj]:U6#None:U7#None:U8#None

Member Filter

T#2011M7,T#2011M8,T#2011M9

Weight Calculation Script Example 1

A#26000:O#Top

The remaining Dimensions are determined by the Destination POV and Member Filter.

Weight Calculation Script Example 2

```
Cb#Houston:E#[Houston Heights]:C#USD:S#Actual:V#Periodic:A#56000:F#None:O#Forms:
I#None:U1#None:U2#None:U3#None:U4#None:U5#[IFRS Adj]:U6#None:U7#None:U8#None
```

The Time Dimension was removed from the script in order to use the three Time Members in the Member Filter.

In the example above, the Weight Calculation Script is identifying three intersections of data. OneStream uses the sum of those three intersections as the |TotalWeight| and each individual intersection as the |Weight|. This determines how to spread the Source Amount amongst the Destination Members.

Enter a value in order to create an even distribution to all specified Destination Members.

Advanced

Advanced Allocations are similar to Weighted Distributions, but allow users to override two destination Dimensions, control how the weights are calculated, and offset amounts for the Source and Destination Members. See Form Allocations in Data Collection Guides for an Advanced Allocation example.

For Source and Destination property definitions, see Even Distribution above.

For Weight Calculation Script, see Weighted Distribution.

Destination Calculation Script

This determines how to calculate the Weight Calculation Script. The default calculation is |SourceAmount| * (|Weight|/|TotalWeight|).

Example:

```
(|SourceAmount| * (|Weight|/|TotalWeight|)) *1.5
```

This will calculate the weighted value for each specified intersection, multiply it by 1.5 and apply that value to the destination data cell.

Translate Destination if Different Currency

Set this to True in order to translate the destination currency if it differs from the currency on the Form.

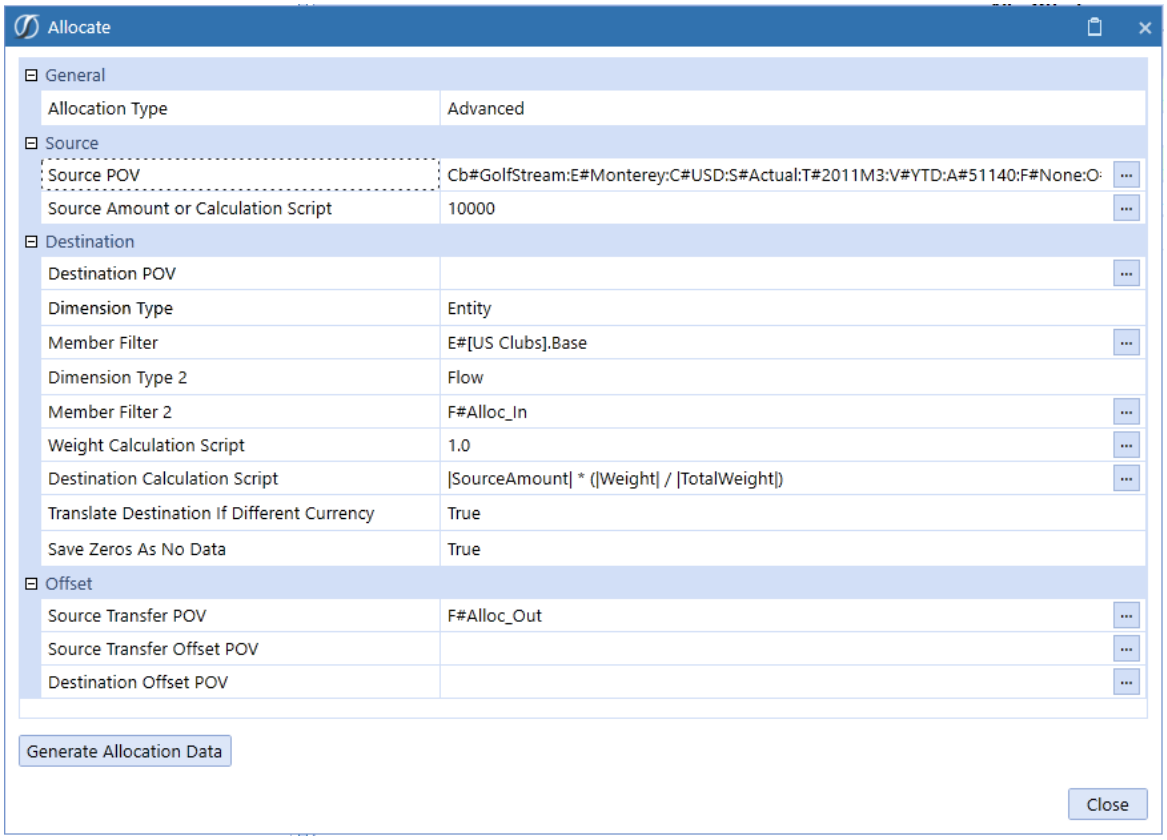
Offset

Specify specific Members to offset when doing an allocation. Transferring values out of the Source POV requires an entry in the Source Transfer POV while entries in Source Transfer Offset POV and Destination Offset POV will ensure that entries are balanced by updating a related Account.

		Monterey	Augusta	Carlsbad	Houston Heights	South Houston
51140 - IT Services Expense	TopFlow	0.00	2,500.00	2,500.00	2,500.00	2,500.00
	None	10,000.00				
	Allocations	-10,000.00	2,500.00	2,500.00	2,500.00	2,500.00
	Alloc_In		2,500.00	2,500.00	2,500.00	2,500.00
	Alloc_Out	-10,000.00				

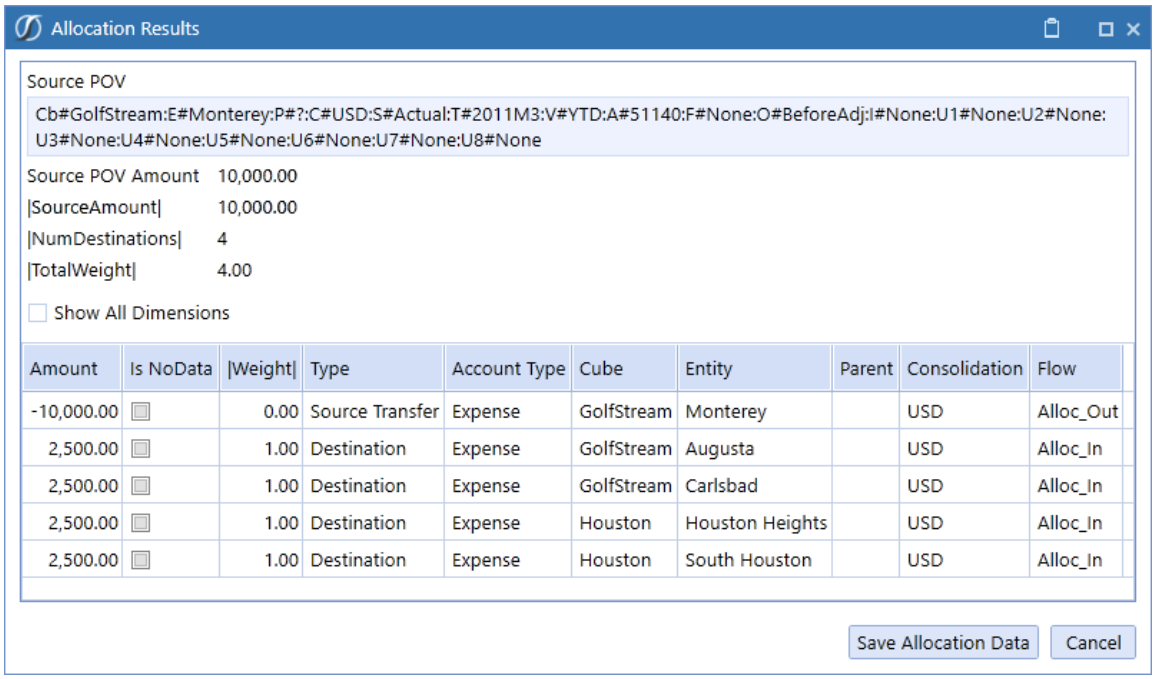
In the example above, Monterey is transferring their IT Services Expense to four other Entities.

If a balanced entry is not required, the method below can be used by configuring the Destination and Source Transfer POV.



Source Transfer POV

This is an optional field. Providing a Member Filter here will result in a transferring entry which will zero out the Source Amount. Using allocation members in the Flow Dimension in the example above, Monterey's original value is reduced to zero after netting F#None with F#Alloc_Out, the Source Transfer POV. Note that the allocated values are targeted to the Alloc_In Member in the Destination.

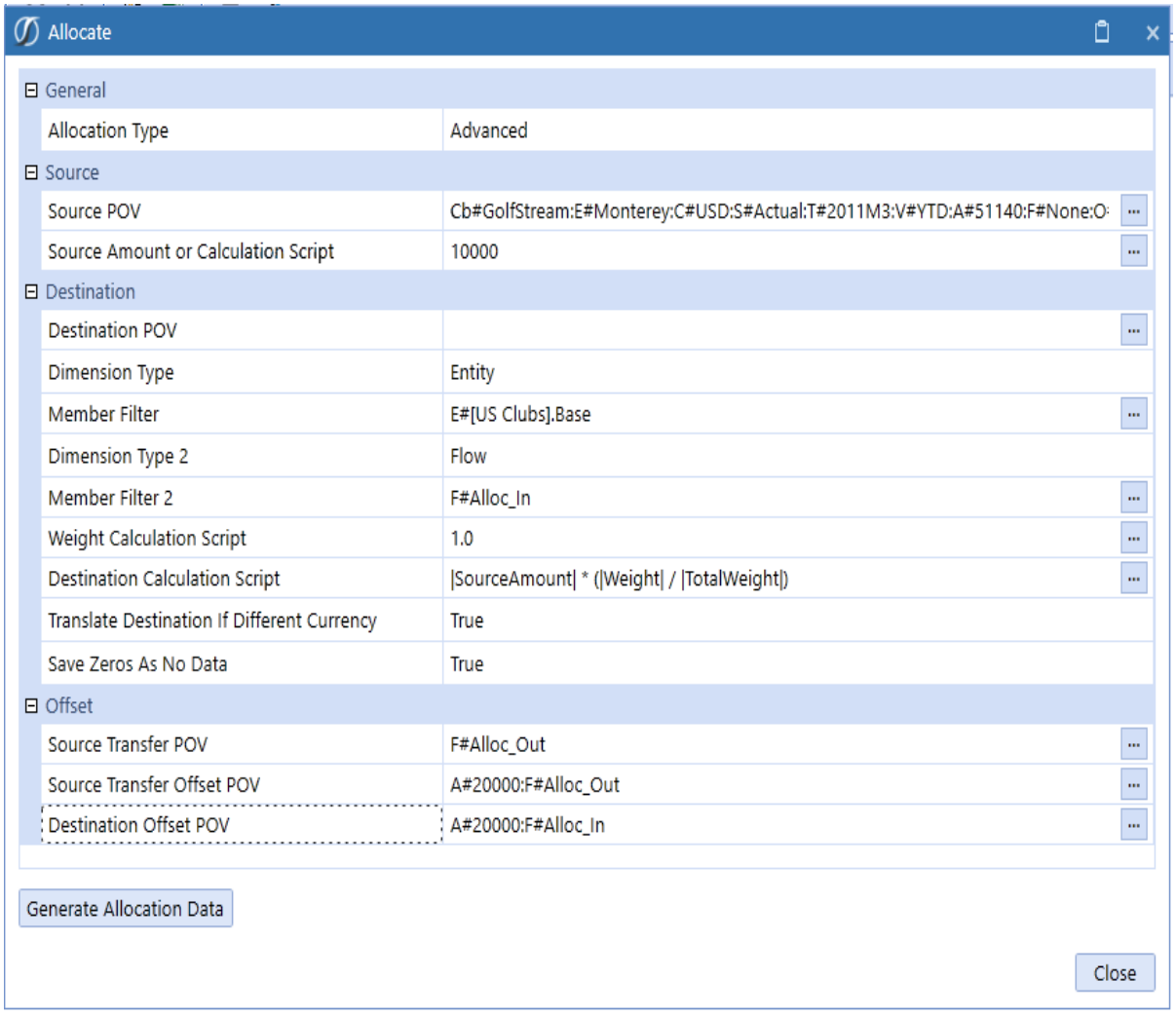


Source Transfer Offset POV

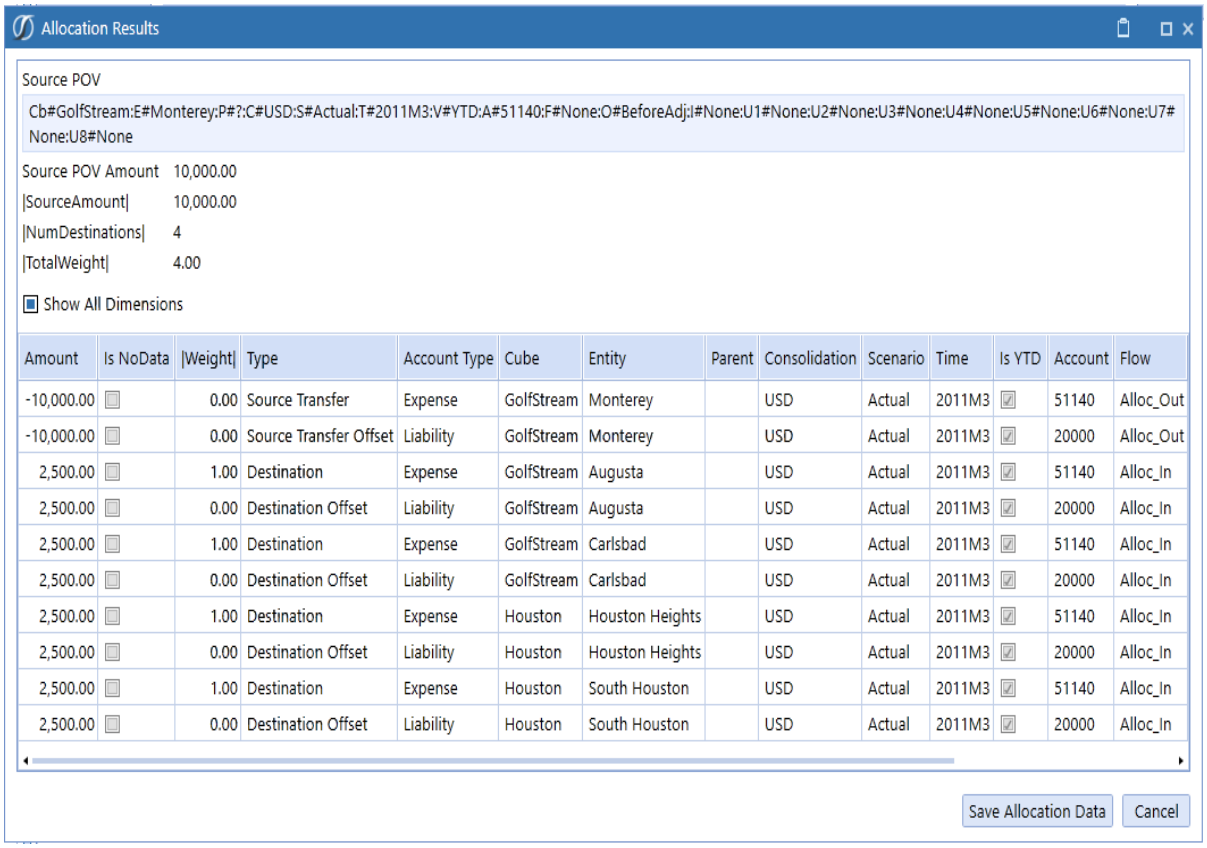
This is an optional field but required to create a balancing entry usually in a different Account in the Source. In the example below, the intersection is a different Account and results in a different intersection from the Source POV being updated to offset the Source Transfer POV update that occurs with the allocation. In the example below, an expense in the Source POV is being reduced so the offset is to a payable (Liability) Account.

Destination Offset POV

This is an optional field but required to create a balancing entry usually in a different Account in the Destination. In the example below, the intersection is a different Account and results in a different intersection from the Destination POV or Member Filter being updated to offset the allocated value updates in the Destination. In the example below, an expense in the Destination is being increased so the offset is to a payable (Liability) Account. This could have been offset to an equity clearing Account instead, as this is typical.



This results in these entries:



Amount	Is NoData	|Weight|	Type	Account Type	Cube	Entity	Parent	Consolidation	Scenario	Time	Is YTD	Account	Flow
-10,000.00	☐	0.00	Source Transfer	Expense	GolfStream	Monterey		USD	Actual	2011M3	☑	51140	Alloc_Out
-10,000.00	☐	0.00	Source Transfer Offset	Liability	GolfStream	Monterey		USD	Actual	2011M3	☑	20000	Alloc_Out
2,500.00	☐	1.00	Destination	Expense	GolfStream	Augusta		USD	Actual	2011M3	☑	51140	Alloc_In
2,500.00	☐	0.00	Destination Offset	Liability	GolfStream	Augusta		USD	Actual	2011M3	☑	20000	Alloc_In
2,500.00	☐	1.00	Destination	Expense	GolfStream	Carlsbad		USD	Actual	2011M3	☑	51140	Alloc_In
2,500.00	☐	0.00	Destination Offset	Liability	GolfStream	Carlsbad		USD	Actual	2011M3	☑	20000	Alloc_In
2,500.00	☐	1.00	Destination	Expense	Houston	Houston Heights		USD	Actual	2011M3	☑	51140	Alloc_In
2,500.00	☐	0.00	Destination Offset	Liability	Houston	Houston Heights		USD	Actual	2011M3	☑	20000	Alloc_In
2,500.00	☐	1.00	Destination	Expense	Houston	South Houston		USD	Actual	2011M3	☑	51140	Alloc_In
2,500.00	☐	0.00	Destination Offset	Liability	Houston	South Houston		USD	Actual	2011M3	☑	20000	Alloc_In

NOTE: Both the expense and liability have been removed from Monterey and transferred to the other four Entities:

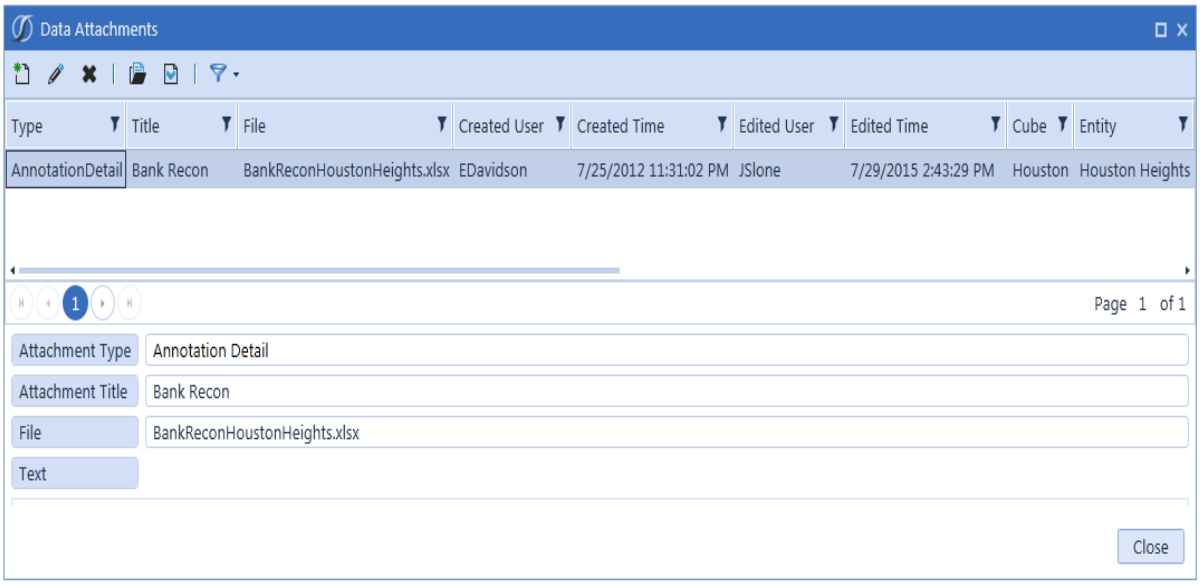
		AllocOffset				
		Monterey	Augusta	Carlsbad	Houston Heights	South Houston
51140 - IT Services Expense	TopFlow	0.00	2,500.00	2,500.00	2,500.00	2,500.00
	None	10,000.00				
	Ending Balance					
	Allocations	-10,000.00	2,500.00	2,500.00	2,500.00	2,500.00
	Alloc_In		2,500.00	2,500.00	2,500.00	2,500.00
	Alloc_Out	-10,000.00				
20000 - Trade Accounts Payable	TopFlow	0.00	3,817,398.83	2,946,851.20	6,787,789.56	10,180,434.35
	None	10,000.00	3,814,898.83	2,944,351.20	6,785,289.56	10,177,934.35
	Ending Balance					
	Allocations	-10,000.00	2,500.00	2,500.00	2,500.00	2,500.00
	Alloc_In		2,500.00	2,500.00	2,500.00	2,500.00
	Alloc_Out	-10,000.00				

Data Attachments for Selected Cell/Data Unit

Data Attachments can be added at the cell level or for the entire Data Unit and can be set up to show a red tick mark on cells that contain an attachment. These can be a textual comment, file attachment or both and with different types of attachments: Standard, Annotation, Assumptions, Audit Comment, Footnote or Variance Explanation. There can be many Standard attachments per cell or Data Unit, but only one of the other types. The other types of Data Attachments can be viewed in a row or column in a Cube View by selecting their related View Dimension Member. Attachment Types are part of the View Dimension which makes them available for a variety of reporting and explanation needed within the information delivery process. This allows them to be included within Cube View results.

Data Attachments for Selected Cell/Data Attachments for Selected Data Unit

Any cell in any grid can contain a data attachment. To attach a file, right click on any cell in a data grid and select Data Attachment for Selected Cell. Click the upper left Create Data Attachment icon and select an Attachment Type from the drop-down menu, give it a title, select a file to attach or simply type in text, and click OK. The attachment will now appear in the Data Attachments box.



> **TIP:** This attachment can be viewed from any data grid that employs a Workflow Unit, Scenario, and Period. To view all data attachments for any Workflow Unit, Scenario and Period, right click on any cell and select Data Attachments For Selected Data Units. This ensures that all data attachments become valuable analysis tools. This is part of the View Dimension because each is given a Type and the data attachments can be reported on as a whole and included in other types of reports.

Spell Checking in Data Attachments

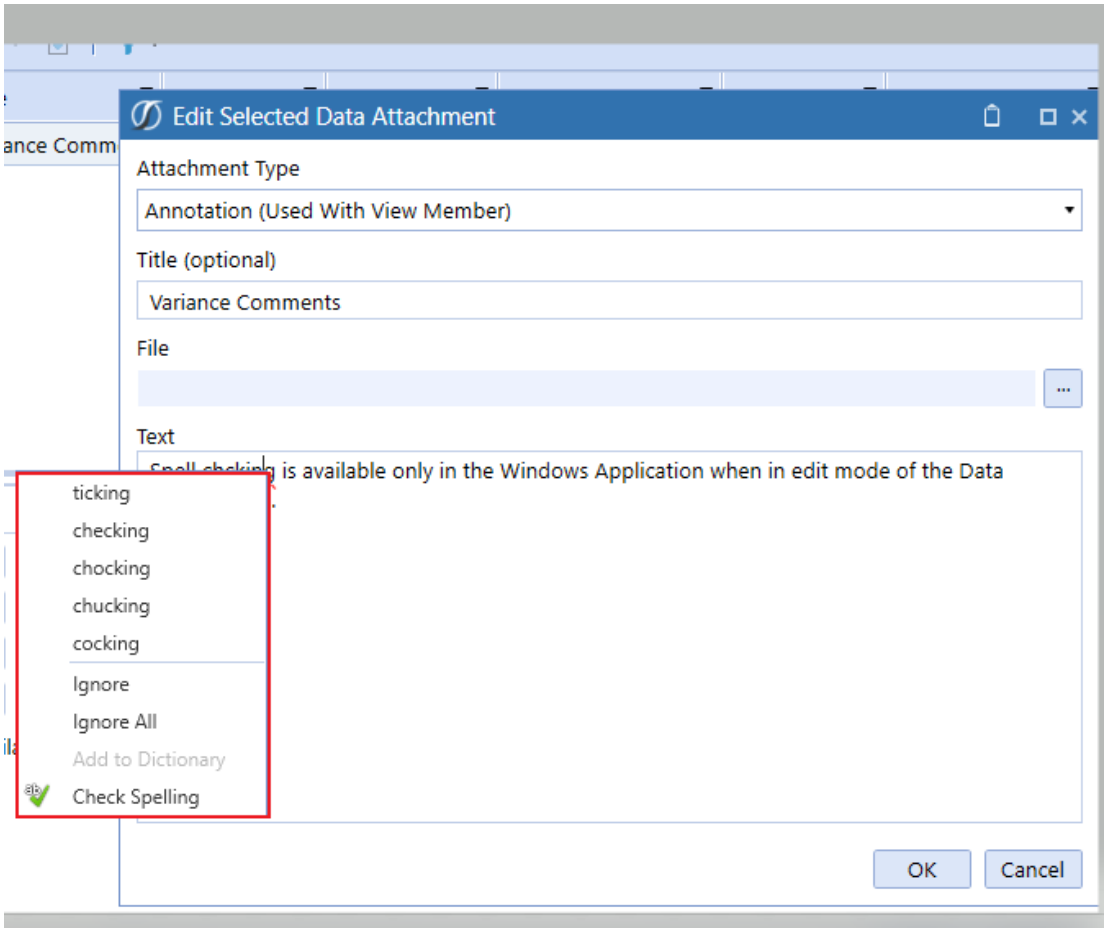
The text spell check feature is available only when using the Windows Application in a right-click menu. This feature is inactive by default. The default is set to Not Used. In Text Editor, the Review ribbon will allow the user to activate Spell Check using the Spell Check button.

The Spell Check feature is enabled for the English language only. Spell Check will only work when English is selected in Windows for International users. The culture is determined by each user's individual culture setting assigned in OneStream User Security. The culture is assigned to the OneStream application on the Application Server Configuration Utility as "en-US".

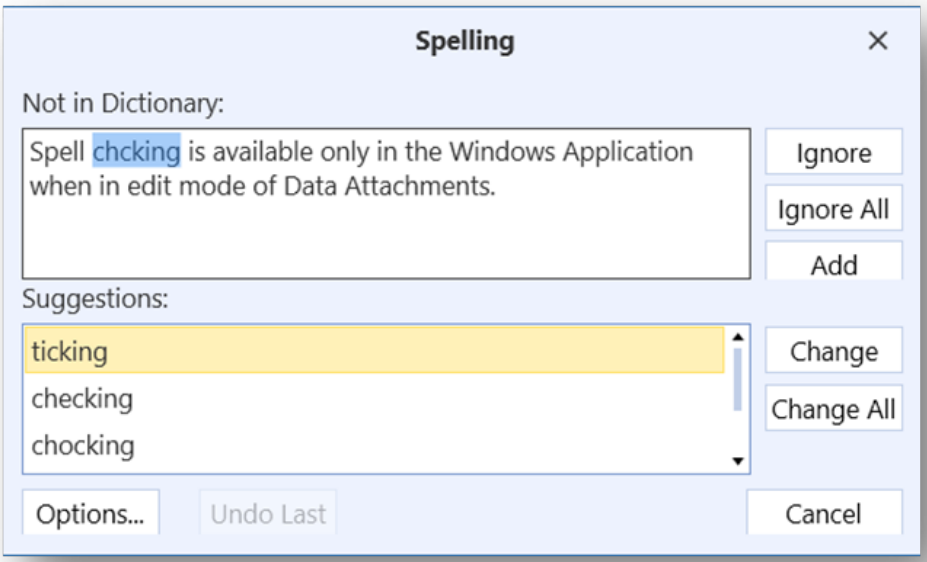
Spell Check in Data Attachments

Spell Check options will be available when creating or editing Data Attachments. Once enabled, the spell check feature will actively check spelling in Data Attachments. The Spell Check will be applied to all Data Attachment dialogs once turned on.

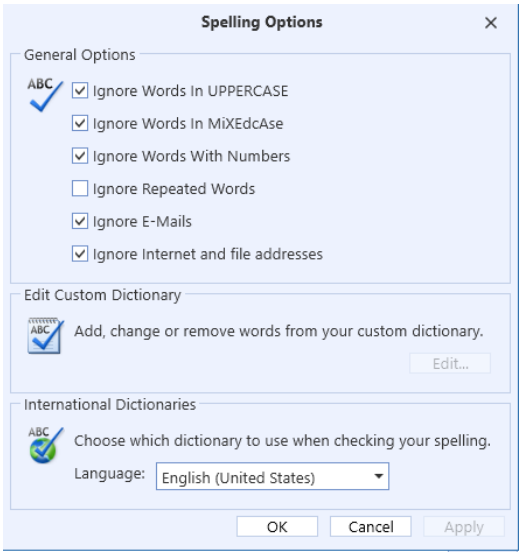
The user must right-click on a word to activate the Spell Check options where suggested solutions are presented. Choosing the "Ignore" option will only be retained for the current session. Closing and re-opening to the edit mode will re-check any previously ignored items.



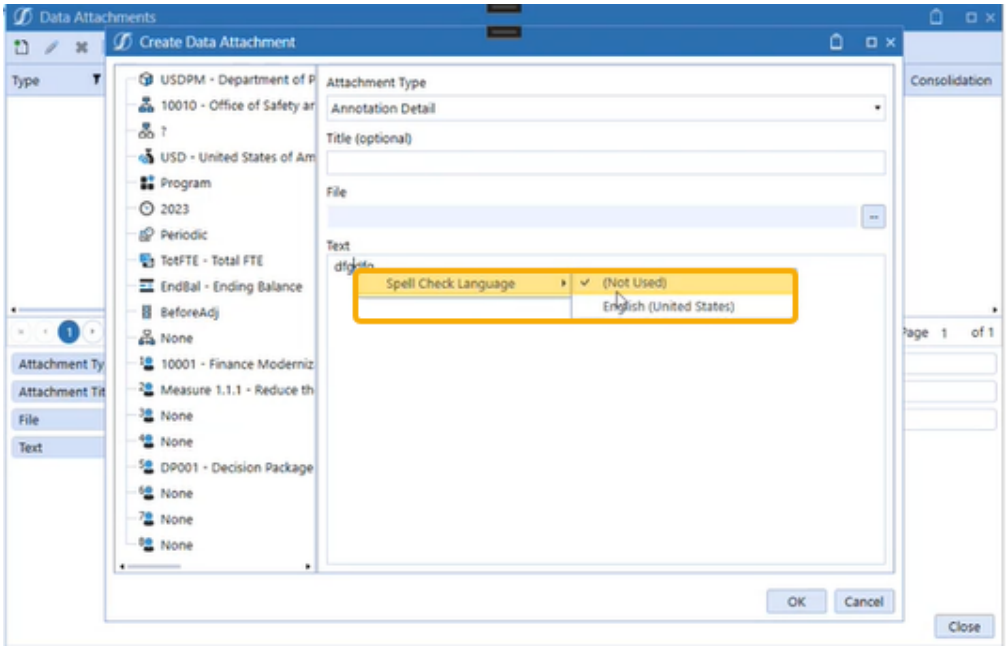
Choosing Check Spelling will check the spelling of words within the text box content from any starting point.



The Options button will allow the user to modify the Spell Check behavior within the current task session. These settings are never saved as a user preference.

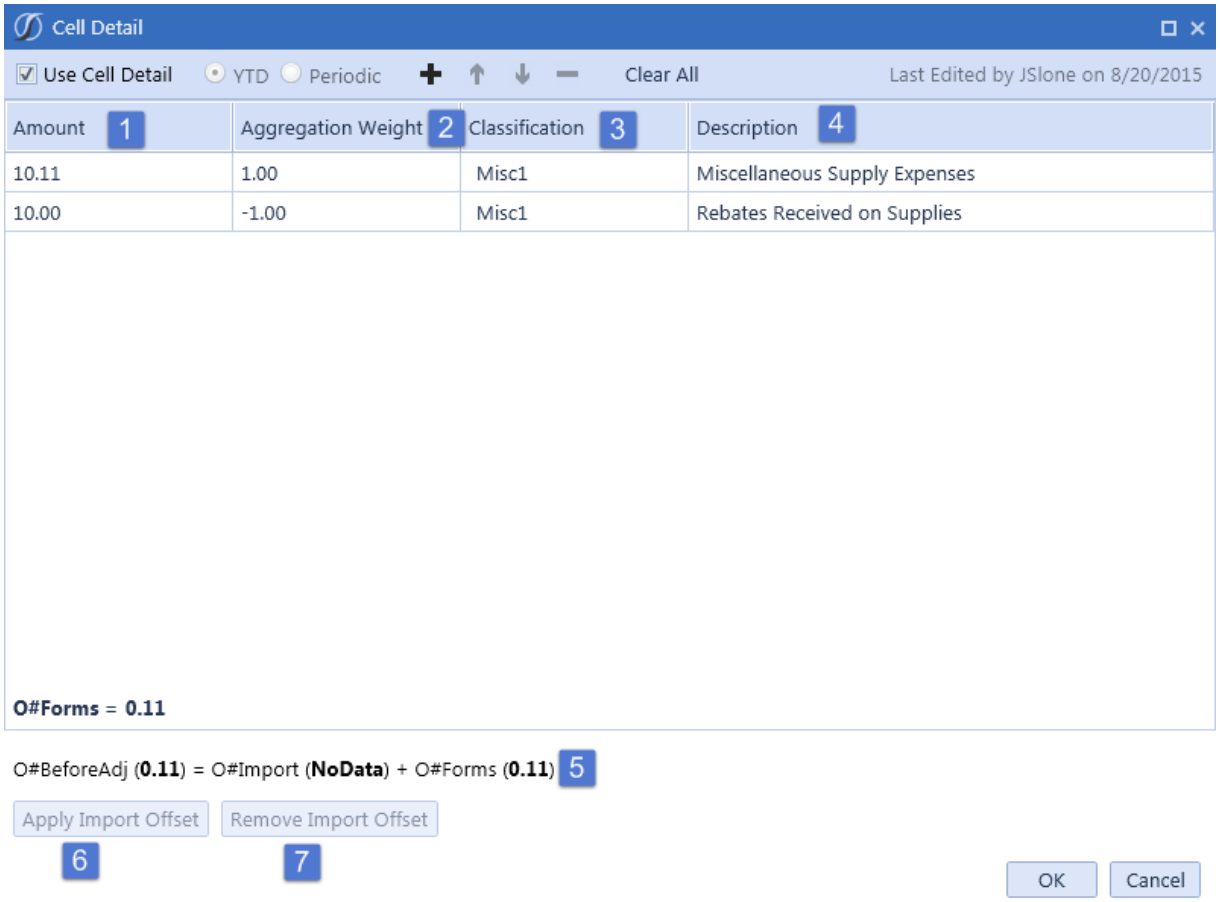


Users can also right-click on a word to turn English Spell Check on or off. The Spell Check Language will be set to (Not Used) by default in the Data Attachments text boxes.



Cell Detail

Cell Detail can be entered on a Cube View used for a data entry form, or on a Cube View or Quick View in Excel or Spreadsheet. Cell Detail can also be loaded via an Excel or CSV template. See Loading Cell Detail in Data Collection Guides for more details on creating these templates. Cell Detail is available on any writeable O#Forms or O#BeforeAdj Member. In order to disable this by account or any other intersection, use a Conditional Input Business Rule. See Finance Business Rules in Application Tools for an example of this rule.



Cell Detail Toolbar

Use Cell Detail

Check this box in order to enable Cell Detail.

YTD/Periodic

This determines how the data is being entered in the Form. This only applies to Income Statement Accounts. If a YTD Form is being used and a Periodic line item is entered, the Form will calculate the YTD value and store it accordingly.

These icons add, delete, and move Cell Detail records in the grid.

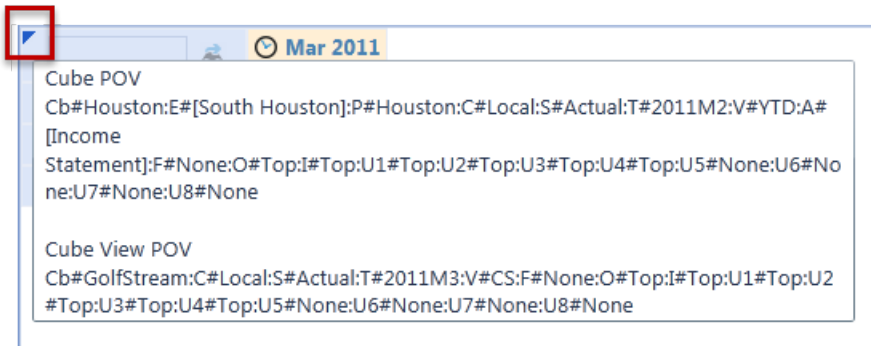
Clear All

This removes all Cell Detail records in the grid but does not remove any stored numerical data from the Forms Member in the Cube.

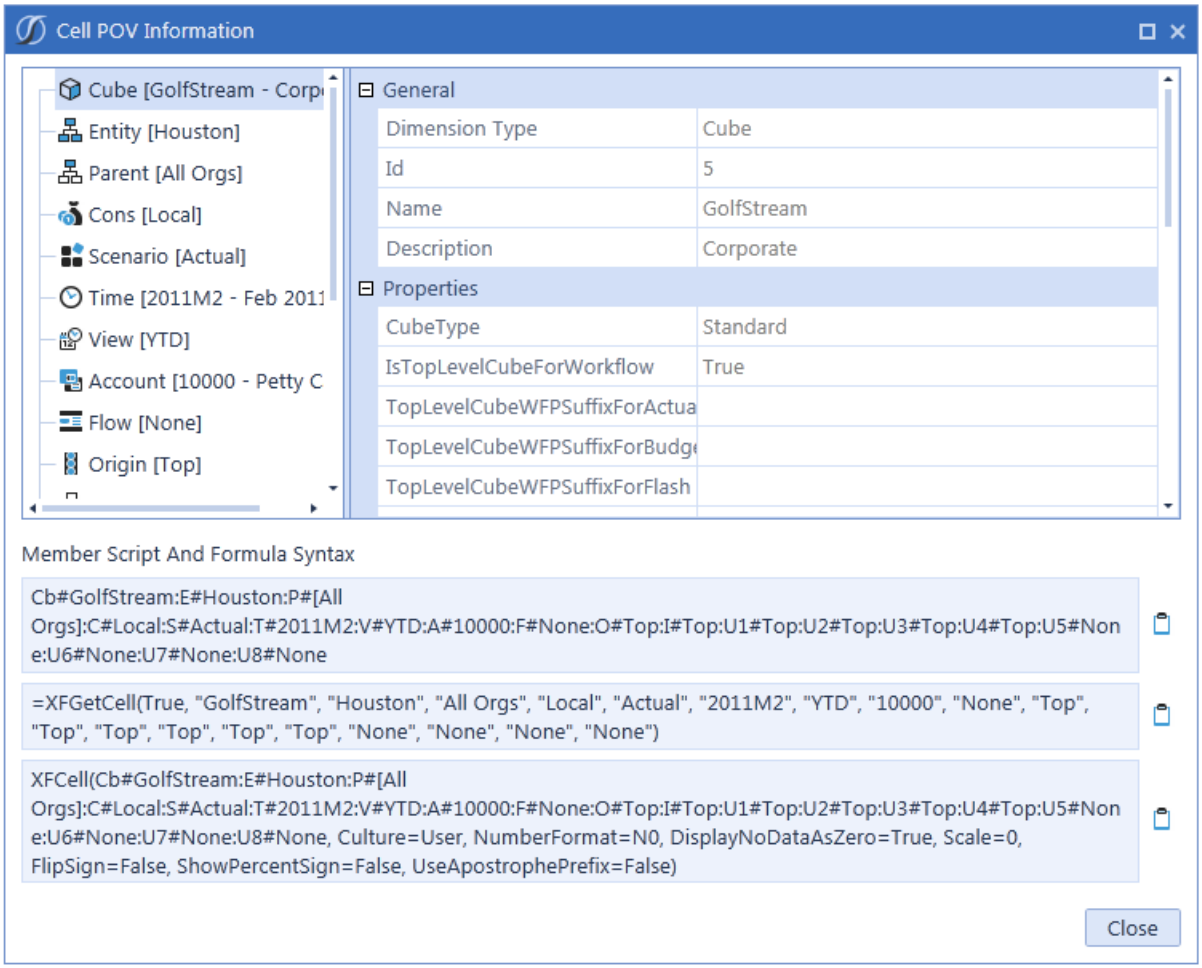
1. **Amount**
   Enter the amount for the Cell Detail Record or Records.
2. **Aggregation Weight**
   Enter an aggregation weight to calculate a cell item using simple multiplication. For example, entering -1 will reverse the value in the amount column.
3. **Classification**
   The user can select a Classification from a drop list on each line item, where the list of Classifications is defined using a Dashboard Parameter specifically named as CellDetailClassifications. Once this Dashboard Parameter is created, Cell Detail will recognize the classifications without having to assign the Parameter to any Cube View or Form.
   Note: Users can add additional value items to a Parameter, or change existing items, however, this will not change any classification assigned and stored to a line item in a form.
4. **Description**
   Enter text information about the Cell Detail.
5. This displays the values that are going to be stored in the Origin Members based on what was stored in the Cell Detail form.
6. **Apply Import Offset**
   This is used mainly for Budgeting purposes. This applies the reverse amount of what is in the Import Origin Member allowing users to enter the total amount for the cell intersection without having to determine what has already been imported.
7. **Remove Import Offset**
   This removes an Import Offset Adjustment previously made via the Cell Detail form.

Cell POV Information

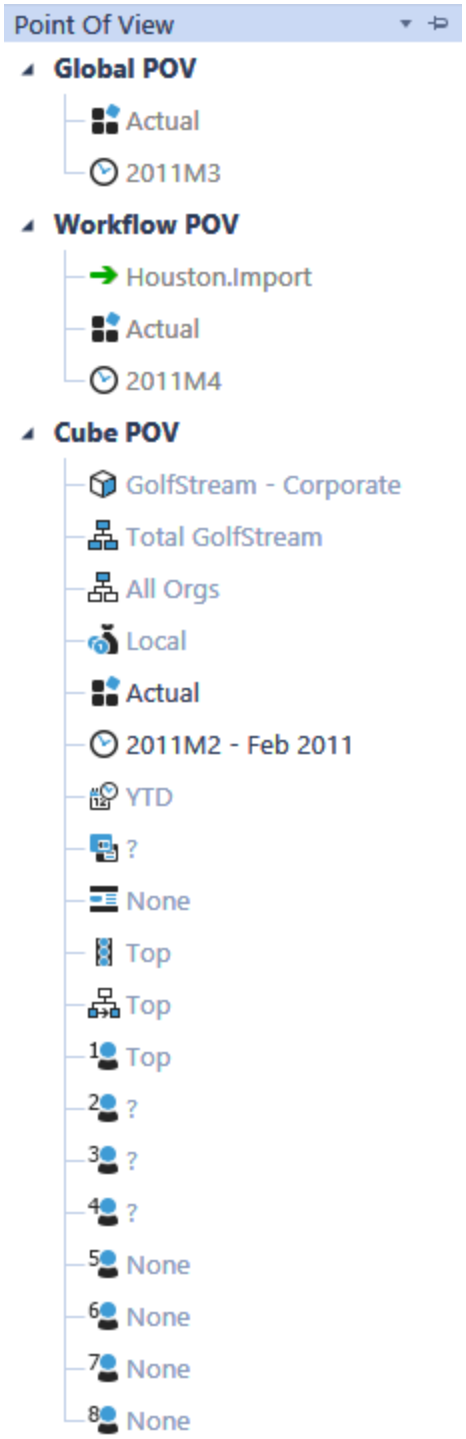
By hovering over the tick mark on a previewed Cube View, defaulted Members from this Cube View's Point of View and the Cube's Point of View settings will display in the form of a tool tip.



Right click on any cell and select Cell POV Information in order to see a detailed summary of the selected Members related to this intersection. All the major properties of these Members can be seen from this dialog. The full Member Script and formula syntax to get to this value is also displayed. Users can use the clipboards to copy and paste the retrieve functions. The XFGetCell formula can be copied in this dialog and used in an Excel file to retrieve this specific data in Excel. The XFCell formula can be copied in this dialog and used in a text file, such as Word or PowerPoint, to retrieve this specific data in an Extensible Document. See Extensible Document Framework in Presenting Data With Extensible Documents for more details on this retrieve function.



Interact with the Point of View in the Context Pane for the Cube View's bolded Dimensions. When these Dimensions are changed, the Cube View's results will change. In the example below the Cube View allows the Scenario and Time to change.

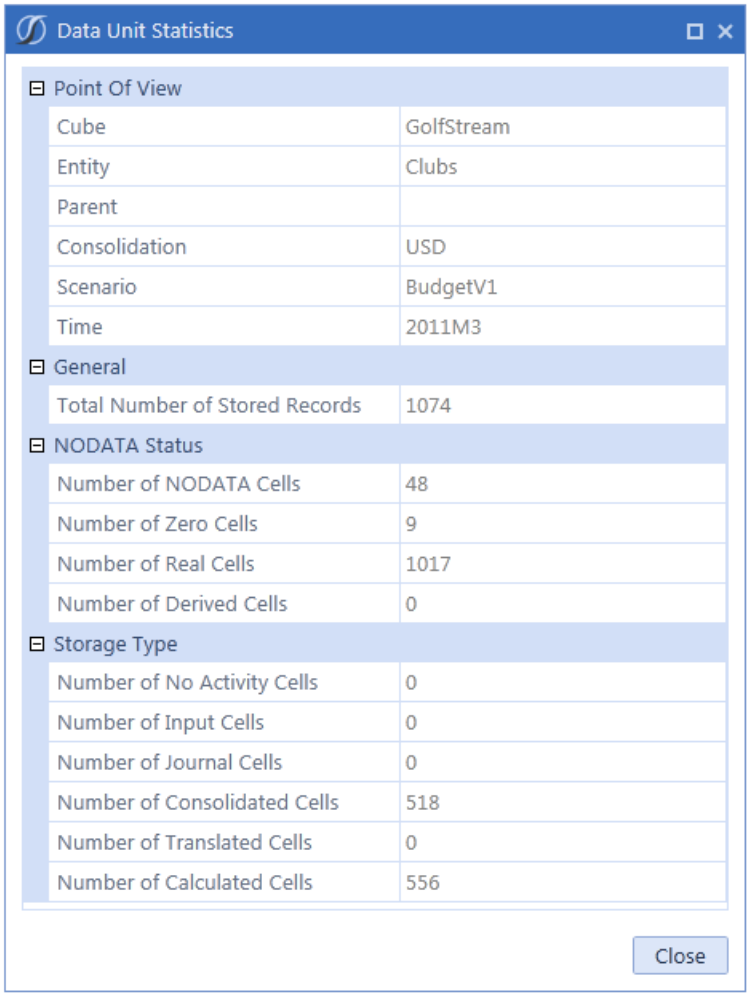

Point Of View
Global POV
Actual
2011M3
Workflow POV
Houston.Import
Actual
2011M4
Cube POV
GolfStream - Corporate
Total GolfStream
All Orgs
Local
Actual
2011M2 - Feb 2011
YTD
?
None
Top
Top
Top
?
?
?
None
None
None
None

Cell Status

Right click and select Cell Status in order to view status properties and Dimension information about a specific cell.

Data Unit Statistics

Right click and select Data Unit Statistics in order to see details on the selected cell's Data Unit. (Number of: zero cells, real cells, derived cells, NODATA cells, calculated cells, consolidated cells, translated cells, journal cells, input cells, etc.)



Drill Down

When choosing Drill Down in the right click menu item in a Cube View or Quick View another tab will open with the Drill Down results. Drill down works the same whether a user is drilling from the data explorer grid in OneStream, from the Excel Add-In or OneStream Spreadsheet. An administrator might want to know what makes up Net Income for all Entities in Europe across all products groups, and with the Drill Down option this can be accomplished.

> **TIP:** Drill into any cell on a data grid or in Excel; it does NOT need to be a base level number.

The resulting screen shows the drilled back intersection in the Drill Down History section. The white cells show base amounts, meaning drilling cannot go further. The green cells can continue to be drilled.

> **NOTE:** Drilldown is available on XFGetCell using the Excel Add-In. Right-click or use the **Drilldown** option from the **Analysis** tab.

Show Data Unit Dimensions

When selected, the upper grid shows the Members for Cube, Entity, Parent, Consolidation, Scenario, Time and View.

Download Results

These icons allow a download of the drill results into either a CSV or Excel file. When downloading drill results into a CSV file from the Drill Down page, the list separator in the CSV file is determined by the culture set on your user profile. If the culture on your user profile matches that of the operating system's regional settings, the file opens correctly in Excel without first having to save the file and then importing it into Excel.

Drill Down History

Every time an administrator drills further down a bread crumb trail of the last action is visible in the Selected Action column. This explains what happened in the drill down process to get to the most recent drill results on the bottom section of the screen. The user can always go back to this area and drill a different way into the data.

The following options become available when a user right-clicks on any field. Not all options will be available for every cell.

Entity Children Contribution (Entity Only)

This shows how each Entity contributed to the Parent.

Local Currency (Consolidation Only)

This will show the local currency Entities.

Member Children and Base

This provides two ways to drill further into a green cell in the top section. The first will drill just into the children of the Member while the other will drill down to the white base cells.

All Aggregated Data Below Cell

This drills to white cells in each Dimension.

All Stored Data in Data Unit

This drills into all of the base stored data within the full Data Unit (Cube, Entity, Parent, Consolidation, Scenario, Time and View), so this is a much broader result set than the initially drilled data.

Copy POV from Data Cell

Select this to copy the Point of View for the selected cell.

Create Quick View Using POV from Selected Cell

Select this to create a Quick View based on the POV from the row selected from either the Drill Down History or the Results for Most Recent Drill Down pane. Individual data cells in the same row will result in the same Quick View being created.

Cell POV Information

Select this to see the full Point of View for the cell.

Cell Status

Select this to see information about the cell such as if the Members have children, if it is calculated, the lock status, etc.

Calculation Inputs

This gives details on the formula source accounts for a specific account.

Load Results for Imported Cell

This will drill back to the source data for the imported cell.

Audit History for Forms or Adjustment Cell

This will drill back to the source data for a Form or Journal.

> **TIP:** OneStream keeps a bread crumb trail of all drill actions in the right side of the top drill screen under the heading Selected Action. Users will never get lost in the data as they can always start over from the top and begin a drill again.

Origin Audit Drill Down

Perform an Origin Audit Drill in order to see where data was delivered into the system. Right click on the Origin cell, typically Top and select Origin Base. This will reveal all the data that came from each Origin Member. This is now at the Origin Base and any of the following drill down processes can be performed.

Import Drill

Right click on the Origin Member Import and select All Aggregated Data. Right click again on the Origin Channel and click Load Results for Imported Cell and then select Navigate to Source Data to get down to the GL Account level. Right click on any cell in the source system data line to get down to the source document that created that line item.

Forms Drill

Right click on the Origin Channel Forms and select All Aggregated Data Below Cell. Right click again on the Origin Channel and click Audit History for Forms or Adjustment Cell. This is the specific line item that created the drilled-on value.

> **TIP:** To see each line item from the Form used to create the drilled-on value, click the button View All Submitted Data Cells. This is now showing all the line items from the Form used to create the actively drilled line item.

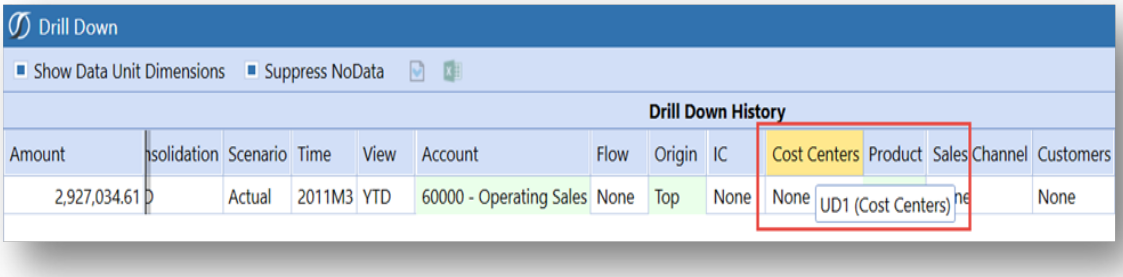
AdjInput (Journals) Drill

Right click on the Origin Channel AdjInput and select All Aggregated Data Below Cell. Right click again on the Origin Channel AdjInput and click Audit History for Forms or Adjustment Cell. This is the specific line item that created the drilled-on value. This view shows all Journal entries affecting this cell including previous Journals, deleted Journals and the current Journal.

> **TIP:** To see each line item from the Journal used to create the drilled-on value, click the button View All Submitted Data Cells. This is now showing all the line items from the Journal used to create the actively drilled line item.

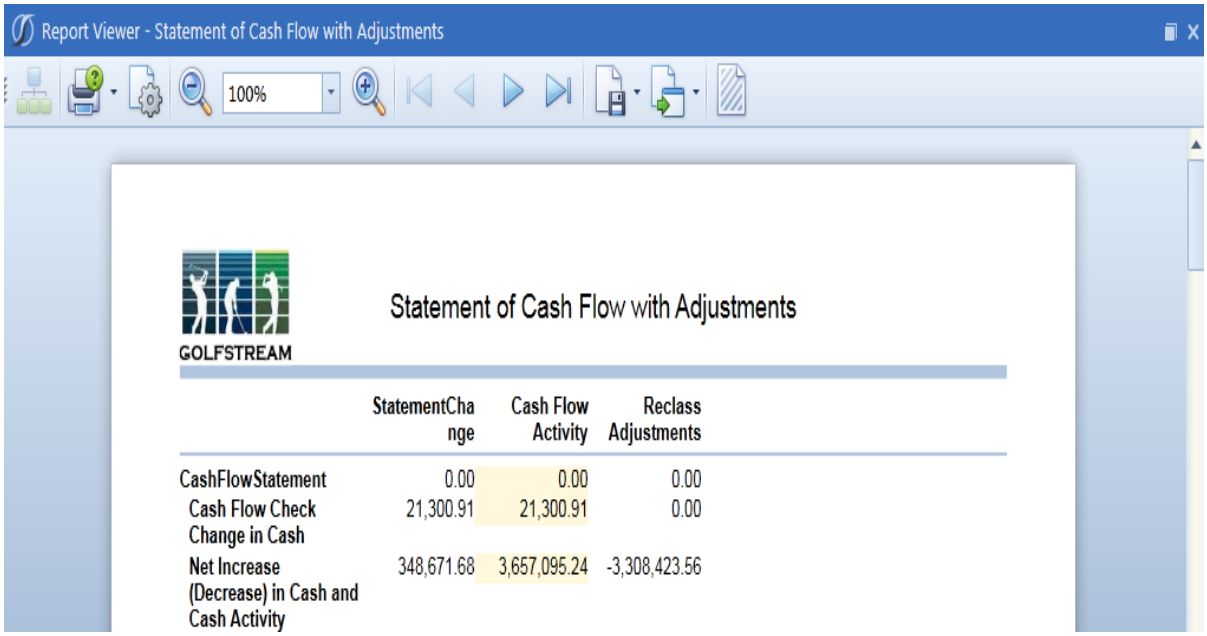
User Defined Description – Drill-Down

The Drill Down window displays the custom User Defined Description as the default header. If no custom description is created, the dimension type label, such as UD1, will display. See Application Properties and then User Defined Dimensions (Descriptions) for more information.



Show Cube View as a Report

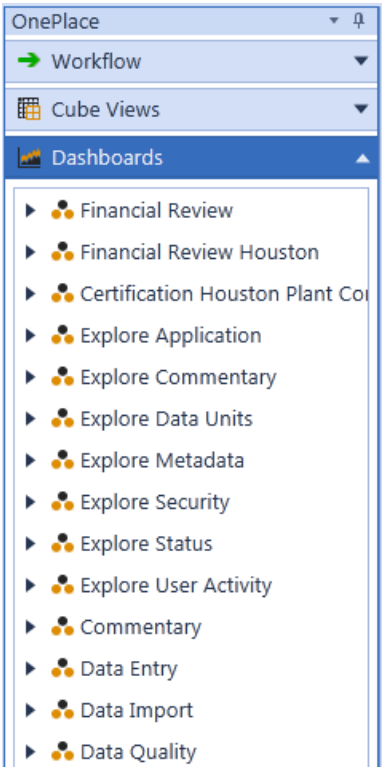
To generate a smoothly formatted presentation of the Cube View, click this button while in preview mode and a report similar to what one would see in Dashboards will open. Control column widths and row heights from within the Cube View. See Application Tab| Application Properties in order to show a company name and logo on all reports. There are also several application-wide settings for these Data Explorer reports under Application Properties and under the Application Tab| Presentation| Cube Views.



In order to print a Cube View, first generate a Report as shown above. Once the Report is generated, print and export that view to other Formats, such as PDF, HTML, RTF (for Word), CSV, Text, XPS, MHT or Excel.

Using OnePlace Dashboards

This is the section where all pre-built Application Dashboards can be viewed.



Select a Dashboard and see a complete and organized series of reports, grids, charts, and graphs all combined. A user may be prompted to enter Parameters in order to make the Dashboard relevant. The Parameters are pre-defined when the Dashboard is created in the Application Tab.

> **NOTE:** For OneStream Windows App, you must right-click the bookmark to jump to that location.

Dashboard Toolbar

Select Parameters

Use this to select specific Parameters when viewing a Dashboard

Reset Parameter Selections and Refresh Dashboard

Use this to change the Parameters and view the data differently in the Dashboard

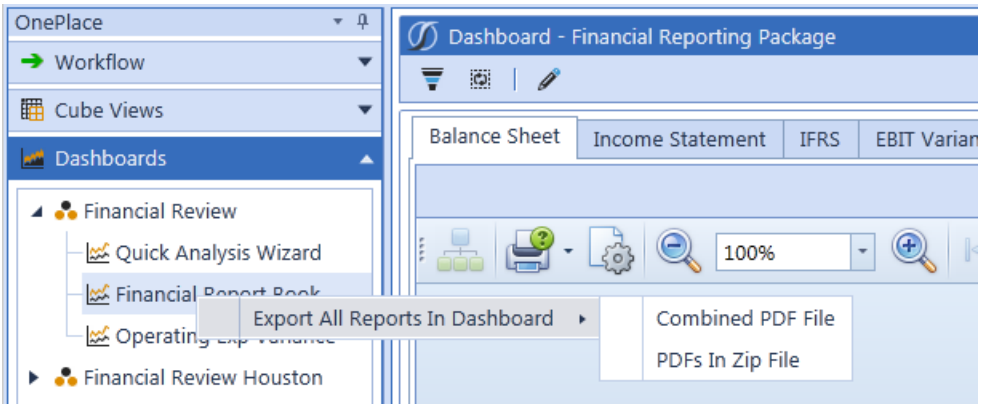
Edit Dashboard

If security permits, select this to launch the Application Tab and make changes to the Dashboard properties and its components.

Printing Options

Select to print a single Dashboard report from the web. Click the down arrow and select Click via PDF to turn the report into a PDF first and then print.

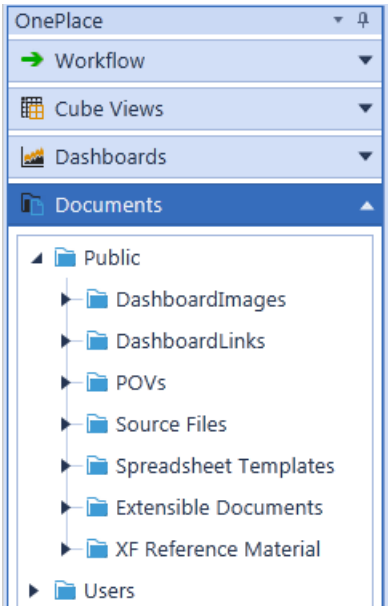
In order to save and print an entire Dashboard Book as one PDF, right-click on the Dashboard name and select Combined PDF File or PDFs In Zip File.



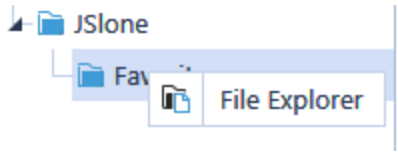
For more information on building Dashboards, see Dashboards in Presentation.

Storing Documents

An administrator can save public documents or templates for users to access for their close process. These documents are saved in the Systems Tab|Documents where only administrators have access. However, a user can access these public documents in OnePlace.



> **NOTE:** Right click on any document or folder and select File Explorer in order to upload files from the OnePlace or Systems tab. See File Explorer in System Tools for more details on this feature.



NOTE: Files created with either the Spreadsheet or Text Editor feature are also visible here. These files can be opened by Right clicking the file and selecting one of the three options.

- Open in Text Editor Page/Open in Spreadsheet Page
- Open- opens file in its related Microsoft application, if loaded on the local PC
- Open With… this allows the user to select which application they want to use to open the file

NOTE: The File Explorer option is also available by right clicking a file. Files can also be uploaded or opened from this window.

Excel Add-In

The Excel Add-In is an alternative way to enter, update, manage, query and analyze application data, using Excel spreadsheets and workbooks.

You can use the Excel Add-In to:

- Query and submit data into the application.
- Change the cube point of view.
- Interact with forms.
- Leverage multiple cube views.
- Establish a live connection to cube views in the application.
- Drill through to source data.
- Create functions to return, amend, or submit application data.
- Refresh sheets and workbooks when changes occur in the application.

You can launch Excel documents from:

- The Navigation pane
    - The **OnePlace** tab: Through public and user-specific access to application functionality and documents.
    - The **Application** tab: Through the Spreadsheet tool.
    - The **System** tab: Using export features for system administrators.
- Workflow forms that have attached templates.

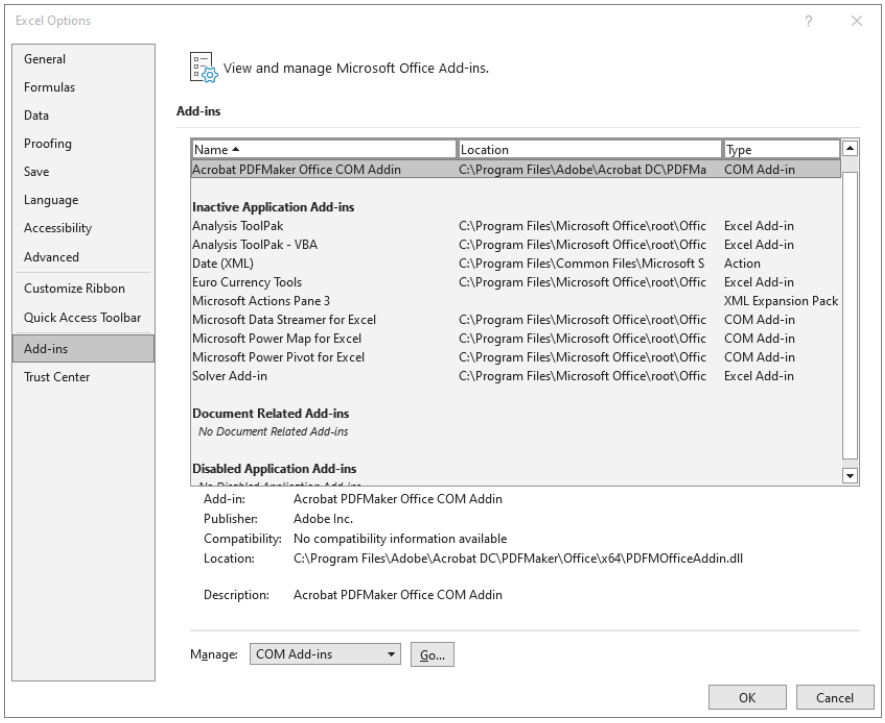
- Cell or data unit attachments.
- Dashboards
- Cube views
- File Explorer

Connect the Excel Add-In

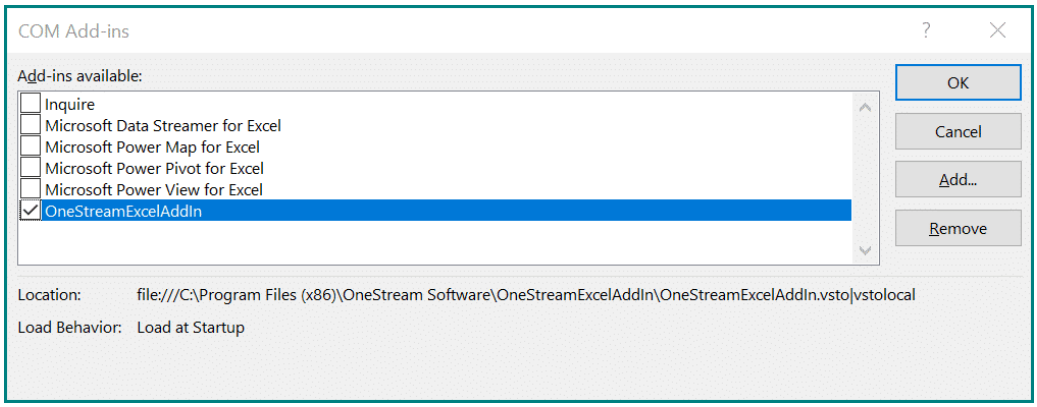
Before you start using Excel Add-In, you must add the OneStream ribbon to your Excel program.

Add the OneStream Ribbon

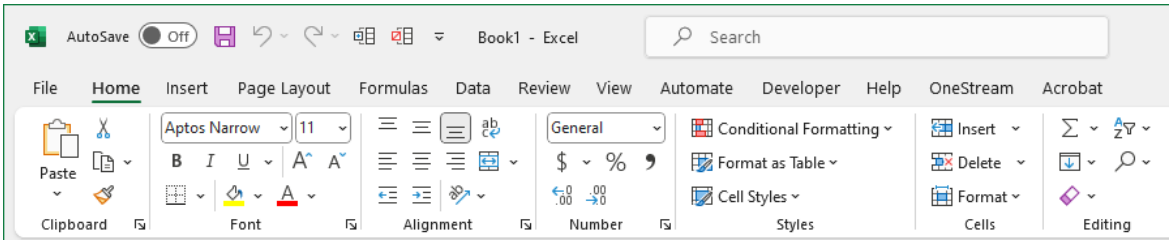
1. In an Excel workbook, navigate to **File** > **Options** to open the **Excel Options** dialog.
2. Select **Add-ins** from the menu in the left pane. Click the **Manage** drop down menu at the bottom of the dialog box and select **COM Add-ins** and then click **Go**.



3. Click the box next to **OneStreamExcelAddIn** and then click **OK**.

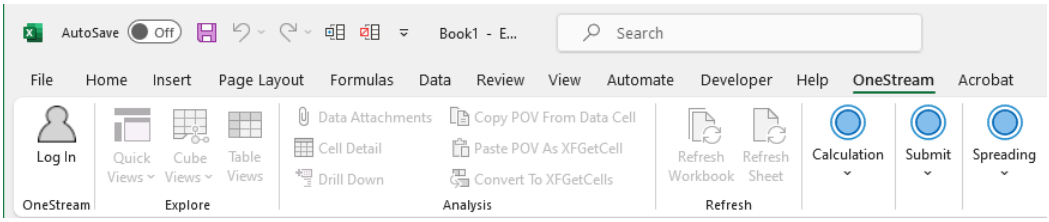


4. The OneStream ribbon is displayed at the top of your workbook page.

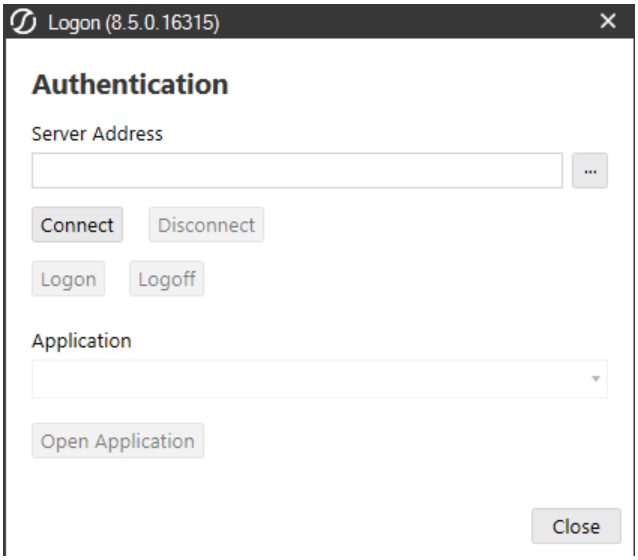


Log Into the Excel Add-In

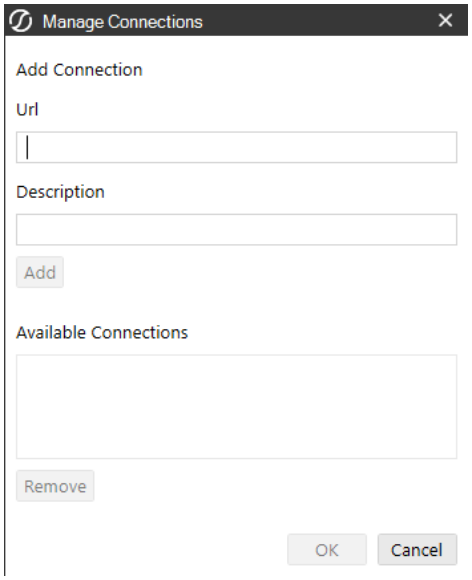
1. From the OneStream ribbon in Excel, click **Log In**.



2. In the **Server Address** field, click the ellipsis to add or select the URL for the server.

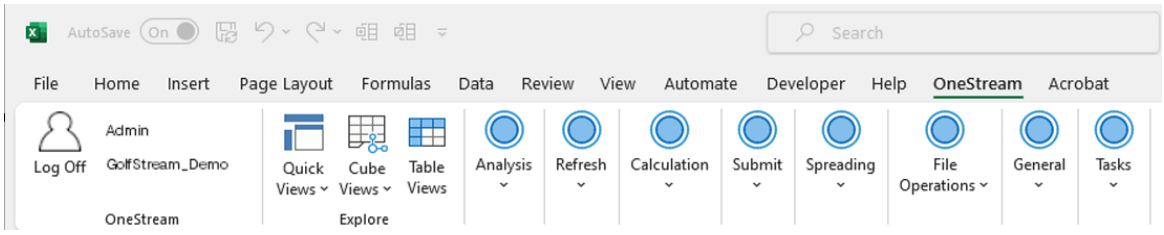


3. Select an available connection or add a new connection by entering the URL and description and clicking **Add**. After you select a connection, click **OK**.



4. Click **Connect** to authenticate the server.
5. Enter your username and password and click **Logon**.
6. In the application field, select an application and click **Open Application**.

   After completing the authentication files and opening the application, your username and application name display in your Excel workbook, indicating that you are logged into the Excel Add-In.



Update the Excel Add-In

If you use the Excel Add-In provided with the Windows application, update the Excel Add-In as part of your client update. See Client Updater. Otherwise, update the Excel Add-In by uninstalling then reinstalling it as described in the *Installation Guide.*

Compatibility

OneStream makes every effort to support complex environments and maintain compatibility with other add-ins, by testing our Excel Add-in using a standard desktop deployment. There are factors outside our control which can negatively impact compatibility, so we cannot make a guarantee or warranty of service for compatibility.

Add-In Components

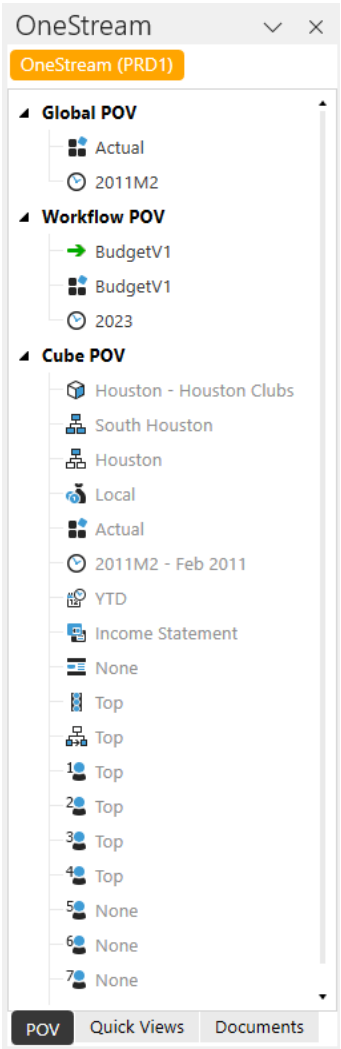
The Excel Add-In contains three main components:

- **Task Pane**: Used to quickly access POV, Quick Views, and Documents.
- **OneStream Ribbon**: Use the OneStream menu item in the ribbon to access data from the platform.
- **Error Logs**: Use Error Logs to troubleshoot Add-In related issues.

Task Pane

The Excel Add-in **Task** pane includes the **POV**, **Quick Views** and **Documents** tabs as shown below.

- **POV**: Identifies the **Global POV**, **Workflow POV**, and **Cube POV**. This tab is linked to the POV in your application. From here, you can make updates to the cube POV.
- **Quick Views**: Lets you build and edit quick views in the Excel Add-in. See Quick Views.
- **Documents**: Displays public or user-specific application or system documents. Access and manage documents with the File Explorer.



OneStream Ribbon

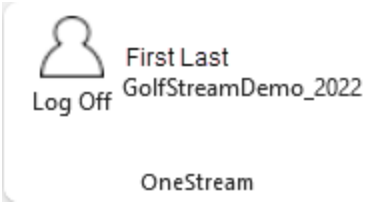
After installing the Excel Add-In, there will be a OneStream menu item and a ribbon in Excel. The ribbon contains these categories:

- OneStream
- Explore
- Analysis
- Refresh
- Calculation
- Submit
- Spreading
- File Operations
- General
- Tasks

> **NOTE:** When working in Excel, decimals are automatically truncated after the ninth character in a cell or a function.

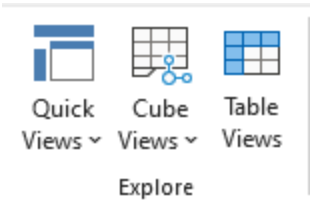
OneStream

This displays the current user and application. A user can log off or logon to a different application by clicking this icon.

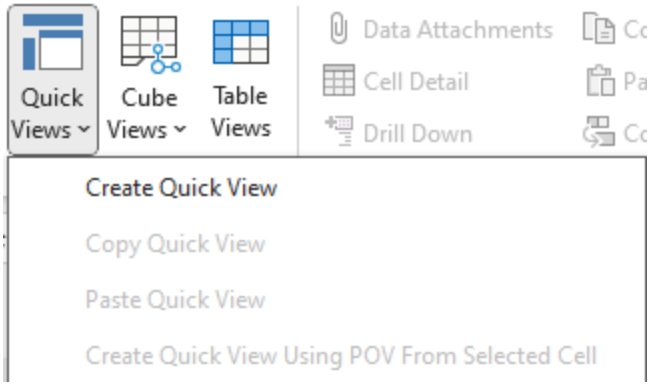


Explore

The Explore category enables you to quickly build a Quick View, Cube View, or Table View in Spreadsheet or Excel.



Quick Views



Create Quick View

This will create a new Quick View in the worksheet's selected cell.

Copy Selected Quick View

This will copy the selected Quick View in order to paste a version of it in another spreadsheet or workbook.

Paste Quick View

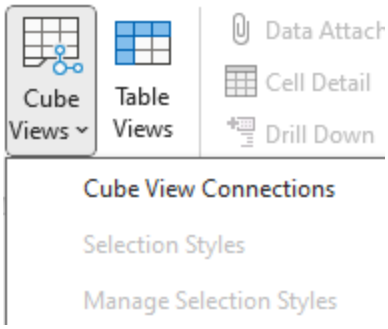
This will paste the copied quick view to the current spreadsheet or another workbook.

Create Quick View Using POV from Selected Cell

This will create a new Quick View based on the current POV from the selected cell. This can be done using a Quick View cell's POV or a Cube View cell's POV.

See Quick Views for more information.

Cube Views



Cube View Connections

Create a new Cube View Connection in the selected cell. .

Selection Styles

Create selection styles for Cube Views. See Selection Styles for Cube Views.

Manage Selection Styles

This enables you to modify available selection styles.

Table Views

Select **Table Views** to insert a Table View into your Excel sheet.

Analysis

Data Attachments
This pulls up the Data Attachments dialog to show existing comments or attachments on a selected cell, or to allow data attachment edits.

Cell Detail
Enter Cell Detail for a Cube View or Quick View data cell. See *Cell Detail* in *OnePlace Cube Views* in the *Design and Reference Guide*.

Drill Down
Drill down on a specific cell in order to see more details or gather more information. See *Drill Down* in *OnePlace Cube Views* in the *Design and Reference Guide*.

Copy POV from Data Cell
This captures the Point Of View of the currently selected cell. After clicking this option, the Paste POV As XFGetCell becomes available and the Copy POV From Data Cell goes to gray. The ability to paste this into another cell is now available and OneStream will automatically convert this into an XFGetCell formula with all of the appropriate Parameters.

Paste POV As XFGetCell

This option is only available after clicking Copy POV From Data Cell. After clicking this option, OneStream will convert the copied cell into an XFGetCell formula. Click Refresh Data to retrieve the data.

Convert to XFGetCells

This will convert an existing Quick View into an XFGetCells. After clicking this option, OneStream will prompt with the following: Are you sure you want to convert all of the data in Quick View 'Name of the Quick View' to XFGetCells? By clicking OK, the Quick View definition will be deleted and converted to XFGetCells.

Refresh

Refresh Sheet

This pulls down updated data from the server and only refreshes the selected worksheet.

Refresh Workbook

This pulls down updated data from the server and refreshes the entire Excel workbook.

	Refresh Sheet	Refresh Workbook
Function Behavior	Refreshes the selected tab only	Refreshes all tabs in the file
Data Impacts	Clears all dirty cells on the selected tab only	Clears all dirty cells on all tabs regardless of selected tab
Parameter Impacts (CV Only)	Prompts the user with any Parameters used on the selected tab	Prompts the user with all the Parameters used in the workbook

Calculation

Consolidate/Translate/Calculate
If permission is granted, these calculations can be performed on the selected cell.

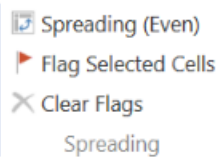
Submit

Submit Sheet
After editing data in Excel, click this icon to send it back to OneStream for Cube Views, Quick Views, XFSetCells, and Table Views on the active sheet.

Submit Workbook
After editing data in Excel, click this icon to send it back to OneStream. This icon will send data back for every tab in the Excel workbook.

	Submit Sheet	Submit Workbook
Function Behavior	Identifies data changes on the selected table and stores these changes to the database	Identifies data changes on every tab and stores these changes to the database
Data Impacts	Submits all data for the selected tab only	Submits all data for all tabs
Parameter Impacts (CV Only)	No prompts	No prompts

Spreading



Enables users to see what type of spreading was used to spread data values over several columns or rows without having to type in each cell's values.

Spreading Types

Fill
This fills each selected data cell with the value in the Amount to Spread property.

Clear Data
This clears all data within the selected cells.

Factor
Multiply all cells by the specified rate.

Accumulate
This takes the first selected cell's value and multiplies it by the rate specified. It then takes that value, multiplies it by the specified rate and places it in the second cell selected, and does this for all selected cells. For example, four cells are selected and the first cell has a value of 900.

Even Distribution
This takes the Amount to Spread and distributes it evenly across the selected cells.

445 Distribution
This takes the Amount to Spread and distributes it with a weight of 4 to the first two selected cells and a weight of 5 to the third cell.

454 Distribution

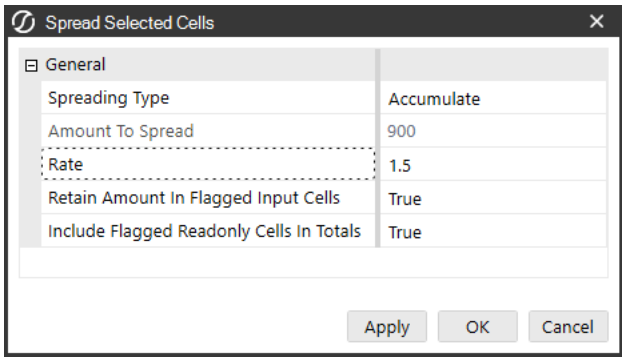
This takes the Amount to Spread and distributes it with a weight of 4 to the first selected cell, a weight of 5 to the second cell and a weight of 4 to the third.

544 Distribution

This takes the Amount to Spread and distributes it with a weight of 5 to the first selected cell and a weight of 4 to the second and third cells.

Apr 2011	May 2011	Jun 2011	Jul 2011
909.00	**14.00**	**622.50**	**33.75**
900.00	0.00	0.00	0.00

The Accumulate Spreading is setup as follows with a rate of 1.5:



When the spreading is applied the outcome is as follows:

Apr 2011	May 2011	Jun 2011	Jul 2011
1,359.00	**2,039.00**	**3,660.00**	**4,590.00**
1,350.00	2,025.00	3,037.50	4,556.25

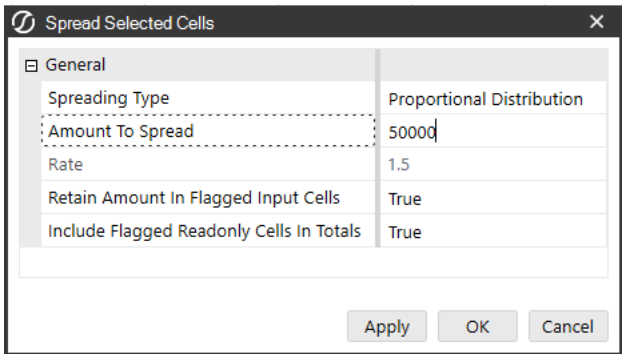
Each cell's value is a factor of the previous cell amount.

Proportional Distribution

This takes the selected cell's value, multiplies it by the specified Amount to Spread, and then divides it by the total sum of all selected cells. If all the cells have a zero value, the Amount to Spread will behave like an Even Distribution.

Apr 2011	May 2011	Jun 2011	Jul 2011
1,359.00	**2,039.00**	**3,660.00**	**4,590.00**
1,350.00	2,025.00	3,037.50	4,556.25

A proportional amount of 50,000 is applied to the cells:



Result:

Apr 2011	May 2011	Jun 2011	Jul 2011
6,162.85	**9,244.77**	**14,468.65**	**20,802.98**
6,153.85	9,230.77	13,846.15	20,769.23

Spreading Properties

Amount to Spread

Specify the value to spread over the selected cells. The value defaults to the last cell selected. The way the amount in this field spreads varies by Spreading Type.

Rate (Factor and Accumulate Spreading Types Only) Enter a rate to multiply by a cell value.

Retain Amount in Flagged Input Cells

Users can flag specific cells in order to retain the data within the cell. If this property is set to True, spreading will not apply to the selected flagged cells.

Include Flagged Readonly Cells in Totals

Set this to True to include locked base-level cell values when calculating spreading totals. True is the default.

Flag Selected Cells

Flags selected cells so the original amount in the cell is retained during the spreading process.

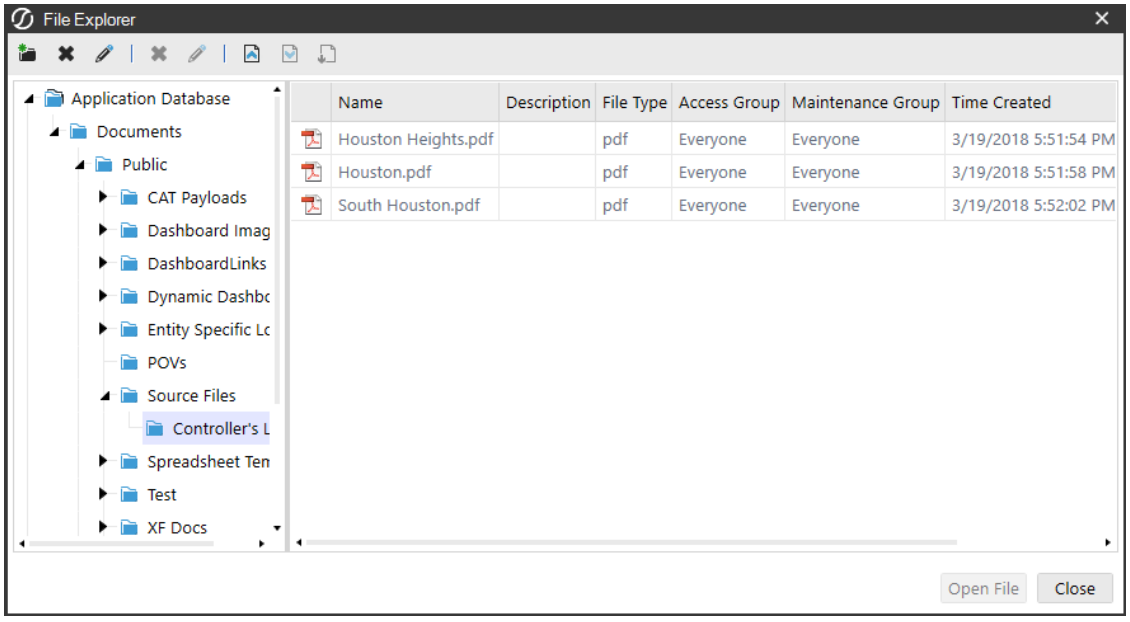
Clear Flags

Select this to clear any flagged cells.

File Operations

File Explorer

Use File Explorer to upload and download files.



Create Folder

This creates a new folder under the selected folder on the left-hand side of the File Explorer pane.

Delete Selected Folder/File

This deletes the selected folder on the left-hand side of the File Explorer pane or the selected file.

Edit Selected Folder/Edit Selected File Information

This edits the Description, Maintenance Group, and Access Group for the selected folder or file.

Upload File

This uploads the selected file and allows the user to save.

Download Selected File

This downloads the selected file and allows the user to save.

Download Selected File's Content File

This downloads the selected file's content file and allows the user to save.

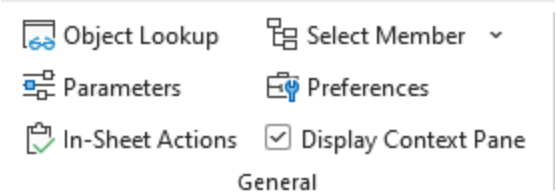
Save to Server

Use this option to save your file to the OneStream folder structure. You can save files to any location matching your security access. If the file was opened from a OneStream location, it will be saved back to that location. If it wasn't, File Explorer will open.

Save Offline Copy

Use this to save an offline copy of the current worksheet without the functions. Users without the Excel Add-In can open this copy and see the saved values.

General



Object Lookup

Use the Object Lookup to insert objects from OneStream into Excel such as Foreign Exchange Rate Types when building formulas. If creating an Extensible Document in Excel, users can use the Object Lookup to insert Parameters, Substitution Variables, or Image Content. See Object Lookup in Presenting Data With Books, Cube Views and Dashboards for more details on this feature.

In-Sheet Actions

Create buttons to execute a Data Management Sequence, Submit data, or Refresh data, without leaving the sheet. See In-Sheet Actions.

Parameters

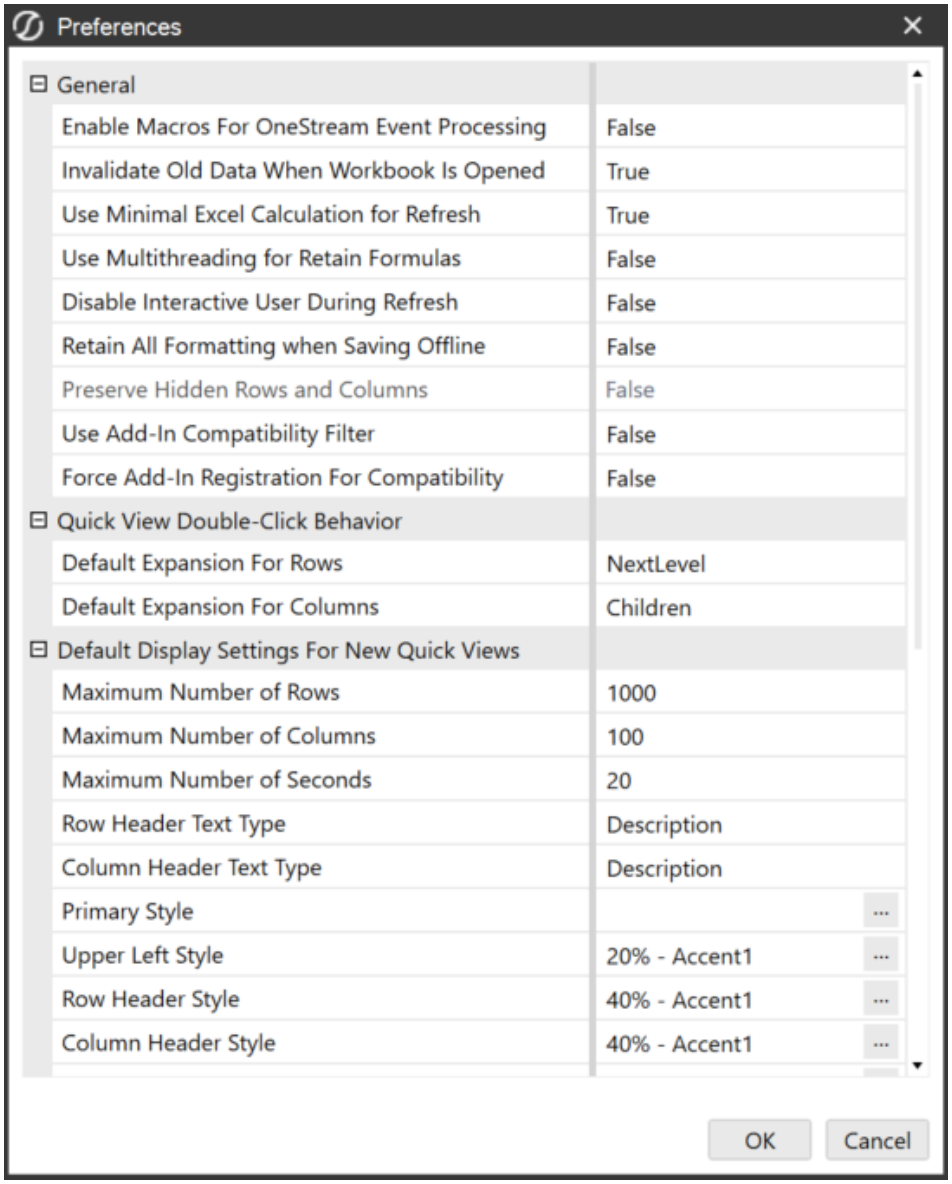
Insert a new parameter to quickly filter Cube Views and Table Views or manage existing parameters in the workbook. See Parameters.

Select Member

Select a Dimension Type from the drop-down list in order to view the Members of that Dimension. Select a Member of the hierarchy, and the Member name will display in the selected cell.

Preferences

Set Preferences for how to interact with and display your OneStream data. Settings include:



General

- **Enable Macros for OneStream Event Processing**: If set to True, this enables Excel macros for OneStream API calls. The default is False.

- **Invalidate Old Data When Workbook is Opened**: If set to True, this will force a data refresh on the opened workbook. The default is False.
- **Use Minimal Calculation for Refresh**: This is for Excel Add-In only, not the Spreadsheet feature in OneStream Windows App. The default is True this will only calculate formulas and Excel functions in the active sheet. Set to False to revert to a full calculation of all workbooks and all sheets.

  > **NOTE:** Performance is best when Excel is set to use Manual Calculation Mode.

- **Use Multithreading for Retain Formulas**: When set to True, processing steps for collecting and applying formulas to Cube Views with retain formulas enabled will run concurrently for multiple worksheets, allowing for faster refresh rates. This is for Excel Add-In only, not the Spreadsheet feature in OneStream Windows App.
- **Disable Interactive User During Refresh**: Accounts for a known Excel situation when running on certain touchscreen hardware. If the **Refresh Sheet** or **Refresh Workbook** is pressed but the cells containing functions do not complete their calculations when processed, change the **Disable Interactive User During Refresh** setting under **Preferences** to True.

  > **NOTE:** Setting this to True may result in incompatibility issues with other Excel Add-ins.

- **Retain All Formatting when Saving Offline**: This is for Excel Add-In only, not Spreadsheet. The default is False to derive basic formatting and better performance. Set this to True to obtain all character by character formatting, this will force a data refresh on the opened workbook.

- **Preserve Hidden Rows and Columns**: This setting becomes active when the option Retain All Formatting when Saving Offline is set to True. When set to True, the rows and columns remain hidden when the workbook is saved offline. Rows and columns are visible when set to False.
- **Use Add-In Compatibility Filter**: When True, only cell selection change events, such as keystrokes or mouse clicks, are allowed. Third-party add-ins and macros cannot change cells. When False, users, third-party add-ins, and macros can make cell changes.
- **Force Add-In Compatibility for Registration**: When set to True, this setting forces a re-registration of the OneStream Add-In. This should be used when the Add-In is turned off on start-up by external add-ins.

Quick View Double-Click Behavior

- **Default Expansion for Rows/Columns**: This determines what level of expansion displays when a user double-clicks a Quick View Row or Column Header. NextLevel is the default setting and allows multiple expansion paths when a user double clicks a row or column header. There is also the ability to double-click an expanded item to collapse it again. This feature only works with the NextLevel setting.

Default Display Settings for New Quick Views and **Default Suppression Settings for New Quick Views**.

See Quick Views in Getting Started with the Excel Add-In.

Display Context Pane

To display the OneStream task pane on the right-hand side of the screen, check this box. To hide the task pane, disable the box.

Default Settings for Preserve Formatting

- **Preserve Excel Formatting by Default (Quick Views)**: When set to True, formatting will be preserved by default for Quick Views. The default setting is False.
- **Preserve Excel Formatting by Default (Cube Views)**: When set to True, formatting will be preserved by default for Cube Views. The default setting is False.
- **Preserve Excel Formatting by Default (Table Views)**: When set to True, formatting will be preserved by default for Table Views. The default setting is False.

Tasks

Task Activity
View all running, scheduled, and completed tasks.

Automatic

The Excel Calculation icon has the option of Automatic, Automatic Except for Data Tables, and Manual. It is recommended that the Calculation be set to Manual when using OneStream] spreadsheets because the Automatic setting results in an Excel re-calculation every time a OneStream's interactive workbook changes data (e.g., when navigating a Quick View). However, this is not forced because a user might prefer Excel's Automatic calculation, especially when there is not a significant amount of OneStream data in the workbook.

Error Logs

The Excel Add-In stores error log details on your local drive in the AppData folder. This allows customers and customer support to retrieve log details when experiencing Excel Add-In related issues, resulting in decreased time to resolution.

Error logs in the folder will automatically be deleted after 60 days.

Work with Data

The Excel Add-In enables you to quickly build and work with data including through Cube Views, Quick Views, Table Views, and XF Retrieves.

Cube Views

Cube View Connection Settings

When you create a new Cube View Connection, you can populate the following settings:

Property	Description
Name	The name is created for you if you select the Cube View first. If you type a name first and then select the Cube View, the typed name is retained.
Refers To	This setting specifies the starting cell or range in your spreadsheet for the Cube View. It establishes the default cell or range reference for the Cube View Connection.
Cube View	Click the ellipsis to look up and add a Cube View to make a Cube View Connection with.
Resize Initial Column Widths Using Cube View Settings	Selected by default, this option automatically re-sizes column widths based on the settings from the Cube View selected.

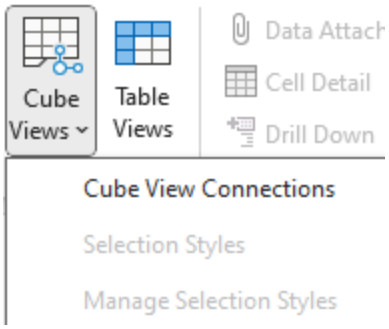
Insert or Delete Rows When Resizing Cube View Content	Select this option if you plan to stack everything vertically on your worksheet. This automatically adds or deletes rows as a Cube View expands and contracts.
Insert or Delete Columns When Resizing Cube View Content	Select this option if you plan to stack everything horizontally on your worksheet. This automatically adds or deletes columns as a Cube View expands and contracts.
Include Cube View Header	This option includes the header on the Cube View.
Retain Formulas in Cube View Content	This option allows you to enter formulas in the Cube View (Excel or Spreadsheet) and retain those formulas before and after submission of the sheet or workbook.
Dynamically Evaluate Highlighted Cells	This option is only available if the previous option is selected. This highlights a cell if the formula reference has changed without having to refresh a spreadsheet.
Preserve Excel Format	This option enables you to preserve native Excel formatting changes made to Cube Views. When enabled, formatting changes made via native Excel formatting will be retained.

Add Parameter Selectors to Sheet	Selecting this option will generate parameter selectors for all parameters in the selected Cube View, including any dependent parameters. These parameter selectors can be drop-down menus or free-form cell inputs. This option must be selected during the creation process and cannot be added retroactively. To edit parameters added during the creation of a new Cube View Connection, navigate to **OneStream** > **General** > **Parameters** from the Excel or Spreadsheet ribbon. **NOTE:** If parameters are created using Add Parameter Selectors to Sheet, they will need to be manually deleted from the Parameters ribbon if the associated Cube View Connection is deleted. Deleting the connection alone will orphan the parameters that were added to the sheet.

Add a Cube View to Excel

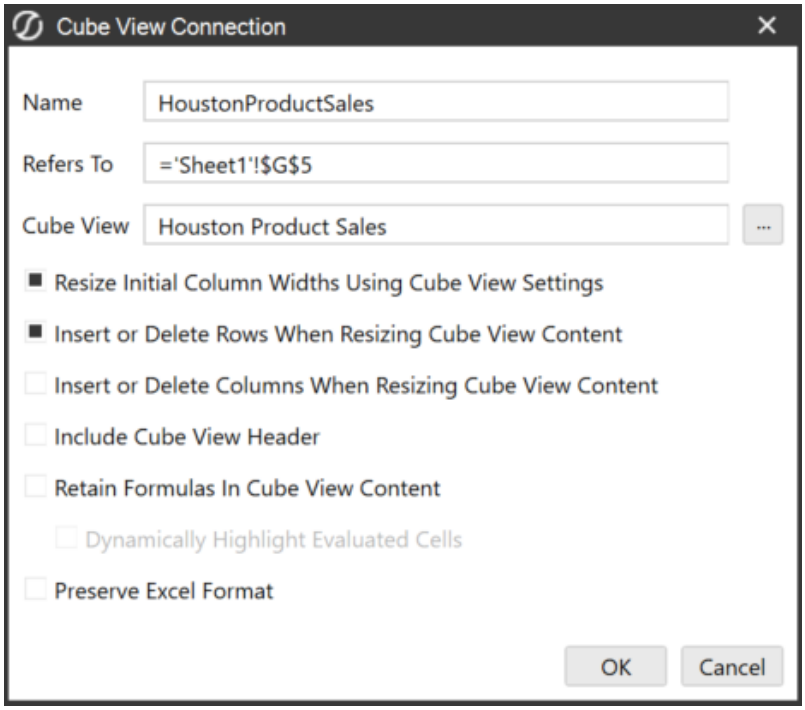
Add a Cube View to an Excel sheet using the below steps:

1. Click **Cube Views** > **Cube View Connections**.



2. From this window, the cube views added to an Excel workbook can be managed. You can add, remove, edit, or go to styles. Click **Add**, to add a new Cube View.

3. Name the connection and then choose the Cube View.



4. Populate the following fields:

    a. **Resize Initial Column Widths Using Cube View Settings** is the default setting. If you disable it, you can change the columns and save the Cube View. However, if you go back into the same Cube View Connection, the check box will be enabled and you'll need to disable it again to keep your new cube view settings.

    b. **Insert or Delete Rows/Colums When Resizing Cube View Content**: Select whether there needs to be inserted or deleted rows and/or columns when resizing. This setting will move around other content in the sheets if the size of the Cube View changed since the last refresh.

    c. **Include Cube View Header**: Choose whether to add header rows to the spreadsheet.

d. **Retain Formulas in Cube View Content** allows you to enter formulas in the Cube View (Excel or Spreadsheet) and retain those formulas pre and post submission of the sheet or workbook. When the sheet or workbook is refreshed the formulas will remain. If the value resulting from the value is different than the value of the OneStream database, the cell will initially become a dirty cell and will turn the cell format to yellow. See Retain Formulas in Cube View Content in Excel and Spreadsheet.

> **NOTE:** When using external Excel workbooks, or after any updates to referenced sheets within the same workbook, you must Refresh Sheet to visualize the dirty cells and then Submit Sheet, unless Dynamically Highlighted Evaluated Cells is turned on.

e. **Preserve Excel Format**: This option enables you to preserve native Excel formatting changes made to Cube Views. When enabled, formatting changes made via native Excel formatting will be retained. See Preserve Formatting.

5. Click the **OK** button and then the **Close** button to view your Cube View.

   After the Cube View is added, it will appear on the sheet. If formatting was applied to the Cube View, see Cube Views in Presentation, the formatting will come forward into the Excel sheet. Otherwise, apply Excel Styles. These styles are stored in the Excel sheet and can be copied between workbooks. For more information on Excel Styles, see Styles. If you enabled Preserve Excel Format, your formatting will remain. See Preserve Formatting.

   > **NOTE:** In order to copy Excel spreadsheet cells into a Data Explorer Grid on the web, click CTRL, select the cells desired, and then click CTRL-C. Navigate to the Data Explorer Grid, select a cell, and click CTRL-V, this will paste the cells into the grid. This can also be done from a Data Explorer Grid into an Excel Spreadsheet.

Retain Formulas in Cube View Content in Excel and Spreadsheet

Retain Formulas in Cube View Content allows you to form Cube View grids of data in Excel, using the Cube Views menu function, that can be linked to other Excel models for easy submission into OneStream. This feature allows formulas, (to writeable cells,) in Excel (or Spreadsheet) for an attached Cube View to be retained on submission and retrieval instead of being replaced with the value of the represented formula.

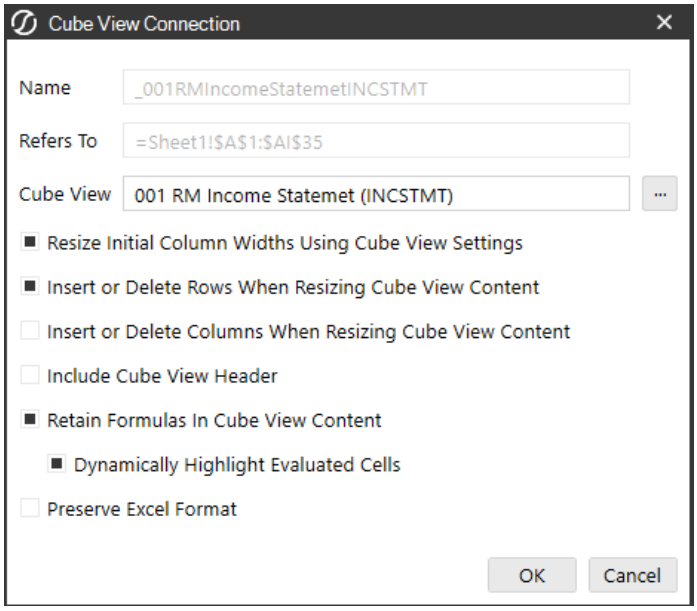
The Retain Formulas in Cube View Content feature, allows users to plan, budget or forecast and use the familiar functionality of Excel while still submitting data back to the OneStream database.

Use the Retain Formulas in Cube View Content feature to enter formulas in the Cube View (Excel or Spreadsheet) and retain those formulas pre and post submission of the sheet or workbook. When the sheet or workbook is refreshed, the formulas will remain. If the value resulting from the formula differs from the existing value in the OneStream database, the cell will initially become a dirty cell and will turn the cell format to yellow.

> **NOTE:** When using external Excel workbooks, or after any updates to referenced sheets within the same workbook, you must Refresh Sheet to visualize the dirty cells and then Submit Sheet, unless you've turned on Dynamically Highlight Evaluated Cells in the cube view.

Retain Formulas in Cube View Content links to other Excel worksheets or worksheets in other Excel workbooks.

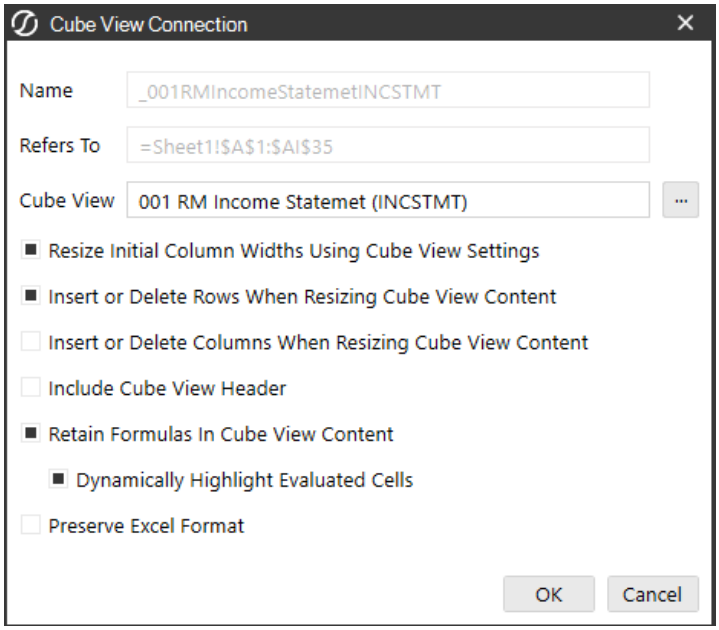
1. From the **OneStream** menu, select **Cube Views** > **Cube View Connections**.
2. Click **Add** in the Cube View Connection window or click **Edit** if you already have a cube view.
3. Click **Retain Formulas in Cube View Content** box and click the **OK** button.



4. Add the Cube View, if one is not already selected, and click the **Close** button.

Dynamically Highlight Evaluated Cells in Excel or Spreadsheet

When **Retain Formulas in Cube View Content** is enabled, the option to **Dynamically Highlight Evaluated Cells** becomes available to enable. When it's enabled, every time you make a change to a cell in Excel or Spreadsheet that is referenced in a Cube View, the cell will immediately update and show the update with a change in color. This cell update is called a dirty cell, which indicates that the cell value is different from the information in the OneStream database.

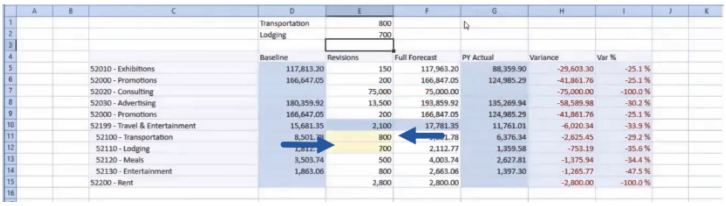


Dynamically Highlight Evaluated Cells saves you a step because the cell changes without requiring a refresh. This feature evaluates all the cells in the spreadsheet and identifies the values in the cube view that have changed relative to its original value in the database.

Excel users who want to continue working in Excel to access can log in through the OneStream menu, update the cube view content and submit it to the database without leaving Excel. You can also perform these tasks in Spreadsheet within the application.

You can **Retain Formulas in a Cube View Content** that are related to values within a function, existing workbook, sheet, other sheets, in external workbooks, and in external renamed worksheets in Excel. Spreadsheet also offers this functionality, but it doesn't allow you to point the cell references to external workbooks.

Click **Refresh Sheet** to see all changes within the cube view content and then click **Submit Sheet** or activate **Dynamically Highlight Evaluated Cells** and the cell updates automatically.



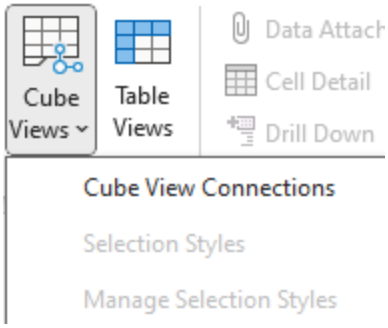
Transportation	800						
Lodging	700						
	Baseline	Revisions	Full Forecast	PY Actual	Variance	Var %	
52010 - Exhibitions	117,813.20	150	117,963.20	88,359.90	-29,603.30	-25.1 %	
52000 - Promotions	166,647.05	200	166,847.05	124,985.29	-41,861.76	-25.1 %	
52020 - Consulting		75,000	75,000.00		-75,000.00	-100.0 %	
52030 - Advertising	180,359.92	13,500	193,859.92	135,269.94	-58,589.98	-30.2 %	
52000 - Promotions	166,647.05	200	166,847.05	124,985.29	-41,861.76	-25.1 %	
52199 - Travel & Entertainment	15,681.35	2,100	17,781.35	11,761.01	-6,020.34	-33.9 %	
52100 - Transportation	8,501.[illegible]	800	[illegible]1.78	6,376.34	-2,625.45	-29.2 %	
52110 - Lodging	[illegible]	700	2,112.77	1,359.58	-753.19	-35.6 %	
52120 - Meals	3,503.74	500	4,003.74	2,627.81	-1,375.94	-34.4 %	
52130 - Entertainment	1,863.06	800	2,663.06	1,397.30	-1,265.77	-47.5 %	
52200 - Rent		2,800	2,800.00		-2,800.00	-100.0 %	

The number of cells with formulas in the cube view determines the amount of time it takes to update the cells. You can turn the feature on or off and only use **Refresh Sheet** to update the values in the cells. Changes will show very quickly, no matter the size of the worksheet, when using Spreadsheet.

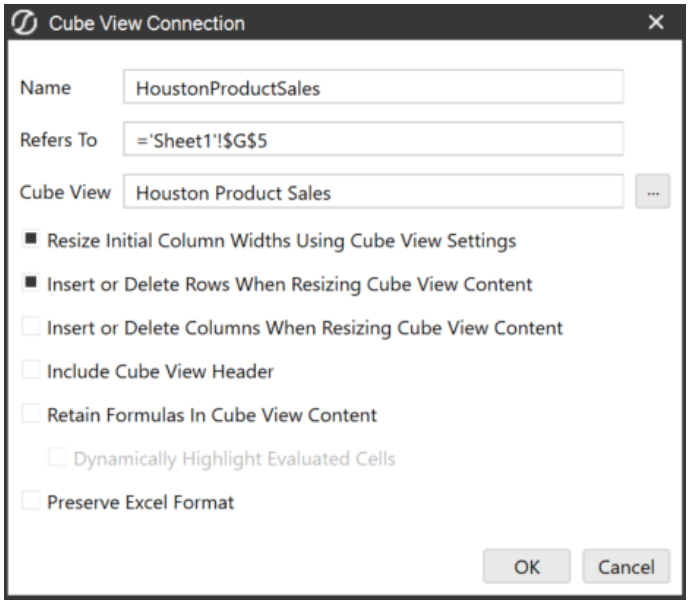
Using Retain Formulas and Dynamically Highlight Evaluated Cells

You can also use retain formulas and dynamically highlighted evaluated cells within a cube view to automatically display updated values in an existing workbook, a sheet or sheets, external workbooks, and external renamed worksheets in Excel. You can also do this in Spreadsheet within the OneStream application, however, you can't point the cell references to external workbooks.

1. In Excel, go to the OneStream menu and Log on.
2. Click **Cube Views** > **Cube View Connections**.

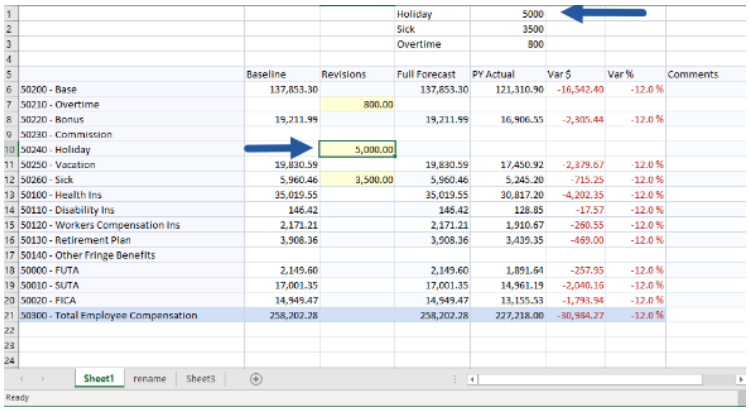


3. Click the **Add** button.
4. In the **Cube View Connection** window, click the ellipsis next to the **Cube View** field.



5. Select your choice and click the **OK** button.
6. Click **Retain Formulas in Cube View Content** to activate **Dynamically Highlight Evaluated Cells**.
7. Then click **Dynamically Highlight Evaluated Cells** so you can see the changes as they are made.
8. Even if you don't activate the dynamically highlight evaluated cells feature, you can click **Refresh Sheet** after you make changes to see them.
9. If you're prompted, click **OK** once you've selected the parameters for the cube view.
10. Once the cube view has been added, you can click **Edit** to review, if needed.

11. Make changes to the sheet and press <Enter> to see the updated cell, which will change from white to yellow.



1				Holiday	5000			
2				Sick	3500			
3				Overtime	800			
4								
5		Baseline	Revisions	Full Forecast	PY Actual	Var $	Var %	Comments
6	50200 - Base	137,853.30		137,853.30	121,310.90	-16,542.40	-12.0 %	
7	50210 - Overtime		800.00					
8	50220 - Bonus	19,211.99		19,211.99	16,906.55	-2,305.44	-12.0 %	
9	50230 - Commission							
10	50240 - Holiday		5,000.00					
11	50250 - Vacation	19,830.59		19,830.59	17,450.92	-2,379.67	-12.0 %	
12	50260 - Sick	5,960.46	3,500.00	5,960.46	5,245.20	-715.25	-12.0 %	
13	50100 - Health Ins	35,019.55		35,019.55	30,817.20	-4,202.35	-12.0 %	
14	50110 - Disability Ins	146.42		146.42	128.85	-17.57	-12.0 %	
15	50120 - Workers Compensation Ins	2,171.21		2,171.21	1,910.67	-260.55	-12.0 %	
16	50130 - Retirement Plan	3,908.36		3,908.36	3,439.35	-469.00	-12.0 %	
17	50140 - Other Fringe Benefits							
18	50000 - FUTA	2,149.60		2,149.60	1,891.64	-257.95	-12.0 %	
19	50010 - SUTA	17,001.35		17,001.35	14,961.19	-2,040.16	-12.0 %	
20	50020 - FICA	14,949.47		14,949.47	13,155.53	-1,793.94	-12.0 %	
21	50300 - Total Employee Compensation	258,202.28		258,202.28	227,218.00	-30,984.27	-12.0 %	
22								
23								
24								

Sheet1 rename Sheet3

Ready

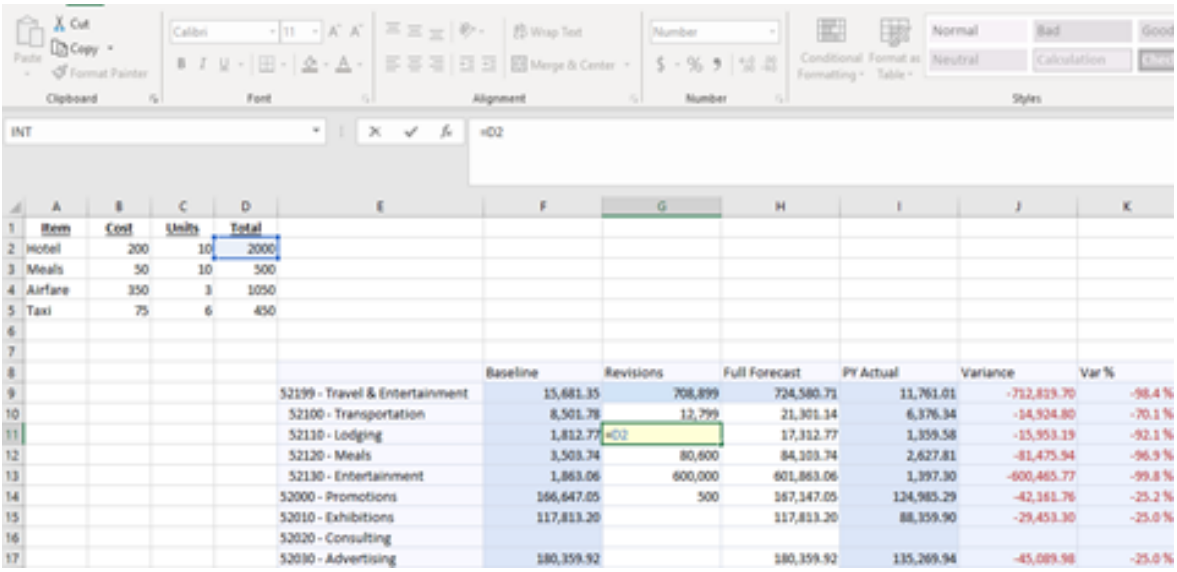
12. Click **Submit Sheet** to automatically save changes to the database.

Use Cases

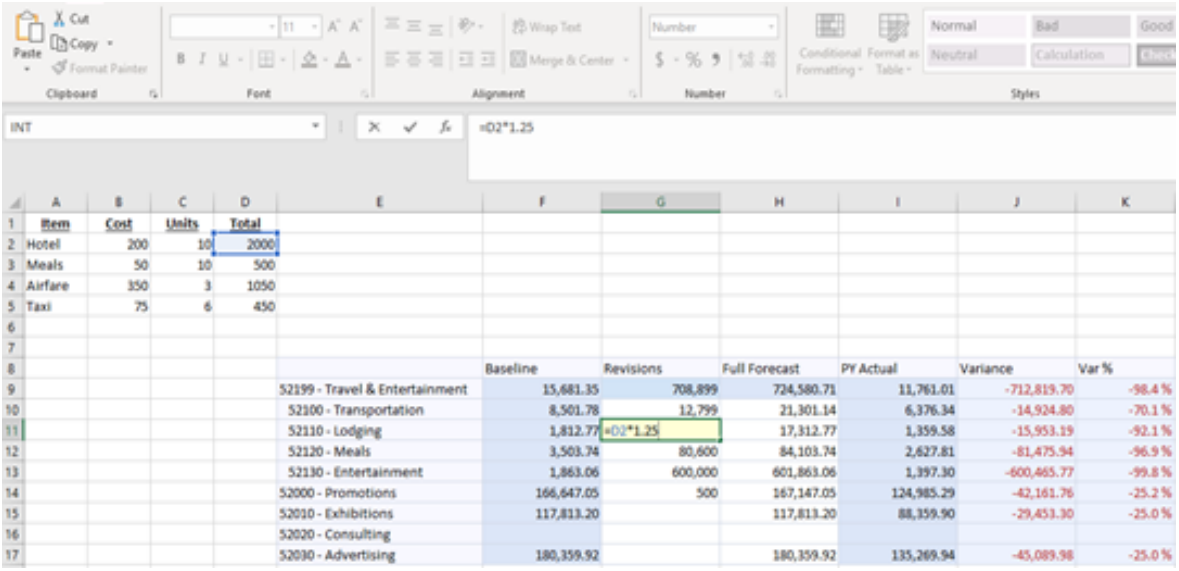
These use cases are for both Excel and Spreadsheet unless otherwise noted.

The placing of formulas or cell references. Retain Formulas can reference the following types of formulas. In all instances the formula will stay after refresh and/or submission.

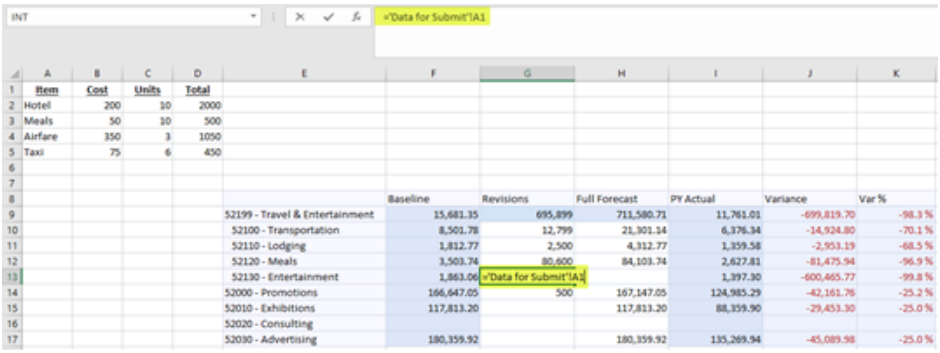
Cell References of individual cells of data on the same sheet.



Cell References to a cell on the same sheet, factored by another value.



Cell References to cells on other sheets. These can also be factored by another value as well.



Referenced cell(s) on another saved workbook can also be factored by another value.

> **NOTE:** This applies to Excel only.

Best Practices

Well-Formed Grid

It is suggested to create a “Well-Formed Grid” (Root.List or Comma Separated List) in Cube Views. When using this “Well-Formed Grid” (Root.List or Comma Separated List) in Cube Views, the Excel/SpreadSheet relative (=C2) and absolute formulas (=C2) will be retained.

However, when using these relative and absolute formulas within an Excel and Spreadsheet formula, users can use either the cell reference or text within the formula depending upon how members will be added or removed:

- =VLOOKUP(D30,Sheet1!A:B,2,FALSE) will work in a List or Comma-Separated list (Well-formed grid) when Accounts are added to the end.
- =VLOOKUP("52000 - Promotions",Sheet1!A:B,2,FALSE) will work in a case when a Member of a Row is moved up or down.

Member Expansion Functions

When using Member Expansion Functions in Cube Views for Excel and SpreadSheet, the cell being referenced within the function (Vlookup, etc), will need to be adjusted and/or referenced as text.

- =VLOOKUP("52000 - Promotions",Sheet1!A:B,2,FALSE) will work in a Dynamic or when a Member of a Row is moved.
- =VLOOKUP(D30,Sheet1!A:B,2,FALSE) will NOT work in a Dynamic or when a Member of a Row is moved as this is using the cell ref of D30.

Other Considerations

- Deselecting the Retain Formulas for Cube View Content will eliminate all formulas that were established /existed on the Cube View grid.
- Pivoting the existing Dimensions of the Cube View will break formulas.
- Changing the “structure” of the Cube View grid in the rows or columns will also break the formulas. For example; If you have Account, Entity, UD3 as the dimensions used in the row and switch it to UD3, Entity, Account, it will break the formulas.
- Users can change the POV to select a new dimension. This will change the Cube View results but retain the existing formulas that were established. The user at this point can choose to utilize the existing formulas, modify or delete. If the original formulas are modified or deleted, the last action will be saved.
- Linking a white cell (writeable cell) to another cell in a different workbook will work ONLY in Excel and NOT in Spreadsheet.
- Prior to establishing links to an external workbook, the user should save the external workbook being referenced.

- When the user renames or saves as the (referenced) file, the user will need to update the links to the newly created file. Updating the links on the spreadsheet should be done BEFORE doing a refresh or submit.
- Formulas with cell references (VLOOKUP, INDEX(MATCH(, etc) that return errors (#N/A, #ERROR, etc) or non-numeric data will not retain the formula and return to its original value from the Cube View ; this error text cannot be converted into a number so the formulas will not retain.
- If a Dimension Member Name is renamed; i.e.; “52200 – Rent” is now “52200 – Rent Commercial”, the formula will break.

Creating Excel Forms and Reports

Well-designed Cube Views help you create flexible and rich Excel-based forms and reports. You can retrieve capabilities through formula, but inserting Cube Views in a spreadsheet offers a richer experience with less maintenance.

Named Regions

Bringing a Cube View into Excel creates several Named Regions that you can select, refer to, and use with Styles. Named Regions are created for the Cube View, column headers, row headers, and data sections.

If there are multiple named columns or rows in the Cube View, go to the intersection-based Named Regions and use different formatting to differentiate sections. For example, a Total row is separated from detailed data. This combination of Named Regions and Styles generates a nicely formatted report:

	Net Income	Net Sales	Cost of Sales	Total Cur Assets
Clubs	5,719,400	204,339,789	114,594,599	282,477,228
NA Clubs	20,559,431	196,445,083	103,266,897	278,972,392
Canada Clubs	(15,679,089)	23,664,927	16,938,624	6,961,760
Montreal	(6,671,953)	10,070,182	7,207,925	2,962,451
Quebec City	(9,007,136)	13,594,745	9,730,699	3,999,309
US Clubs	35,947,682	173,219,127	86,642,475	272,149,769
Augusta	11,058,078	23,378,642	11,434,701	47,394,427
Carlsbad	11,920,655	17,421,183	5,151,048	40,585,020
Houston	12,983,949	132,419,302	70,056,727	184,172,322
Frankfurt	(11,216,955)	5,967,276	8,562,133	2,649,158

Cube Views in Spreadsheet

Cube views are the primary reporting tool in OneStream to display financial data for your organizational needs. You can also use cube view reporting in the Spreadsheet and Excel Add-in using cube view connections.

Define Cube View Connections

Cube view connections are different from exporting a cube view to Excel because the connection remains live, meaning data can be refreshed and submitted.

Before defining cube view connections:

- Make sure you have the appropriate security settings to access your Cube View and the Spreadsheet page in OneStream.
- Make sure your cube view is in a group within a profile that has visibility set for Excel.
- If using the Excel Add-in, connect to OneStream. In Excel, from the OneStream menu, click Logon.

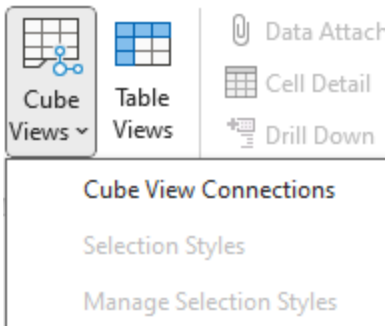
To add a cube view to Excel or Spreadsheet:

1. Select the cell that you want to place the cube view on. The uppermost left corner of your cube view will start on the cell you have selected.
2. Select **Cube Views** > **Cube View Connections** > **Add** from the OneStream ribbon.

Configure Cube View Connections

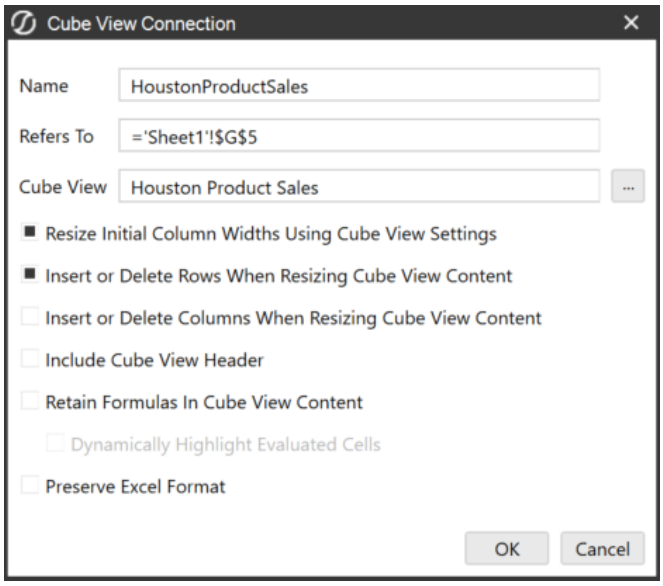
You can configure cube view connections in OneStream from the Spreadsheet functionality.

1. Click **OneStream** from the **Spreadsheet** ribbon.
2. From **Cube Views**, select **Cube View Connections** to open the cube view connections dialog box.

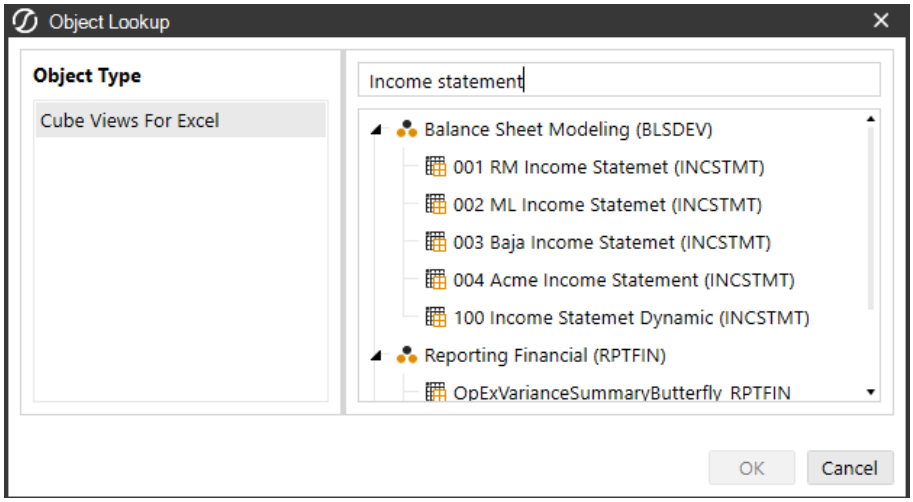


Add a Cube View to Excel or Spreadsheet

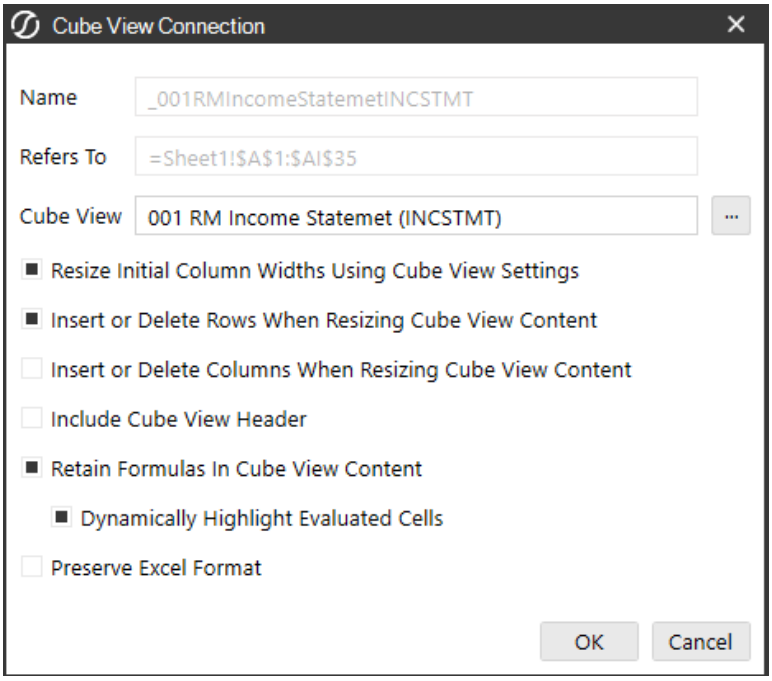
1. Click **Add** to create a cube view connection.



2. Click the ellipsis next to the Cube View field to open the **Object Lookup** dialog box. Object Lookup enables you to search for an existing cube view.
3. Use the Filter box to search for the name of the cube view and select it from the list of results.



4. Click the **OK** button to confirm the cube view selection.
5. The **Name** and **Refers To** fields automatically populate for the selected cube view. Click the **OK** button to continue.
6. Click **Close** to exit the dialog box. Your selected cube view is now added to the spreadsheet.



> **NOTE:** See Cube Views for more context on these properties.

Format Cube View Connections

In Spreadsheet, you can apply standard formatting to a cell, a group of cells, or to a row or column. Check the Preserve Excel Formatting setting in the Cube View Connection dialog box, or use Selection Styles and Named Ranges to format a Cube View Connection in Spreadsheet. See Formatting in Excel.

Quick Views

Quick views are ad hoc reports that allow you to pivot, drill back, create data sets, and design workbooks to analyze data. Quick views can also be used to submit data. You can create a quick view by:

- Entering members.
- Copying and modifying another quick view.

Quick views function similarly to cube views. See Cube Views in Spreadsheet in *Excel Add-In*.

Manage Quick Views

The **Quick Views** tab displays your dimensions. You can move and replace the default Time and Account dimensions. This table identifies the tasks you can perform on the **Quick Views** tab.

Button	Description
	Create a quick view.
	Rebuild a quick view created by direct data entry by selecting new or modified fields.
	Rename a quick view.
	Delete a quick view.
	Edit quick view properties.

Button	Description
	Refresh a quick view with the latest data set.
	Undo up to 100 changes.
	Restore deleted changes.
	Select members, identified by dimension type or dimension, to add to a quick view.
	Clear everything except selected items.
	Clear only a selected item.
	Go to the next level in the hierarchy.
	Go to the top of the hierarchy.

Button	Description
	Go to the parent member.
	Go to the immediate child members.
	Go to the branch base.

Quick View POV

If you do not select a member, the quick view POV reflects your default POV settings. User-defined descriptions also display in the quick view POV. Hover over dimension labels to view dimension descriptions.

See *Application Properties* and *User Defined Dimensions 1-8* in the *Design and Reference Guide*.

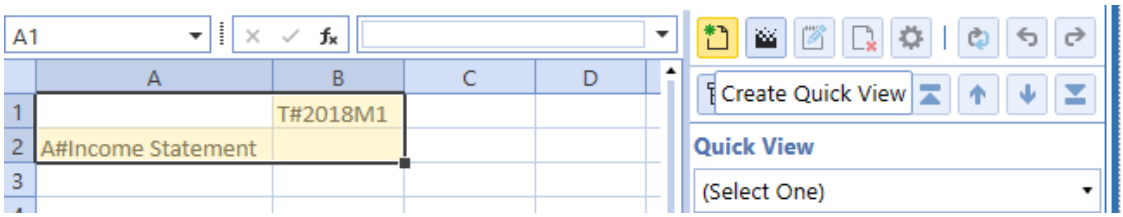
Create and Modify Quick Views

This topic describes how to define and modify quick views.

Create Quick Views

Create a quick view using a sample income statement account. See Find and Add Members to Quick Views and Manage Quick Views.

1. In Cell **A2** specify the row: **DimensionToken#MemberName A#IncomeStatement**.
2. In Cell **B1** specify the column: **DimensionToken#MemberName T#2018M1**.
3. Highlight the dimensions you entered and click **Create Quick View**. If you are using the Excel Add-in, click the **Quick Views** tab.



4. In the **Create Quick View** window, click **OK**.

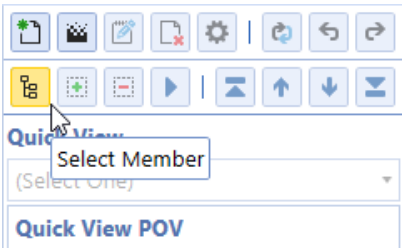
You can create quick views using the Quick Views menu on the OneStream ribbon.

1. On the OneStream ribbon, click the **Quick Views** drop-down menu.
2. Click **Create Quick View** and one will populate in the cells below.

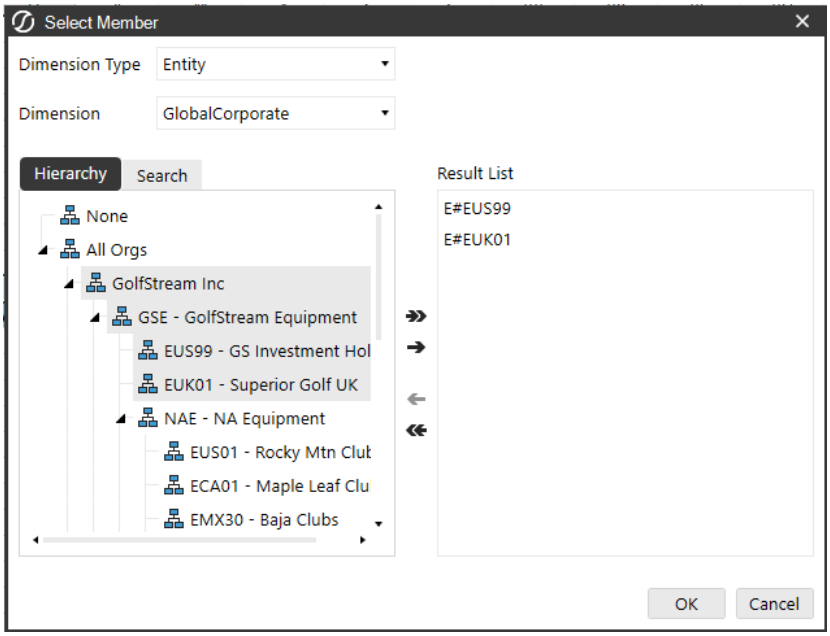
> **NOTE:** When Preserve Excel Format is enabled on Quick Views, structural changes such as drill down, will not adjust formatting automatically. Manual reformatting may be required when modifying Quick Views.

To add members to your new quick view, manually enter the dimension tokens into the cells or add them using the Quick View toolbar.

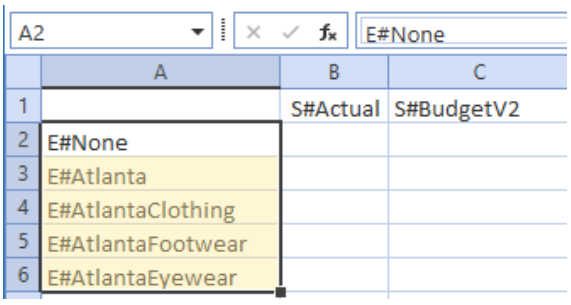
1. In cell **B1** specify the column: **DimensionToken# MemberName S#Actual**
2. In cell **C1** specify the column: **DimensionToken#Member Name S#BudgetV2**
3. Click on the cell where your quick view rows will begin. In this example, specify column **A2**.
4. Click **Select Member** from the quick view toolbar to the right of the screen.



5. From the Select Member dialog box, move dimensions to include in your quick view to the **Result List** using the single arrow. Click **OK**.

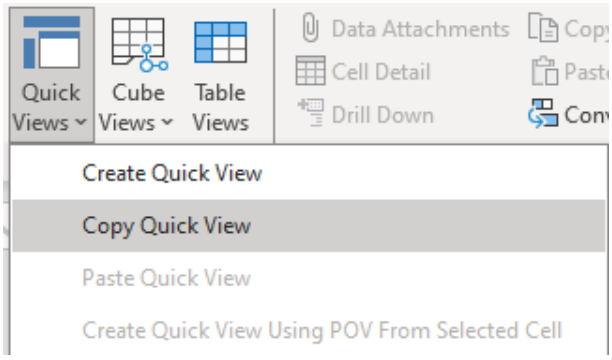


6. Highlight the cells with your data and click **Create Quick View**.



You can create a quick view by copying an existing quick view or expanding on the data in a quick view cell.

1. Select an existing quick view.
2. On the OneStream ribbon, click the **Quick Views** drop-down menu.
3. Select **Copy Quick View**.



4. Select any cell within the current sheet, workbook, or another workbook.

> **NOTE:** You can only paste a copied Quick View into the same workbook when using Spreadsheet.

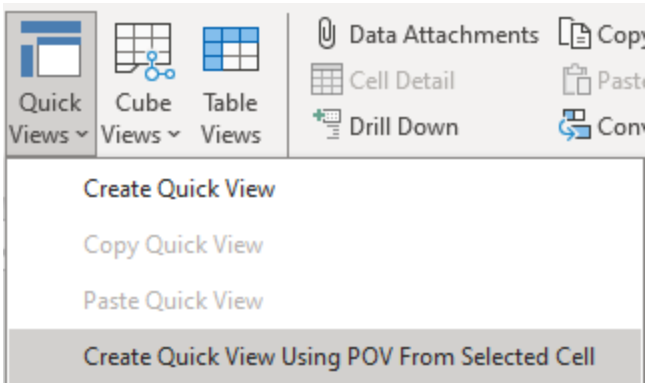
5. Click Quick Views drop-down menu and select Paste Quick View.

> **NOTE:** Paste any quick view by right-clicking > Quick View > Paste Quick View. See Right-Click Options.

To create a quick view based on the POV from a single cell in an existing quick view:

1. Select an intersection of data in an existing quick view.
2. On the OneStream ribbon, click the **Quick Views** drop-down menu.

3. Select **Create Quick View Using POV From Selected Cell**.



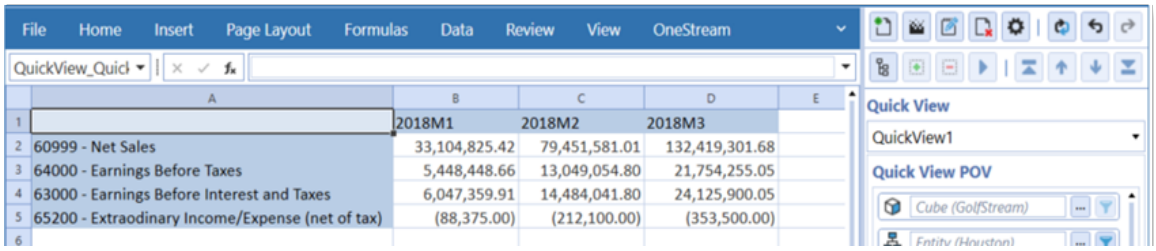
Modify Quick Views

The following section outlines how to rebuild quick views after adding or removing dimensions and members.

Use a comma-separated member list in column and row cells to build the quick view. In this example, enter:

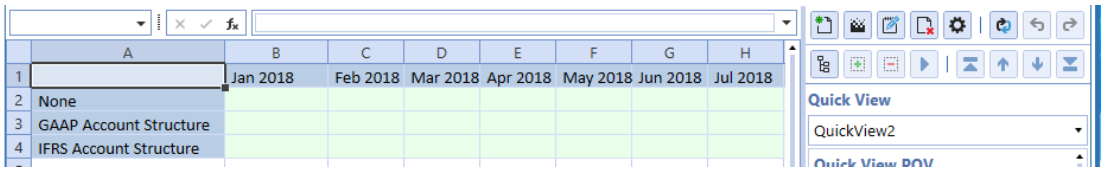
1. **DimensionToken#Root: A#60999,64000,63000,65200** in **A2**
2. **DimensionToken#Root: T#2018M1,2018M2,2018M3** in **B1**
3. Select the updated area and click **Create Quick View**.
4. Enter a quick view name and click **OK**.

The quick view is displayed:



Use the member expansion functions in cells to refine the quick view. For example, enter:

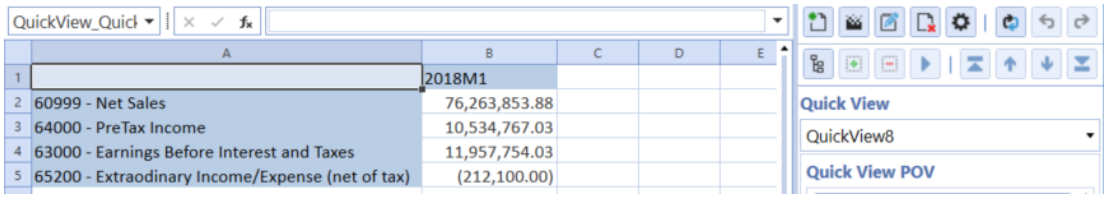
1. **DimensionToken#Root A#Root.Children** in **A2**
2. **DimensionToken#Root T#2018.Base** in **B1**
3. Select the updated area, click **Create Quick View** and specify a name.



Use the member expansion functions to enter comma-separated members in cells to finalize the quick view. For example, enter:

1. **DimensionToken#Root A#Root.List(60999,64000,63000,65200)** in **A2**
2. **DimensionToken#Root T#2018M1** in **B1**
3. Select the updated area and click **Create Quick View**.

The quick view updates:



NOTE: Check the suppression options if items do not display as expected.

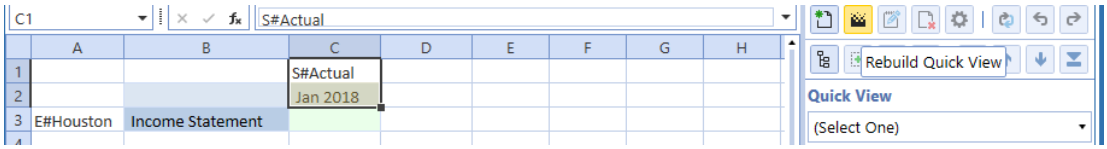
Extend a Quick View

To add a dimension or new dimension member to a quick view, enter **#** before the specified dimension or member names. If the dimension is already used, **#** is not required.

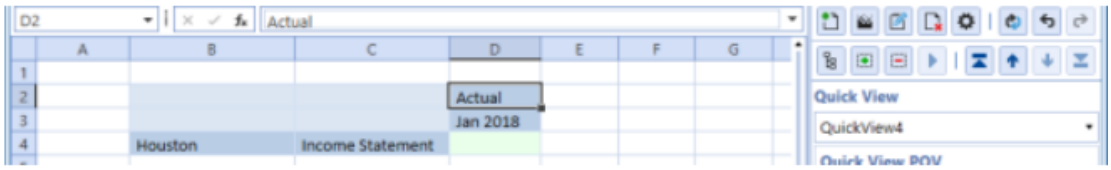
1. Specify the data to use. For example, enter:

   Dimension type, such as **DimensionToken#MemberName S#Actual**

2. Select the area and click **Rebuild Quick View**.



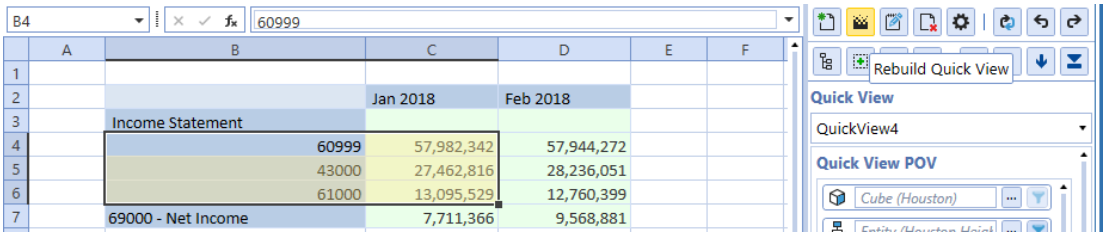
The updated quick view is displayed:



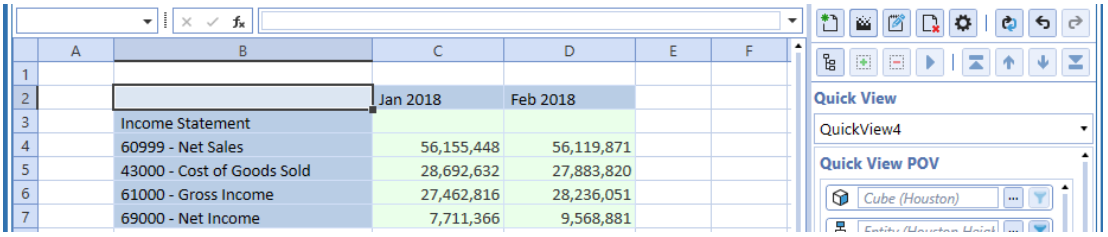
Change Dimension Members

You can switch dimension members by entering another member name. You do not need to specify a dimension token if you switch to members in a dimension that is being used. For example:

1. Enter these members: **60999**, **43000**, and **61000** .
2. Select the members and click **Rebuild Quick View**.

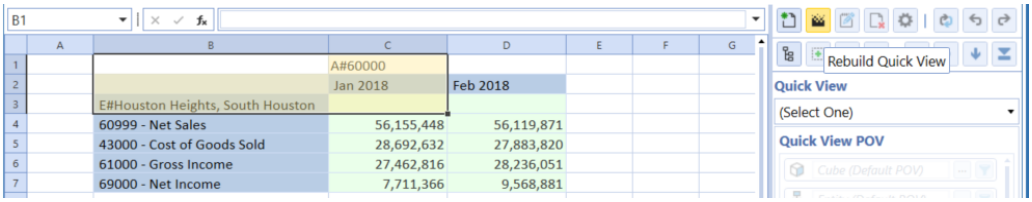


Members **60999**, **43000**, and **61000** replace **60000** and **54500** as shown in the updated quick view:



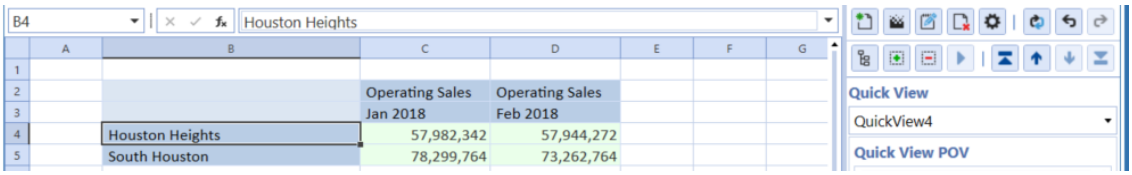
You can change from one dimension to another by entering comma-separated members in the same cell. In the example below, the Account dimension is changed to the Entity dimension.

1. Above the **Time** dimension enter the following, separated by commas:
    - **E#HoustonHeights**
    - **South Houston**
    - **Operating Sales A#60000**



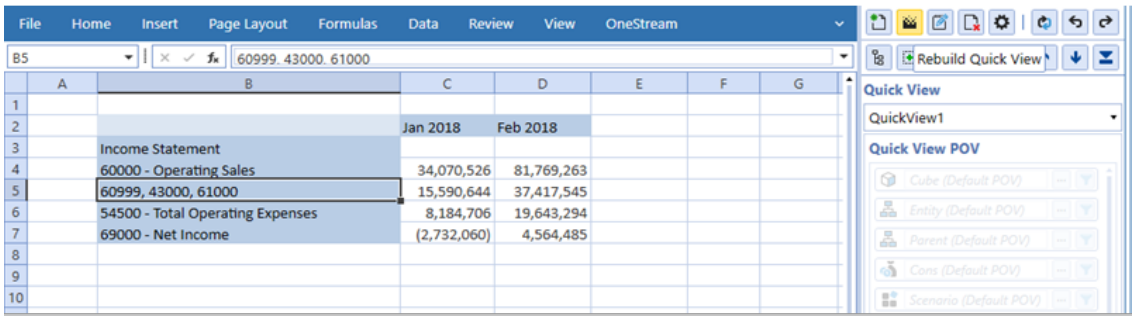
2. Select the area and click **Rebuild Quick View**.

    Operating Sales columns are created for the Houston entities:

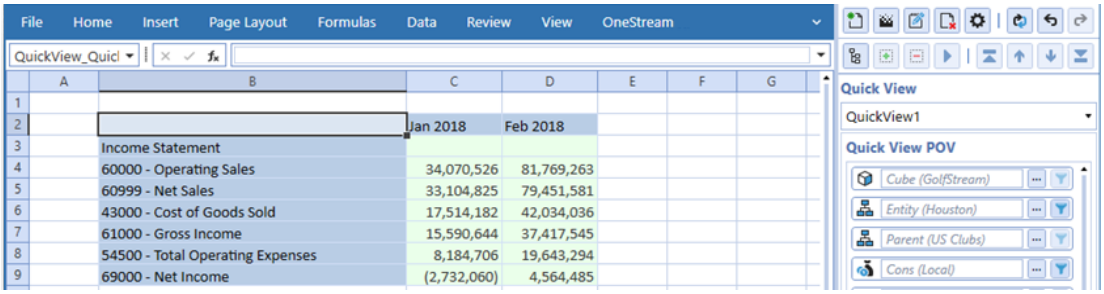


You can add to an existing dimension's members by entering comma-separated member names in the same cell. If names are numeric or start with an apostrophe, insert a space after each comma.

1. In the same cell, enter the following members. Insert a space after each comma because the member names are numeric.
    - **60999**
    - **43000**
    - **61000**
2. Select the area and click **Rebuild Quick View**.



Members 60999, 43000, and 61000 are added to the existing member list:

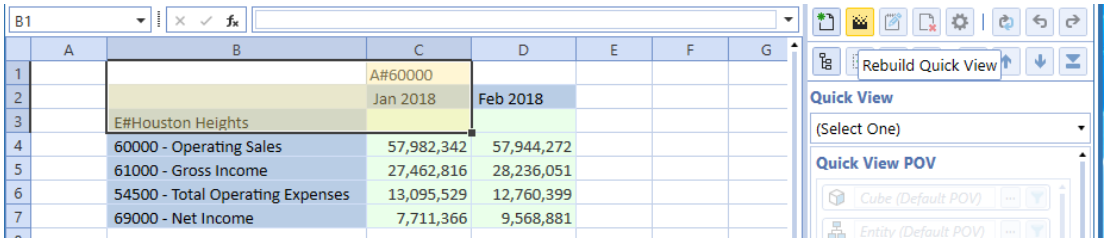


Change Dimensions

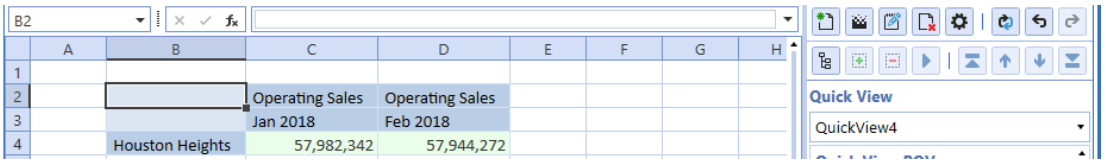
Dimensions can be replaced by modifying row and column headers with the updated dimension information and rebuilding the quick view.

1. Click the dimension to be changed, then specify the dimension to use. As shown in the example below, you could:
    a. Replace **Account** with an **Entity** such as **E#HoustonHeights**.
    b. Add **OperatingSales A#60000** above **Time**.

2. Select the area and click **Rebuild Quick View**.



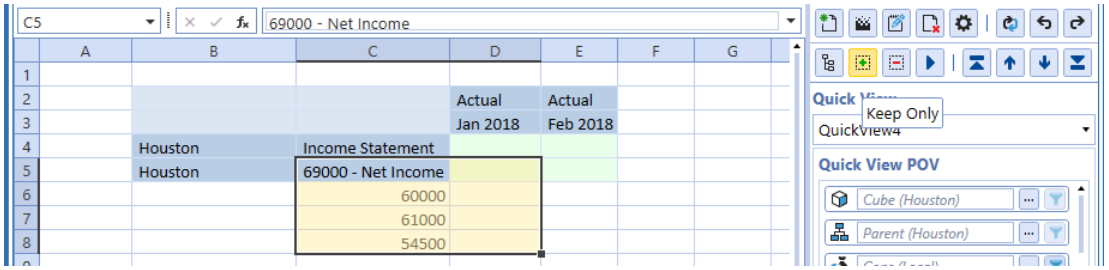
The updated quick view will look like this:



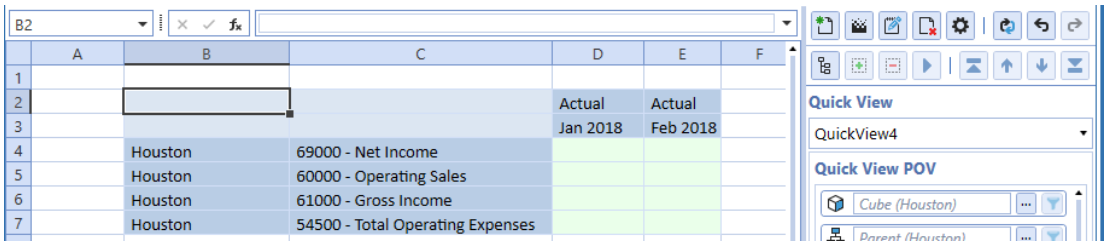
Using "Keep Only" When Extending a Quick View

When modifying a quick view, select the rows and columns you want to remain unchanged using the **Keep Only** option. This allows you to clear a quick view without losing selected data.

1. In the row below an existing dimension, add these members: **60000**, **61000**, and **54500**.
2. Select the area and click **Keep Only**.



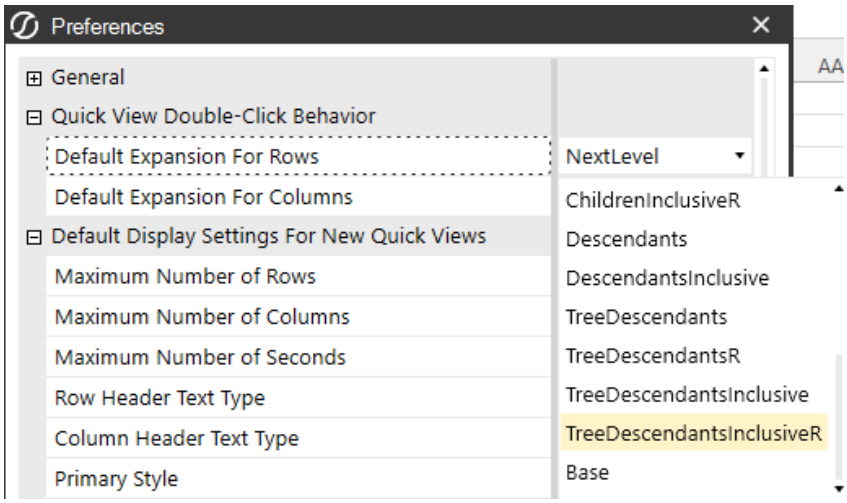
After updating, the quick view will look like this:



Reverse Member Display Order

You can double-click to reverse the display order of members by specifying **Quick View Double-Click** preferences.

1. Select **Administration** > **Preferences**.
2. In **Quick View Double-Click**, set **Default Expansion for Rows** to **TreeDescendantsR**.



Format and Suppress Data

Use the following to manage and customize your quick view display:

- **Row Header/Columns Header Text Types**: Reformat columns and rows.
- **Suppress Repeating Member Names**: Display or hide repeating member names. For example, if False, the repeating name displays in each row.
- **Data Style**: Apply a provided or custom data style. See Styles.
- **Suppress Invalid Rows/Columns**: Hide cells that have invalid data.
- **Zero Suppression Threshold**: Suppress all values that are less than a number you specify, for rounding purposes. For example, specifying 499.99 suppresses any value less than that, such as 499.19.

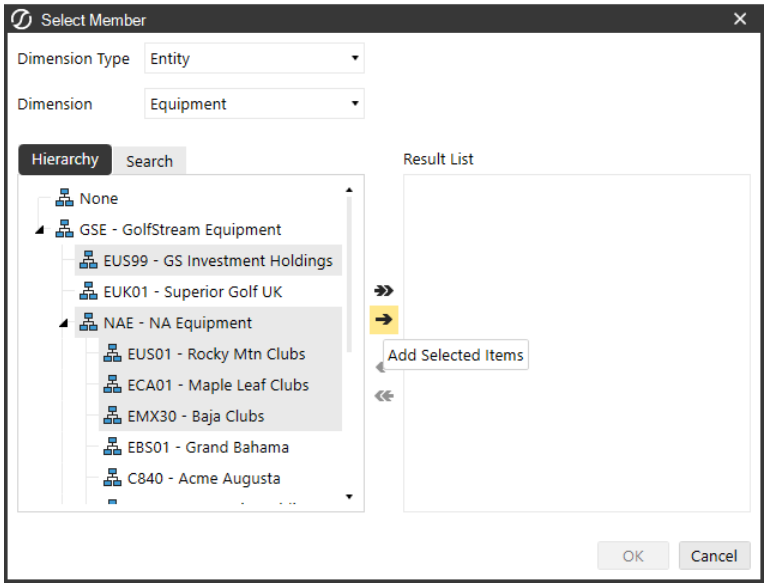
Find and Add Members to Quick Views

You can add multiple members to a quick view, filtering and searching as necessary.

> **NOTE:** To add members while preserving the current quick view column, select the columns to fill outside the current quick view range. After adding members, select the new column range and click **Rebuild Quick View**.

1. Select a dimension, then click **Select Member**.
2. On the **Select Members** dialog box, perform a task:
    - To find members not shown in the hierarchy, click the **Search** tab, enter the member name, select members, and then click **Add Selected Items**.
    - To select multiple members, press **Ctrl** or **Shift**, selecting the members, and then

click **Add Selected Items**.



3. Select the area to which the members were added and click **Rebuild Quick View**.

Table Views

A Table View definition for the Windows Application Spreadsheet Tool is defined in a Business Rule. The Administrator designing the rule can define the rows and columns which should be returned to the Spreadsheet from the source table presented in the Table View.

The Table View Business Rule can collect data from multiple data sources. For example, a single Spreadsheet worksheet can display a Table View which collects data from two or more sources.

The Administrator has full control over the write back “save” process through Business Rules. When designing the Table View Business Rule, the BRAPI Authorization functions should be designed into the Business Rule to control access to the viewing or modifying the data. This can be applied to the entire table or to specific cells. A workbook can contain multiple Table Views. These can be on the same worksheet or across worksheet pages.

A single Business Rule file can be used to define multiple Table Views by calling the Business Rule argument, TableViewName. Additionally, a single named range can be used to manage table data cells within the Spreadsheet using user defined named ranges (XFTV_*).

> **NOTE:** When browsing for Table Views Business Rule, only Spreadsheet Type Rules will be displayed in the Object Lookup dialog.

Table Views in Excel

Insert a Table View into Excel.

1. Click Table Views to open the Table View Definition dialog box.
2. From this window, manage the Table Views within Excel. Add, remove, edit, and refresh tables. Click the Add button to add a new Table View.

3. In the Table View Definition dialog box, populate the following fields:

a. **Name**: Table View name.

b. **Refers To**: Cell range where the Table View will exist.

c. **Table View Business Rule**: Click the ellipses to search for and select a business rule.

d. **Insert or Delete Rows When Resizing Table View Content**: Select this option if you plan to stack everything vertically on your worksheet. This automatically adds or deletes rows as a table view expands and contracts.

e. **Insert or Delete Columns When Resizing Table View Content**: Select this option if you plan to stack everything horizontally on your worksheet. This automatically adds or deletes columns as a cube view expands and contracts.

f. **Preserve Excel Format**: This option enables you to preserve native Excel formatting changes made to cube views, quick views, and table views. When enabled, formatting changes made via native Excel formatting will be retained. See Preserve Formatting.

4. Click the **OK** button, then the **Close** button to view the Table View in the Excel worksheet.

Table Views Spreadsheet Control

Table Views is a OneStream Windows Application only spreadsheet control. The primary purpose of Table Views is to provide an easier method to creating Dashboards where relational data is required. The use of Table Views in Spreadsheet enables the designer to work in a more flexible environment to design a form or data collection tool.

Accessed via the OneStream Windows Application Spreadsheet, the Table Views provide a client-based tool to support Dashboard forms. Table Views are not intended as an alternative to other tools, such as the SQL Table Editor or Grid Viewer, Dashboard Components.

Key Use

- Tool kit item for advanced OneStream Solution or dashboard designers
- Designed to collect records from relational tables, or other sources
- Present the information in the Spreadsheet format
- Utilize client-side functionality, found in the Spreadsheet tool, such as calculations and pick-list validation lists
- Update table records only

Design Considerations

- The current functionality is designed to update records in target tables
- Does not perform inserts, to create new records
- Controlling elements must be designed into the Table View Business Rule by the creator to ensure data integrity, security and performance

Table View Size Considerations

- Spreadsheet support of Table Views depends upon the number of rows and row content
- The Spreadsheet Control does not support paging, therefore all rows and content must be returned
- Performance testing and design expectations is to support approximately 8000 KB of data per Table View.

Parameters

Generate a drop-down menu in a Cube View or Table View column based on selected parameters in the Excel Add-In or Spreadsheet to quickly filter your data. This functionality eliminates the need to manually create drop-down menus. When a parameter in the drop-down menu is changed, the worksheet will refresh to show the updated output. Parameters that are dependent on one another will updated automatically when the related parameter is altered.

> **NOTE:** See the *Table View Business Rule Example* in the *Table Views Guide* to set up your business rule code to generate a nested parameter in a column in a Table View.

Supported Parameters

The following types of parameters are supported:

- Literal
- Input
- Delimited
- Bound
- Member

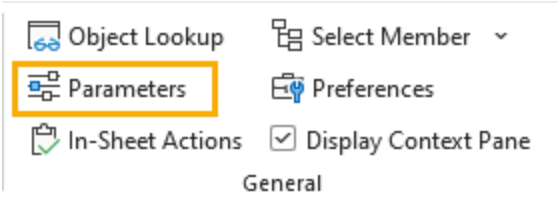
Unsupported Parameters

The member dialog parameter is not supported.

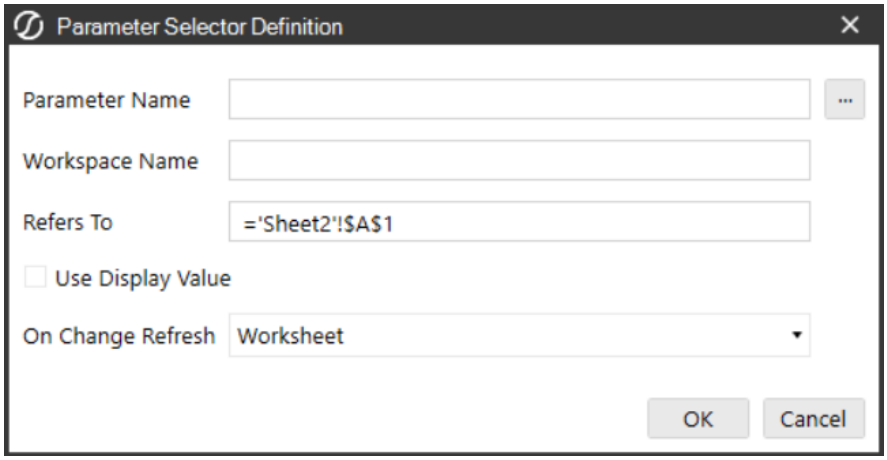
Create a Parameter

Follow these steps to create a parameter driven drop-down menu for a Cube View:

1. Select the cell to display the drop-down menu.
2. In the Excel ribbon, navigate to **OneStream** > **General** > **Parameters**.



3. In the **Parameters** dialog box, click the **Add** button.
4. In the **Parameter Selector Definition** dialog box, populate the following fields:



- **Parameter Name**: Click the ellipsis to search for and select the parameter. The name is automatically populated based on the selected parameter.

  > **NOTE:** It is best practice when creating parameters that the names be unique, even across Workspaces. See *Best Practices* in *Create a Parameter* in the *Design and Reference Guide*.

- **Workspace Name**: Name of the application workspace of the parameter. This field automatically populates when a parameter is selected.
- **Refers To**: Cell or range that will display the button.
- **Use Display Value**: The Use Display Value setting enables you to choose which item to use in your drop-down menus. In **Application** > **Workspaces** > **[...]** > **Parameters**, you can configure Display and Value Items.

  For example, if you configure the following settings:

  - Display Items: MI, OH
  - Value Items: Michigan, OH

  Then, if Use Display Value is selected, the Display Items (MI and OH) will display in the drop-down menu. If it is clear, the Value Items (Michigan and OH) will display.
- **On Change Refresh**: When you switch parameters, refresh the worksheet or workbook. Select **Nothing** to keep the sheet as is.

5. Click the **OK** button, then click the **Close** button on the **Parameters** dialog box.

Parameter Best Practices

- When working with parameters in Excel and Spreadsheet, remember that nested parameters and input parameters need to be consumed by a Cube View or Table View. Using nested, standalone parameters may generate undesirable results.
- When adding dependent parameters to a worksheet manually, remember that parameters must be added in the order in which their sequence must be resolved. When using Add Parameter Selectors to Sheet, the parameters will automatically be added in their sequential orde.r

> **Example:** Year must be added before Month. Region must be added before Sub-Region.

- For dependent parameters to refresh correctly when the parent parameter is changed, the Workspace containing the Cube View or dependent parameter within the Spreadsheet or the Excel Add-In must be sharable. To enable this, navigate to **Application** > **Workspaces** > **Sharing** and ensure Is Sharable Workspace is set to True.

In-Sheet Actions

In-sheet actions enable you to quickly interact with your data directly in Excel and Spreadsheet. Creating a button streamlines the following actions:

- Refreshing a worksheet or workbook
- Submitting a worksheet or workbook
- Running a data management sequence

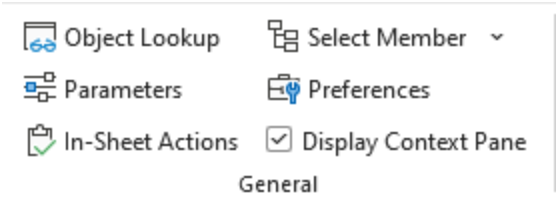
Create In-Sheet Button

Build an In-Sheet Action button using these steps:

1. Navigate to the Excel worksheet you would like to place your In-Sheet Action.
2. Select a cell.

   > **NOTE:** Make any In-Sheet Action button larger by selecting multiple cells. If a button consists of more than one cell, when opened in Spreadsheet, the button will revert to a single cell.

3. From the ribbon, select **OneStream** > **General** > **In-Sheet Actions**.



4. In the **In-Sheet Actions** dialog, click the **Add** button.
5. In the next **In-Sheet Actions** dialog, populate any applicable fields:
    a. **Label**: Button text to show in cell.
    b. **Refers To**: Cell or range where the button will appear.
    c. **Submit**: Submits the Worksheet or Workbook after clicking the button. Select **Nothing** from the drop-down menu if you do not want to make a submission.
    d. **Data Management Sequence**: Use your In-Sheet action button to run a data management job. Click the ellipsis to browse for and select a data management sequence.
    e. **Workspace Name**: Populates with the name of the Workspace after you select a data management sequence.
    f. **Parameters**: Designate explicit values for parameters needed for your data management sequence or leave blank to resolve parameters based on sheet selections. Enter the parameter name and the value it should be set to at run time, for example, myparam = value. When adding multiple parameters, separate each parameter with a comma. For example, myparam=value1, myparam2=value2. If the value for your parameter contains a space, use brackets in your syntax. For example, myparam=[South Houston].

> **IMPORTANT:** Parameters can only be used if they have been resolved in an existing Cube View or Table View in the sheet or workbook. To use parameters that don't exist in your Cube View or Table View, add it anywhere in the Cube View in OneStream, for example in the page caption field or other unused field that does not impact Cube View rendering. For Table Views, add the parameter to the GetCustomSubsVars section of the Spreadsheet business rule.

> **NOTE:** In-sheet actions will not prompt for unresolved parameters.

g. **Do not wait for task to finish**: Enable this setting to continue working while the data management sequence runs in the background.

h. **Refresh**: Refreshes the Worksheet or Workbook after clicking the button. Select **Nothing** from the drop-down menu if you do not want to make a submission.

i. **Text Color**: Choose a custom button text color and input an eight character hex code, or select a color from the drop-down menu.

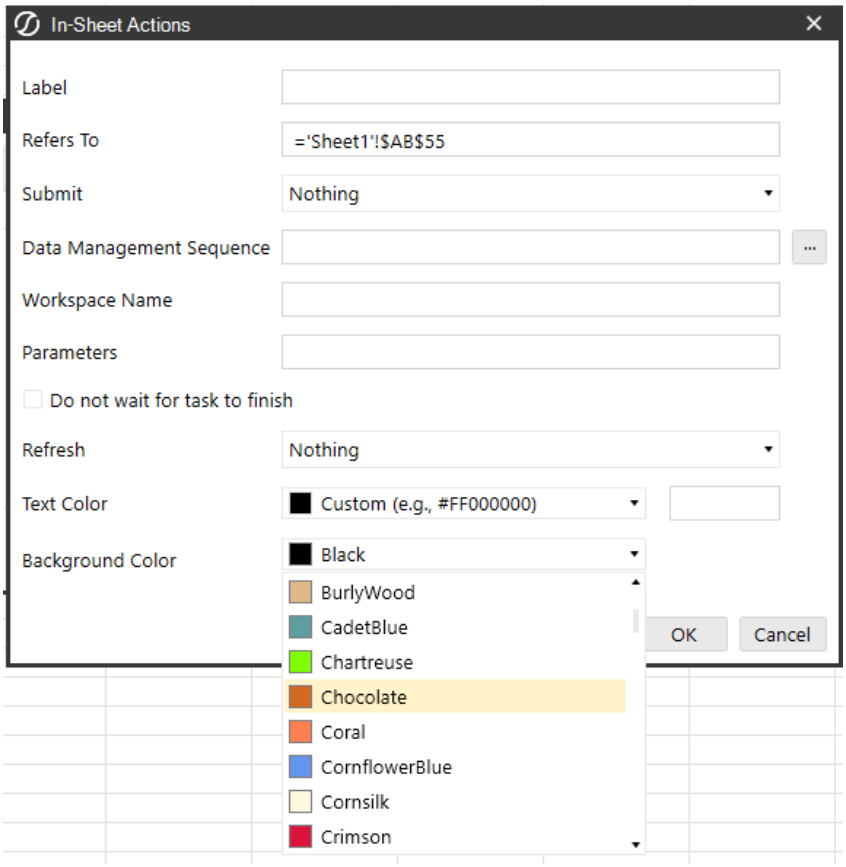
j. **Background color**: Choose a custom button background color and input an eight character hex code, or select a color from the drop-down menu.

6. Click the **OK** button and then the **Close** button. Your button will appear in the cells you selected.

Format In-Sheet Actions

By default, In-Sheet Action buttons will display as black with white text. After initial creation, customize the display of your buttons by formatting them with custom colors and styles. Modify your button using the steps below:

1. From the ribbon, select **OneStream** > **General** > **In-Sheet Actions**.
2. In the **In-Sheet Actions** dialog box, select the button to format and click the **Edit** button.
3. Use the **Button Theme** drop-down menu to select a color.



4. Click **OK** and then **Close** to save changes.

Right-Click Options

When working with data in the in Excel Add-In, right-click and navigate to OneStream to see the following available actions:

Cube Views

Cube View Connections

Create a new Cube View connection.

Selection Styles

Set up a selection style. See Selection Styles.

Manage Selection Styles

Modify an existing selection style. See Selection Styles.

Quick View

For a complete overview on Quick Views, see Quick Views.

Create Quick View

This will create a new Quick View in the worksheet's selected cell.

Copy Quick View

This will copy the selected Quick View in order to paste a version of it in another spreadsheet or workbook.

Paste Quick View

This will paste the copied quick view to the current spreadsheet or another workbook.P

Create Quick View Using POV from Selected Cell

This will create a new Quick View based on the current POV from the selected cell. This can be done using a Quick View cell's POV, a Cube View cell's POV, or a XFGetCell's POV. For information on creating a Quick View using a POV from the Drill Down page, see Drill Down in "Using OnePlace Cube Views" in the *Design and Reference Guide*.

Expand

Select the cell of a Member and choose how to view its data. Select from these options:

- **AllTops**: This returns the Top of the given Dimension.
- **AllBase**

  This returns all Base Members of the given Dimension regardless of what Member is selected.
- **All**

  This returns all Members in a given Dimension.
- **NextLevel**

  This returns the next level of Members under the selected Member.
- **KeepOnly**

  This will only keep the selected Members.
- **Parents**

  This returns the direct Parents of the selected Member regardless of how many hierarchies to which the Member belongs.
- **Ancestors**

  This returns all Members up the chain from the selected Member.
- **Children**

  This returns the first level of Children under the selected Member.
- **ChildrenInclusive**

  This returns the selected Member and its first level of Children.
- **Descendants**

  This returns every Member under the selected Member in a list, not a hierarchy.
- **DescendantsInclusive**

  This returns the selected Member and every Member under it in a list, not a hierarchy.
- **TreeDescendants**

  This returns every Member under the selected Member in a hierarchy.

- **TreeDescendantsInclusive**

  This returns the selected Member and every Member under it in a hierarchy.

- **Base**

  This returns the Base level for the selected Member.

Paste POV

Enables you to Paste a POV into a selected cell in order to change the data within that Quick View.

Apply POV from Selected Cell

This enables a POV to be passed from a selected cell within a Quick View.

Apply User POV

This enables a Point of View to be passed between Quick Views.

Clear POV

This clears the POV for the selected Quick View.

For the following properties, see Manage Quick Views

- Undo
- Redo
- Options
- Refresh

General OneStream Right-Click Options

Calculate, Translate, Consolidate

If permitted, these calculations can be performed on the selected cell. These function the same as the icons in the Ribbon, but you have the choice to do Force operations and additional Logging.

Select Member

Select a Dimension Type from the drop down list in order to view the Members of that Dimension. Select a Member of the hierarchy, and the Member name will display in the selected cell.

Copy POV from Data Cell

This captures the Point of View of the currently selected cell. After clicking this option, the Paste POV As XFGetCell becomes available.

Paste POV as XFGetCell

This option is only available after clicking Copy POV From Data Cell. After clicking this option, OneStream converts the copied cell into an XFGetCell formula.

Convert to XFGetCells

This converts an existing Quick View into an XFGetCells.

Allocate

See Allocation in *Using OnePlace Cube Views* in the *Design and Reference Guide*.

Data Attachments

This opens the Data Attachments dialog so you can view existing comments or attachments on the selected cell, or to edit data attachments. See Data Attachments for Selected Cell/Data Unit in *Using OnePlace Cube Views* in the *Design and Reference Guide*.

Cell Detail

See Cell Detail in *Using OnePlace Cube Views* in the *Design and Reference Guide*.

Cell POV Information

This gives a detailed summary of the selected Members related to this intersection as well as the full Member Script used to get this value. All the major properties of these Members can be seen from this dialog.

Cell Status

This returns a long list of properties about a given cell.

Navigate
Enables you to launch a cube view or dashboard from Excel when configured. See Linked Cube Views and Linked Dashboards in the *Design and Reference Guide*.

Drill Down
See Drill Down in *Using OnePlace Cube Views* in the *Design and Reference Guide*.

Spreading
This enables users to enter data into an aggregate Member, like an annual time period, and spread values over several columns or rows without having to type in each cell's values. Select from the following options:

- Spread Selected Cells: See Spreading Types.
- Flag Selected Cells: Flags selected cells so the original amount in the cell is retained during the spreading process.
- Clear Flags: Select this to clear any flagged cells.

Retrieve Functions

Retrieving and changing data can be done by using functions. To see the functions and their Parameters, open Excel and select the Formulas tab. Select Insert Function and select OneStreamExcelAddIn.XFFunctions where it says to Select a category.

The output of the function will look something like this: =XFGetCell(A1,A2,A3,A4,A5)

The equivalent functions like XFGetCell provide a separate Parameter to specify each Dimension Member without using the Member Script syntax. (e.g., E#CT:A#Sales would not be used, CT and Sales would be used in the correct Parameter for that Dimension)

Here are the main functions to use:

> **NOTE:** If a field within the function is not needed, enter a double quote to ignore it.

XFGetCell

This function retrieves data based on the Parameters supplied. Each Parameter needs to be defined.

Example:

XFGetCell(NoDataAs Zero, Cube, Entity, Parent, Cons, Scenario, Time, View, Account, Flow, Origin, IC, UD1, UD2, UD3, UD4, UD5, UD6, UD7, UD8)

XFGetCell5

This has the same functionality as XFGetCell except it limits the User Defined Dimensions to five instead of eight.

XFGetFXRate

This function retrieves rates from the system. Each Parameter needs to be defined.
XFGetFXRate(DisplayNoDataAsZero, FXRateType, Time, SourceCurrency, DestCurrency)

XFGetCalculatedFxRate

This function directly retrieves an exchange rate even if only the inverse rate exists in the system.

XFGetMemberProperty

This function retrieves any Dimension Member property from the Member Properties tab in the Dimension Library. Note there are no spaces used when defining property name.

Example:

XFGetMemberProperty(“DimTypeName”,“MemberName or Script”,“PropertyName”, “VaryByCubeTypeName”,“VaryByScenarioTypeName”,“VaryByTimeName”)

> **NOTE:** If the function does not need to vary by Cube Type, Scenario, or Time, enter a double quote in order to ignore it.

Example: Retrieving Currency for the Houston Entity
XFGetMemberProperty(“Entity”,”Houston”,”Currency”,””,””,””)

Example: Retrieving an Account Formula that only occurs in the Budget Scenario
XFGetMemberProperty(“Account”,”51000”,”Formula”,””,”Budget”,””)

Example: Retrieving the Short Description Property for Time Dimension
XFGetMemberProperty("Time","2015M8","ShortDescription","","","")

XFGetRelationshipProperty

This function retrieves any Dimension relationship property from the Relationship Properties tab in the Dimension Library.
XFGetRelationshipProperty(“DimTypeName”,“ParentMemberName or Script”,“ChildMemberName or Script”,“PropertyName”,“VaryByScenarioTypeName”,“VaryByTimeName”)

> **NOTE:** If the function does not need to vary by Cube, Scenario, or Time, enter a double quote in order to ignore it.

Example: Retrieving a Flow Members Aggregation Weight
XFGetRelationshipProperty(“Flow”,”TotalBalance”,”Total Movement”,”AggregationWeight”,””,””)

Example: Retrieving an Entity’s Percent Consolidation for July, 2015
XFGetRelationshipProperty(“Entity”,”Houston”,”South Houston”,”PercentConsolidation”,””,”2015M7”)

XFGetHierarchyProperty

This function determines whether or not a Dimension has children and returns True or False.

Example:

XFGetHierarchyProperty(“DimTypeName”,”DimName”,“MemberName or Script”,“PropertyName”,”PrimaryCubeName”,”ScenarioTypeNameForMembers”,”MergeMembersfromReferencedCubes”)

Example: Retrieving Child Hierarchy

XFGetHierarchyProperty("entity","HoustonEntities","Houston Heights","HasChildren","Houston","Actual",FALSE)

XFGetDashboardParameterValue

This function is available to Excel Add-in and Spreadsheet. If that function is used within an XLSX file that is using a function like XFGetCell or XFSetCell (or similar) where these are referencing a custom parameter value (e.g. ParamEntity) that is on the Dashboard that references this Spreadsheet from within it as a Component. The practice to get this Custom Parameter value is to use XFGetDashboardParameterValue to fetch the text from that Parameter or its default value and place it in a cell on the Spreadsheet (e.g. B1). Then the cell that is using a retrieve function such as XFGetCell would reference this other cell (i.e. B1).

XFGetDashboardParameterValue("myParamName", "Text For Default Value")

XFGetMemberInfo

This function retrieves the description in the system. Each Parameter needs to be defined.
XFGetMemberInfo(MemberInfoType, DimTypeName, MemberName, NameorDesc, NameandDesc)

XFInternalGetDataFromServer

This function returns True or False. It does not take any arguments.

XFSetCell

This function saves data to the amount field based on the Parameters supplied. Each Parameter needs to be defined.
XFSetCell(CellValue, StoreZeroAsNoData, Cube, Entity, Parent, Cons, Scenario, Time, View, Account, Flow, Origin, IC, UD1, UD2, UD3, UD4, UD5, UD6, UD7, UD8

XFSetFXRate

This function saves rates to the system. Each Parameter needs to be defined.
XFSetFXRate(Value, StoreZeroAsNoData, FXRateType, Time, SourceCurrency,DestinationCurrency)

XFGetCellUsingScript

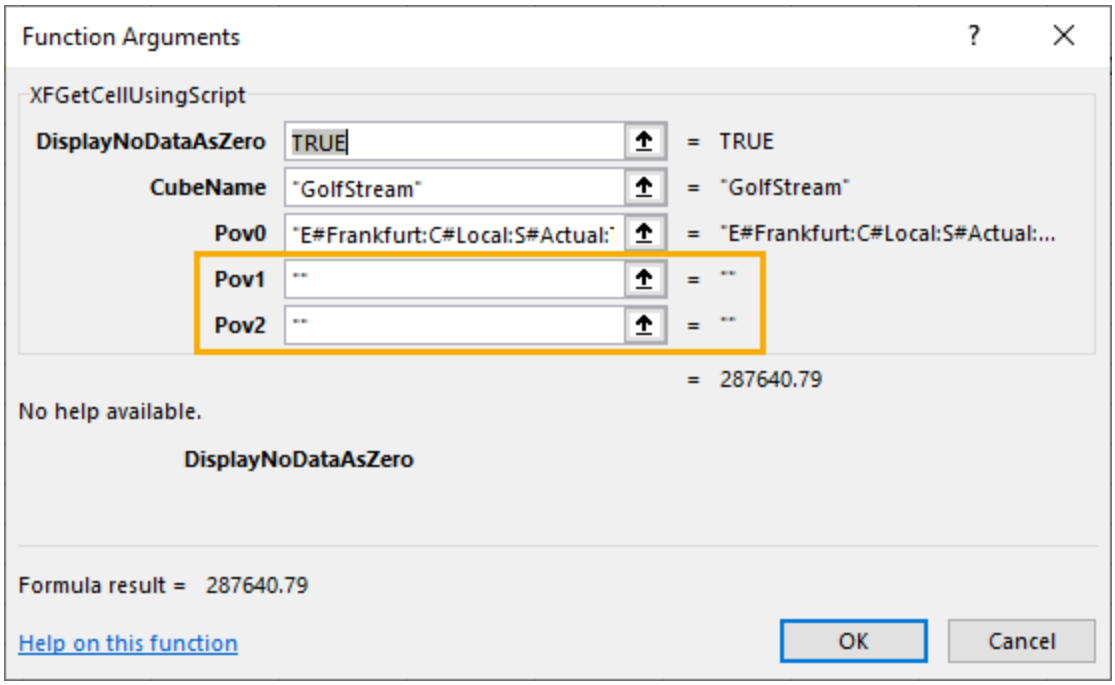
XFGetMemberInfoUsingScript

XFSetCellUsingScript

All of the functions that have UsingScript are based on a Member Script (e.g., A#Sales:E#Texas). The multiple Parameters provide the ability to specify multiple portions of the full Member Script using different Excel cells. All of the Member Scripts in the function Parameters combine to create one Member Script. It will then use the combined Member Script to retrieve the data cell.

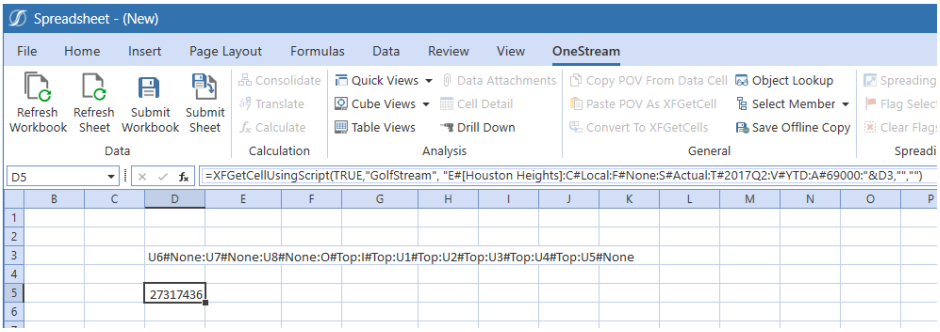
Example:=XFGetCellUsingScript

(TRUE,"GolfStream","E#Frankfurt:C#Local:S#Actual:T#2022M1:V#YTD:A#10100:F#None:O#Top:I#Top:U1#Top:U2#Top:U3#Top:U4#Top:U5#None:U6#None:U7#None:U8#None","","")



NOTE: Add quotation marks to POVs not being used.

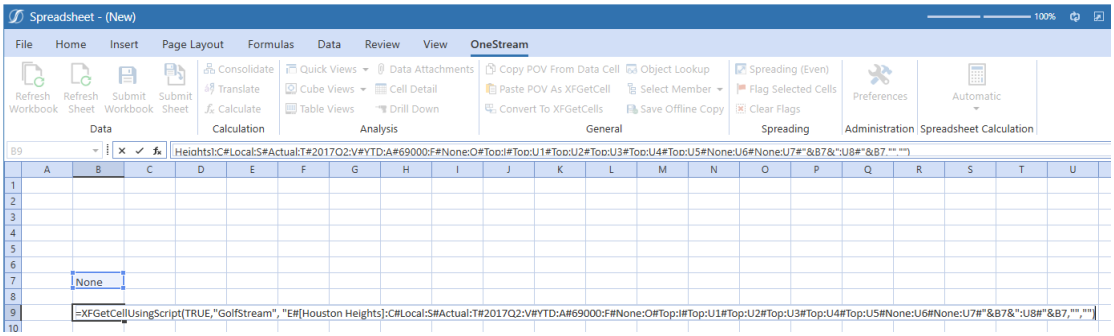
The following examples detail uses of XFGetCellUsingScript.



Example:

=XFGetCellUsingScript(TRUE,"GolfStream", "E#[Houston Heights]:C#Local:F#None:S#Actual:T#2017Q2:V#YTD:A#69000:"&D3,"","")

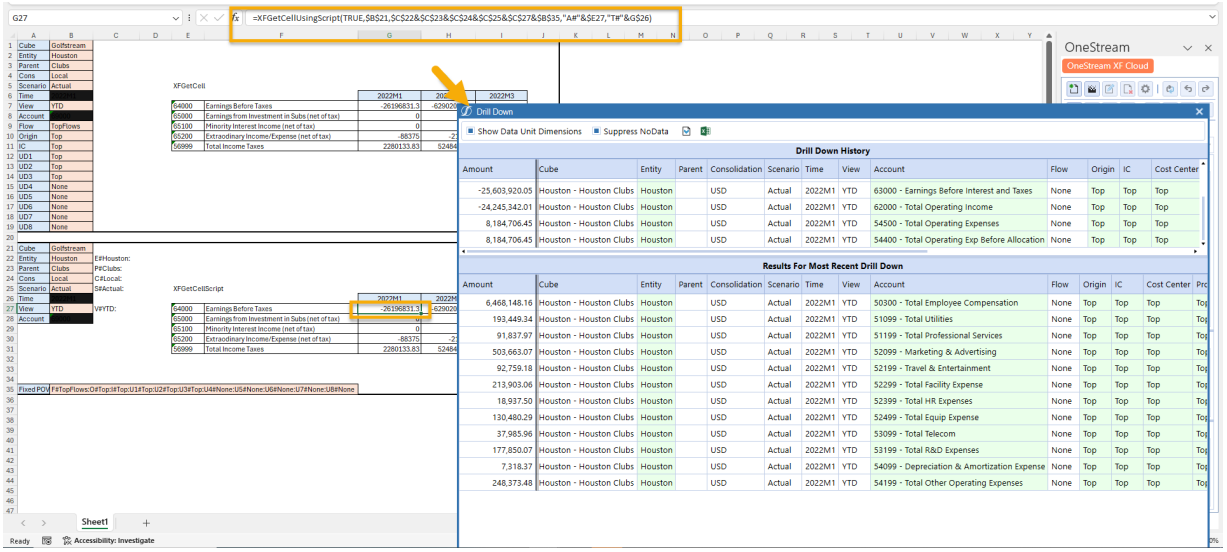
D3= “U6#None:U7#None:U8#None:O#Top:I#Top:U1#Top:U2#Top:U3#Top:U4#Top:U5#None”



Example:

=XFGetCellUsingScript(TRUE,"GolfStream", "E#[Houston Heights]:C#Local:S#Actual:T#2017Q2:V#YTD:A#69000:F#None:O#Top:I#Top:U1#Top:U2#Top:U3#Top:U4#Top:U5#None:U6#None:U7#"&B7&":U8#"&B7,"","")

B7=”None”



Example:

=XFGetCellUsingScript

(TRUE,B21,C22&C23&C24&C25&C27&B35,"A#"&$E27,"T#"&G$26)

XFGetCellUsingScriptEx

XFGetMemberInfoUsingScriptEx

XFSetCellUsingScriptEx

All of the functions that have …Ex have many more Parameters to use for combining Member Scripts (e.g., commonly used when creating another version of a function that has extra Parameters). Ex would also be used to combine many Member Script Parameters.

XFSetCellLocalForms

XFGetCellLocalAdjInput5

XFGetCellLocalForms5

XFGetCellLocalImport5

XFGetCellLocalOTop5

XFGetCellTransAdjInput5

XFGetCellTranForms5

XFGetCellTransImport5

XFGetCellTransOTop5

XFSetCellLocalForms5

These functions use the Consolidation and Origin Dimensions. For example, XFSetCellLocalForms is using Local Consolidation and the Forms Origin Member. The number five at the end of the functions limit the User Defined Dimensions to five instead of eight.

XFGetCellUSingScriptVolatile

XFGetCellVolatile

XFGetFXRateVolatile

XFGetMemberInfoUsingScriptExVolatile

XFGetMemberInfoUsingScriptVolatile

XFGetMemberInfoVolatile

XFSetCellUsingScriptExVolatile

XFSetCellUsingScriptVolatile

XFSetCellVolatile

XFSetFXRateVolatile

In some cases, Excel requires a volatile function for proper refreshing, for example, some Excel Charts that reference calculated cells.

XFInternalPrepareCalculationStep

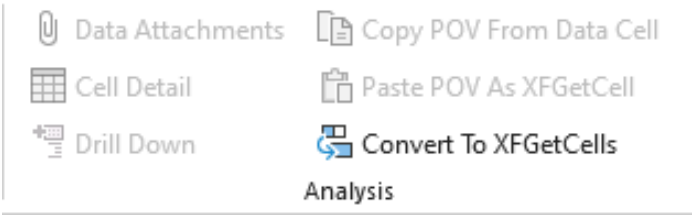
XFInternalSendDatatoServer

XFInternaSetConnectionInfo

All of the functions that begin with XFInternal only work for internal processes.

XF Retrieves

In an existing quick view, you can convert to formula-based retrieves (XFGetCell) on intersections of data using the **Convert To XFGetCells** option in the top ribbon toolbar.



After selecting **Convert To XFGetCells**, OneStream displays the following prompts:

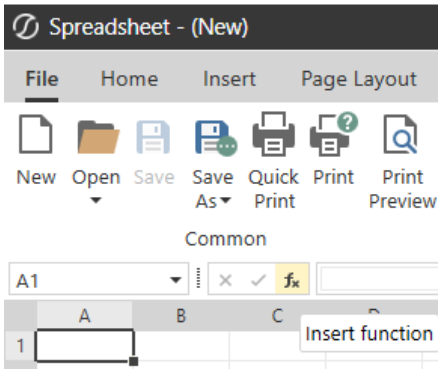
- Are you sure you want to convert all the data in Quick View 'Name of the Quick View' to XFGetCells?
- By clicking **OK**, the Quick View definition will be deleted and specific intersections of data will be converted to XFGetCell formulas.

This is a great way to set up a lot of formulas at once. However, these will hardcode all the dimension members in your formulas. If the member name is being displayed, the rows and columns can be synchronized to the pre-made quick view and cube view rows and columns. Your quick view can be adjusted to display the member name instead of the description through the quick view options.

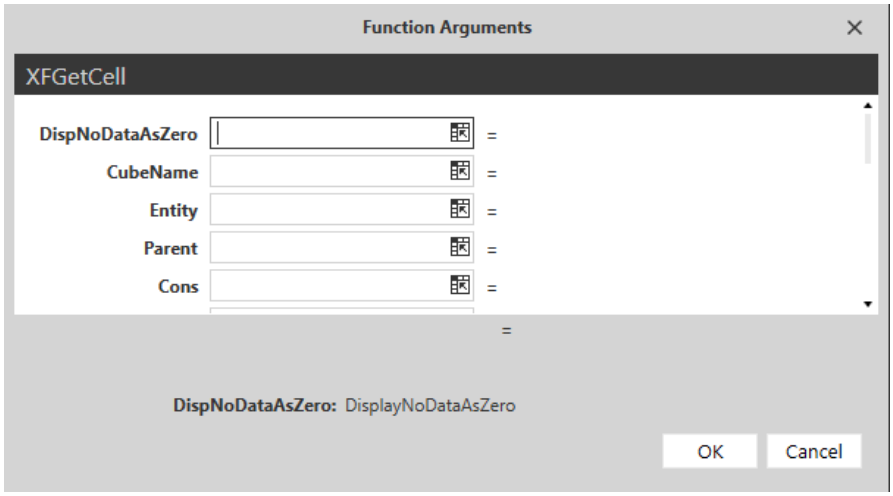
> **NOTE:** Quick view data that has been converted to XFGetCell cannot be converted back into a quick view. However, quick view content may be saved as a template before converting.

Additionally, XF Functions can be manually entered:

1. Select a cell and click on the insert function icon.
2. Select an XFGet function from the pop-up menu.



3. Fill out the necessary dimensions to build your XF Retrieve. If you do not want to fill one out, place a “” in the property and it defaults to the data in your cube POV.



4. Click **OK** to save the dimensions entered and the selected cell data changes to **#REFRESH**.
5. If your query is in the correct syntax, data populates in the cell after refreshing the sheet.

> **NOTE:** If a member name has been misspelled, the cell returns an error calling out the first misspelled member name.

Absolute Cell References

Absolute cell references are a functionality in the Excel Add-in that allows an exact cell reference to be kept when you copy the formula to another cell. To do this, put a $ in front of the portion of the cell reference that you want to keep.

For example, if you want to keep the entire A1 cell reference to copy to another cell you would change the cell reference to A1. If you only want to keep the column, the cell reference would be $A1 and if you only want to keep the row it would be A$1.

Relative Cell References

Relative cell references are the default state of cells in an Excel workbook. The reference is relative to the location of the cell. Formulas copied to new cells retain the location information and apply it to the new set of cells.

For example, if you reference A1 in cell C1 and then copy the formula into D1, the results will be a reference to B1 in cell D1 because the formula is referencing 2 columns before.

Visual Basic for Applications (VBA) Procedures

A Sub procedure is a series of Visual Basic statements that perform a specific task. It can take arguments, such as constants, variables, or expressions that are passed by a calling the procedure. If a Sub procedure has no arguments, the Sub statement must include an empty set of parentheses.

A Sub procedure begins with a Sub statement, followed by the tasks to be performed, and ends with an End Sub statement. The following snippet of VBA code represents the structure of a Sub procedure:

```
Sub [ProcedureName] (Arguments)
[Statements]
End Sub
```

OneStream Functions are used with Sub procedures to automate data submission from XFSetCells formulas.

> **NOTE:** To use these functions in Excel, you must have the latest version of OneStream and its corresponding Excel add-in installed.

Procedures currently supported are:

- LogonAndOpenApplication - Login varies depending on the authentication type setup for your application:

  > **NOTE:** Resource Owner Password Credentials (ROPC) is no longer supported. As a result, the following ProcessSSOAuthenticationAndCreateToken has been deprecated. See Legacy Single Sign On (SSO) for Bearer Token procedure.

  ```
  Sub SSOLogon()
  Set xfAddIn = Application.COMAddIns("OneStreamExcelAddIn")
  If Not xfAddIn Is Nothing Then
  If Not xfAddIn.Object Is Nothing Then
  ssoToken = xfAddIn.Object.ProcessSSOAuthenticationAndCreateToken
  ("https://golfstream.onestreamtest.com/OneStreamWeb",
  "user1@mycompany.com", "P@$$w0&D")
  xfUserName = xfAddIn.Object.GetXFUserNameFromSSOToken(ssoToken)
  ```

- OneStream IdentityServer (OIS) - (url, PAT, application)

  Supports the use of Personal Access Tokens. Refer to the Identity and Access Management Guide for creating and managing PATS.

```
Sub OIS_PAT_Logon()
 Set xfAddIn = Application.COMAddIns("OneStreamExcelAddIn")
 If Not xfAddIn Is Nothing Then
 If Not xfAddIn.Object Is Nothing Then
 ' OIS using PAT
 loggedIn = xfAddIn.Object.LogonAndOpenApplicationWithToken
("https://yoursite.onestreamcloud.com/OneStreamWeb", InsertPersonalAccessToken,
"GolfStreamDemo_v36")
 MsgBox ("Is Logged In : " & loggedIn)
 End If
 End If
End Sub
```

- Legacy Single Sign On (SSO) - (url, ssoToken, application)

  Supports the use of Bearer token from your identity provider

```
Sub SSOLogon()
 Set xfAddIn = Application.COMAddIns("OneStreamExcelAddIn")
 If Not xfAddIn Is Nothing Then
 If Not xfAddIn.Object Is Nothing Then
 ' Legacy SSO via Bearer Token
            Dim ssoToken As String
 ssoToken = <obtain Bearer token from your Identity Provider>"
 loggedIn = xfAddIn.Object.LogonAndOpenApplicationWithToken
("https://yoursite.onestreamcloud.com/OneStreamWeb", ssoToken, "GolfStreamDemo_
v36")
 MsgBox ("Is Logged In : " & loggedIn)
 End If
 End If
End Sub
```

- Native Authentication (Self-hosted) - (url, user, password, application)

  Support login via userid and password

```
Sub Native_Logon()
 Set xfAddIn = Application.COMAddIns("OneStreamExcelAddIn")
 If Not xfAddIn Is Nothing Then
 If Not xfAddIn.Object Is Nothing Then
 ' Native - UserName & Password
```

```
 loggedIn = xfAddIn.Object.LogonAndOpenApplication
("https://yoursite.onestreamcloud.com/OneStreamWeb", UserName, Password,
"GolfStreamDemo_v36")
 MsgBox ("Is Logged In ): " & loggedIn)
 End If
 End If
End Sub
```

- Logoff()
- RefreshXFFunctions() -- refer to the following example:

```
Sub RefreshXFFunctions()
Set xfAddIn = Application.COMAddIns("OneStreamExcelAddIn")
If Not xfAddIn Is Nothing Then
If Not xfAddIn.Object Is Nothing Then
Call xfAddIn.Object.RefreshXFFunctions
End If
End If
End Sub
```

- RefreshXFFunctionsForActiveWorksheet()
- RefreshQuickViews()
- RefreshQuickViewsForActiveWorksheet()
- RefreshCubeViews()
- RefreshCubeViewsForActiveWorkSheet()
- ShowParametersDlg()
- ShowParametersDlgForActiveWorksheet()

- SubmitXFFunctions () -- Automates the data loading process and eliminates the need to open the Excel files individually and submit data manually. Using a VBA routine, files with XFSET functions that are linked to other cells, sheets, and files can be programmatically submitted to OneStream. This procedure calls only XFSetCells. Refer to the following example:

```
Sub SubmitXFFunctionsTest()
 Set xfAddin = Application.COMAddIns("OneStreamExcelAddin")
 If Not xfAddin Is Nothing Then
 If Not xfAddin.Object Is Nothing Then
 Call xfAddin.Object.SubmitXFFunctions
 Call xfAddin.Object.RefreshXFFunctions
 End If
 End If
End Sub
```

Formatting in Excel

These formatting options are available in Excel:

- **Preserve Formatting**: Quickly preserve native Excel formatting in Cube Views, Quick Views, and Table Views when you submit or refresh your worksheet.
- **Styles**: Create custom styles to consistently format elements, like cells and headings.
- **Conditional**: Set styles for conditional text and data within your Excel worksheet.
- **Selection Styles for Cube Views**: Bring a Cube View into Excel, and format it using native Excel formatting.

Preserve Formatting in Excel

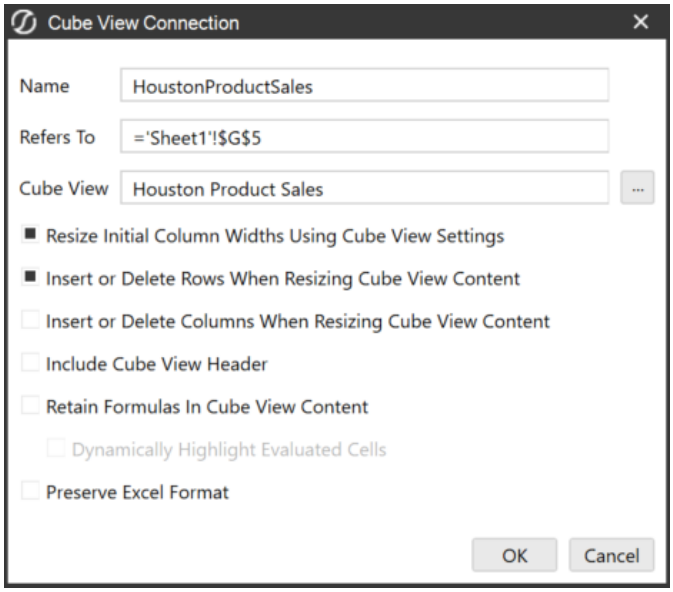
When you build a Cube View, Quick View, or Table View you have the option to enable the Preserve Excel Format setting. This setting enables you to preserve local Excel formatting without the need to create custom, conditional, or selection styles. When enabling preserve formatting on Cube Views or Table Views, we recommend disabling the Insert or delete rows or columns when resizing cube view content.

> **NOTE:** When Preserve Excel Format is enabled for Table Views, the DataType column must be specified as numeric. See DataType Object for Column Fields in the *Table Views Guide*.

Enable Preserve Excel Format

To turn on the Preserve Excel Format setting, create a Quick View, Cube View, or Table View. In the respective dialog box, enable the Preserve Excel Format setting.

Cube View



Quick View

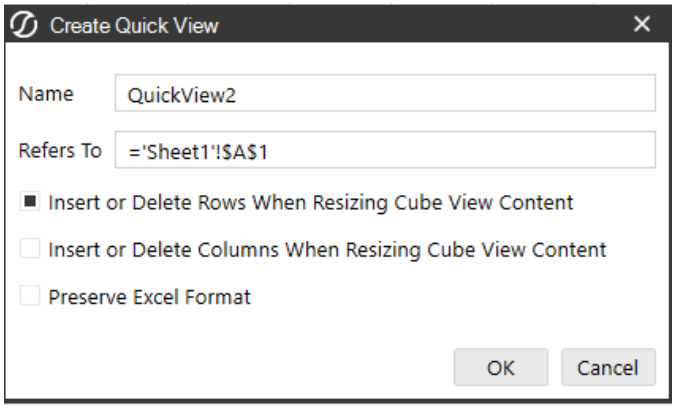
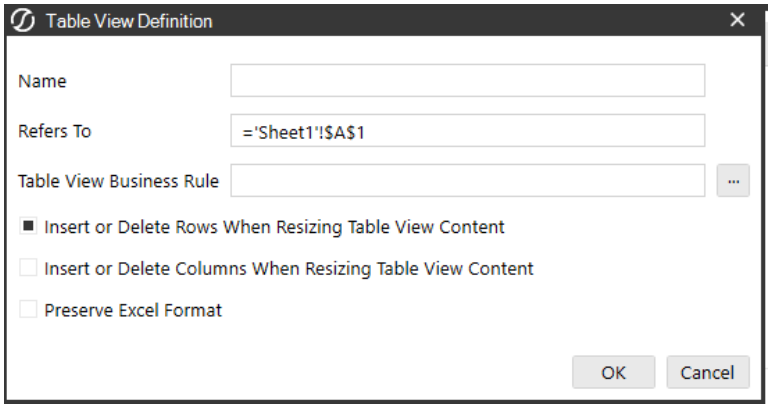


Table View



On already created views, you can edit the view and enable the setting.

When enabled, changes made to the sheet will be kept, even after you save and refresh your sheet. Additionally, you can copy your view and associated formatting will remain on the duplicated view when the Preserve Excel Format setting is enabled on the source Cube View, Quick View, or Table View.

> **NOTE:** When Preserve Excel Format is enabled on Quick Views, structural changes such as drill down, will not adjust formatting automatically. Manual reformatting may be required when modifying Quick Views.

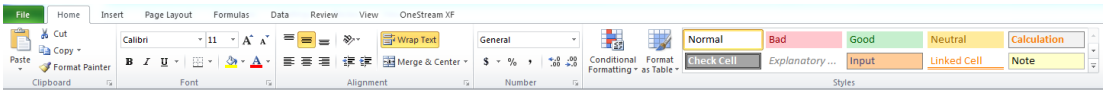
Styles

The same standard Styles are used in Excel, however, if you want to create a new style in order to change the format of how the numbers are displayed, see the example that follows.

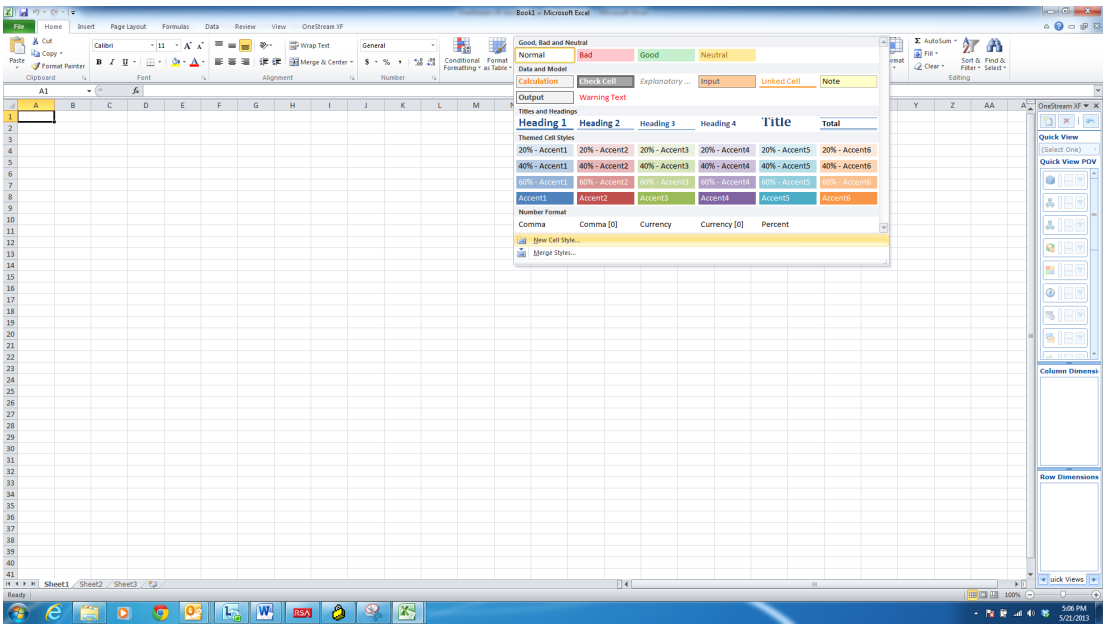
Creating a Custom Style

The following example was created in the Excel 2010 Version.

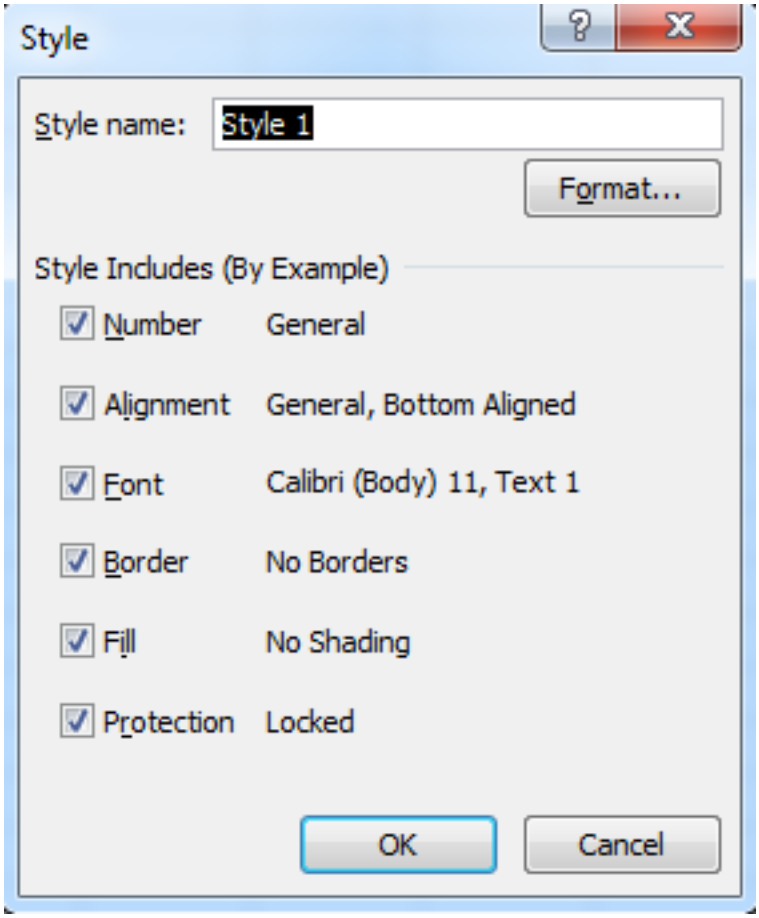
On the Home Tab, click on the new style sheet.



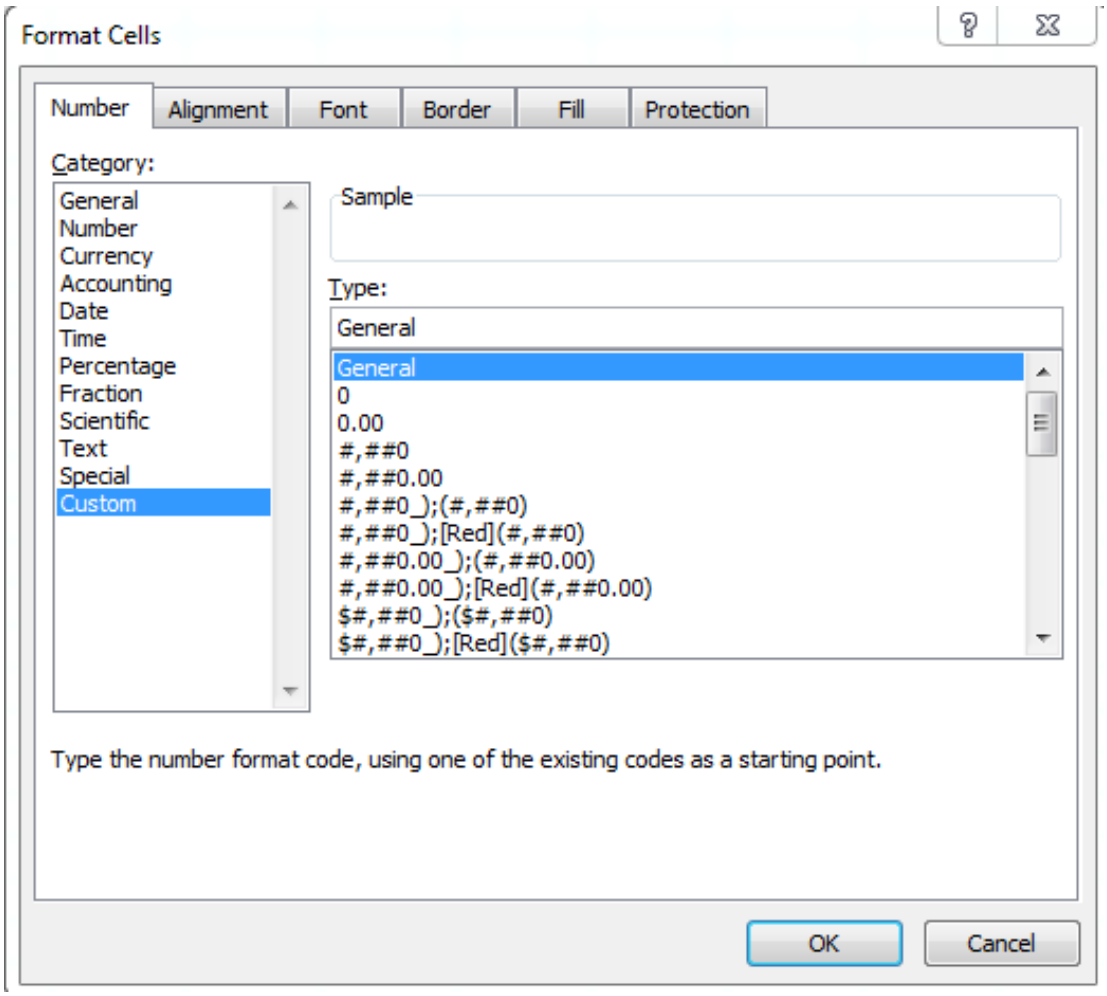
Select New Cell Style.



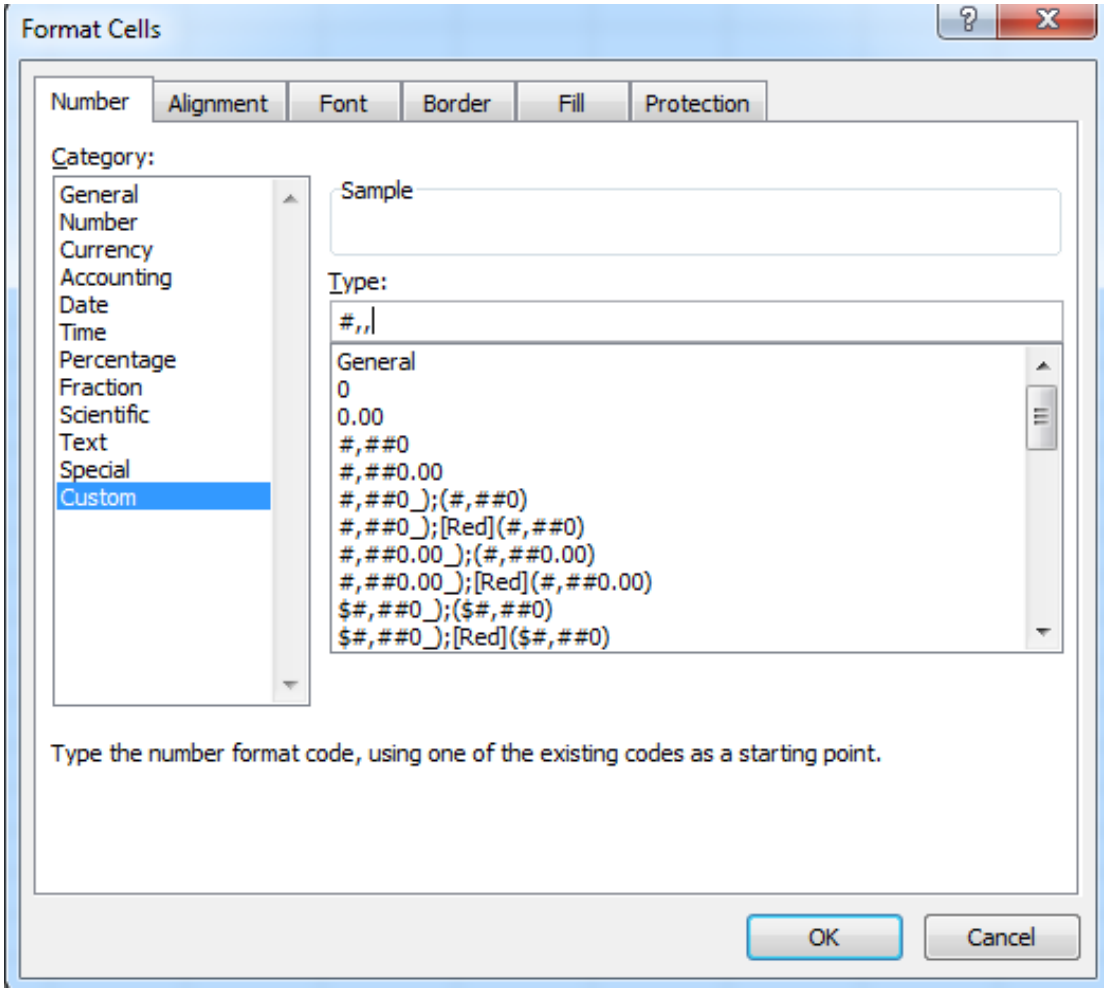
A new style window will appear, click Format.



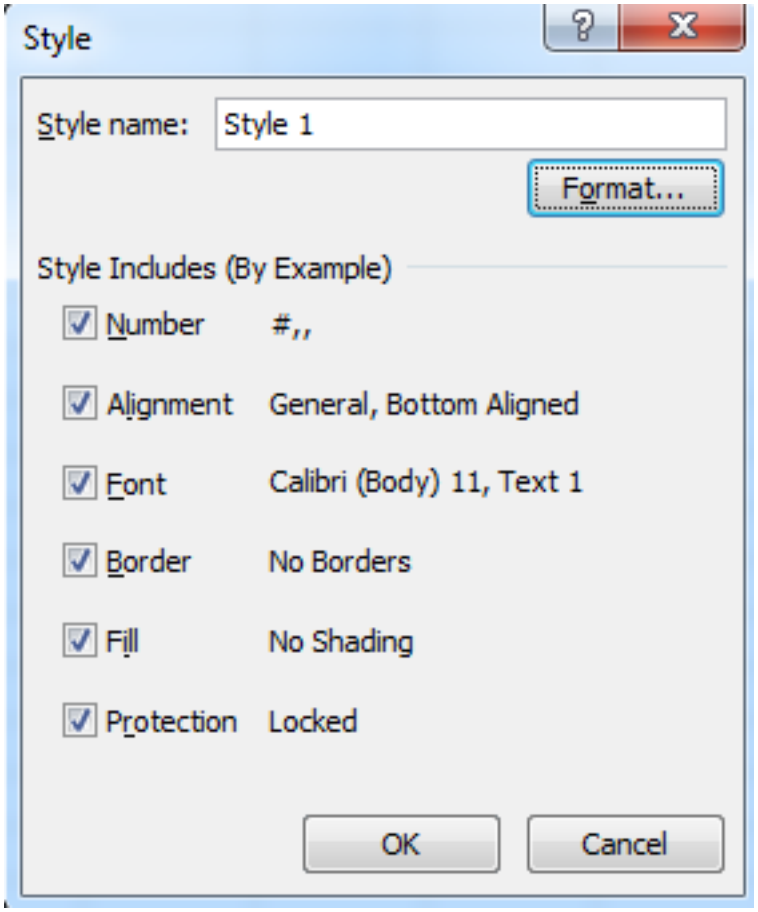
The format page will then display, click Custom.



Finally, under Type, enter the custom formatting. This example will be formatting for Millions. (#,,)


Format Cells
Number
Alignment
Font
Border
Fill
Protection
Category:
General
Number
Currency
Accounting
Date
Time
Percentage
Fraction
Scientific
Text
Special
Custom
Sample
Type:
#,,
General
0
0.00
#,##0
#,##0.00
#,##0_);(#,##0)
#,##0_);[Red](#,##0)
#,##0.00_);(#,##0.00)
#,##0.00_);[Red](#,##0.00)
$#,##0_);($#,##0)
$#,##0_);[Red]($#,##0)
Type the number format code, using one of the existing codes as a starting point.
OK
Cancel

Click OK.



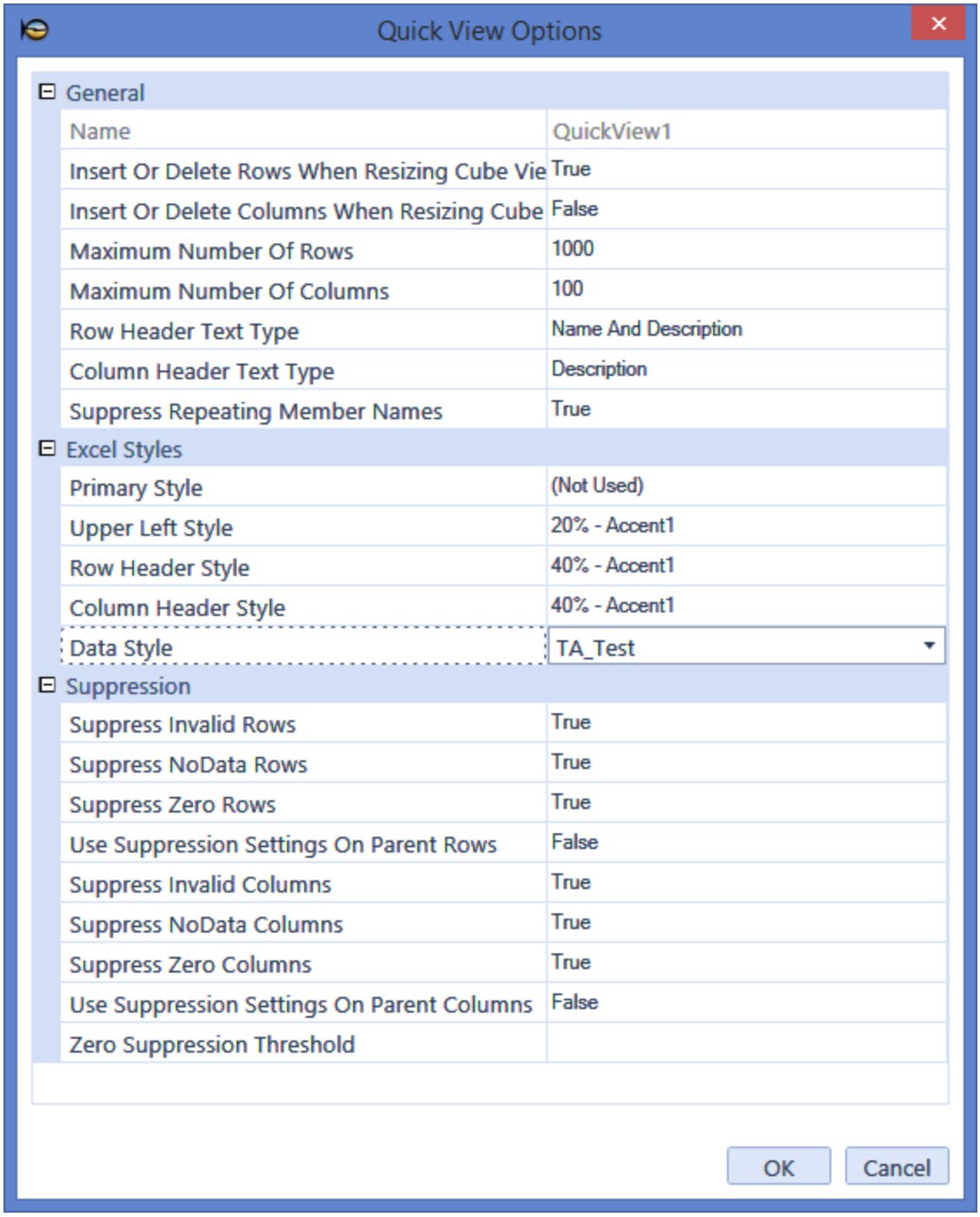
Type in a Style Name and click OK.

Now that a new style sheet has been created in Excel, it can be assigned to a Quick view.

This is the Quick View before the formatting:

	2013M2 - Feb 2013	
	MXPISTN - Federal-Mogul Puebla Pistons	
	MXN - Mexico, Pesos	USD - United States of America, Dollars
6099 - Total Revenue	389672805.8	30671437.83
1699 - Net Inventories	192558938	15068034.87

To add the formatting, click on the Edit Quick View Options in the Quick View Tab on the right side of the screen.



This is the Quick View after the update:

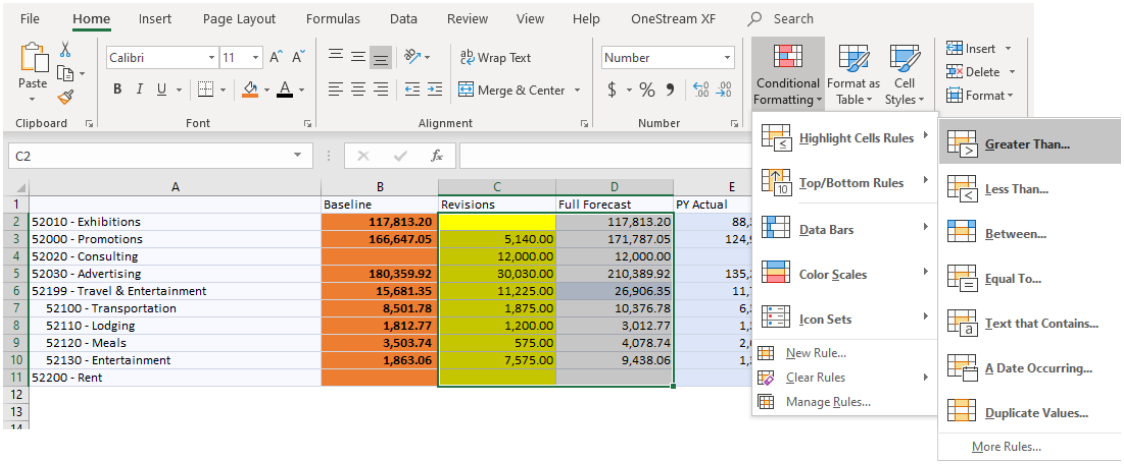
	2013M2 - Feb 2013	
	MXPISTN - Federal-Mogul Puebla Pistons	
	MXN - Mexico, Pesos	USD - United States of America, Dollars
6099 - Total Revenue	390	31
1699 - Net Inventories	193	15

Conditional Formatting

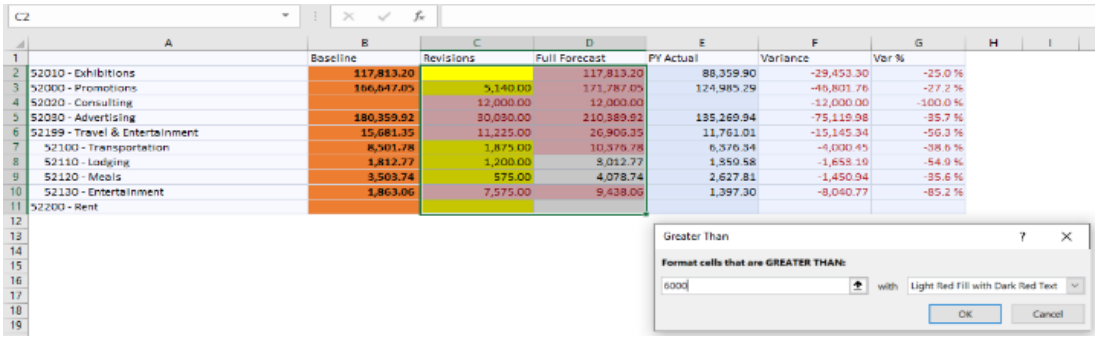
Use Conditional Formatting in Cube Views to visually explore and analyze data. You can highlight cells or ranges of cells, identify key values, and represent data using data bars, color scales, and icon sets that correspond to specific variations in the data. If there are any existing formats prior to applying Conditional Formatting, they will be retained if the range of cells containing the conditional formats do not meet the conditions of the rule. All styles from the cube view and the selection styles that had been previously applied to that range are overridden by conditional formatting.

Create Conditional Formatting in an Existing or New Cube View

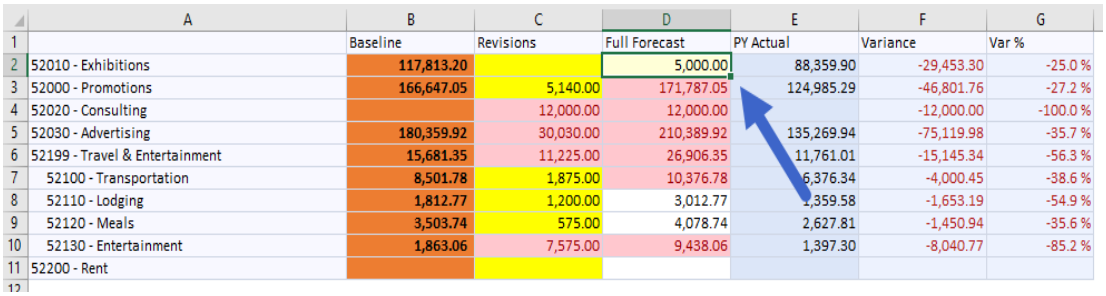
1. Click Home > Conditional Formatting > Highlight Cells Rules > Greater Than.



2. Enter 6000 and click OK.

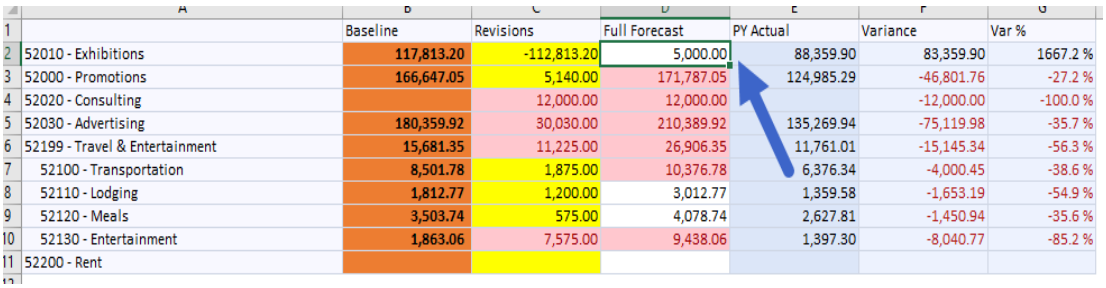


3. Go to OneStream and click Refresh Sheet to see that your changes have been applied.
4. If you make a change that is different than the value in the database, the cell will change to pale yellow, until you refresh or submit.



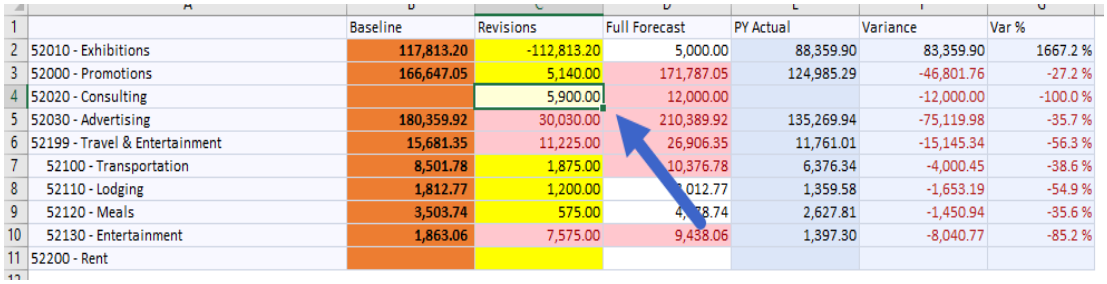
	A	B	C	D	E	F	G
1		Baseline	Revisions	Full Forecast	PY Actual	Variance	Var %
2	52010 - Exhibitions	117,813.20		5,000.00	88,359.90	-29,453.30	-25.0 %
3	52000 - Promotions	166,647.05	5,140.00	171,787.05	124,985.29	-46,801.76	-27.2 %
4	52020 - Consulting		12,000.00	12,000.00		-12,000.00	-100.0 %
5	52030 - Advertising	180,359.92	30,030.00	210,389.92	135,269.94	-75,119.98	-35.7 %
6	52199 - Travel & Entertainment	15,681.35	11,225.00	26,906.35	11,761.01	-15,145.34	-56.3 %
7	52100 - Transportation	8,501.78	1,875.00	10,376.78	6,376.34	-4,000.45	-38.6 %
8	52110 - Lodging	1,812.77	1,200.00	3,012.77	1,359.58	-1,653.19	-54.9 %
9	52120 - Meals	3,503.74	575.00	4,078.74	2,627.81	-1,450.94	-35.6 %
10	52130 - Entertainment	1,863.06	7,575.00	9,438.06	1,397.30	-8,040.77	-85.2 %
11	52200 - Rent						

5. If you submit, it will revert to the formatting that was in the cube view since it is no longer greater than 6000.



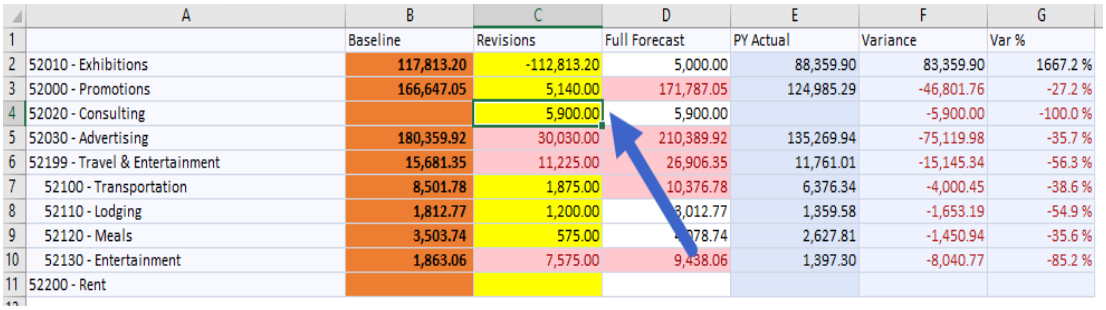
	A	B	C	D	E	F	G
1		Baseline	Revisions	Full Forecast	PY Actual	Variance	Var %
2	52010 - Exhibitions	117,813.20	-112,813.20	5,000.00	88,359.90	83,359.90	1667.2 %
3	52000 - Promotions	166,647.05	5,140.00	171,787.05	124,985.29	-46,801.76	-27.2 %
4	52020 - Consulting		12,000.00	12,000.00		-12,000.00	-100.0 %
5	52030 - Advertising	180,359.92	30,030.00	210,389.92	135,269.94	-75,119.98	-35.7 %
6	52199 - Travel & Entertainment	15,681.35	11,225.00	26,906.35	11,761.01	-15,145.34	-56.3 %
7	52100 - Transportation	8,501.78	1,875.00	10,376.78	6,376.34	-4,000.45	-38.6 %
8	52110 - Lodging	1,812.77	1,200.00	3,012.77	1,359.58	-1,653.19	-54.9 %
9	52120 - Meals	3,503.74	575.00	4,078.74	2,627.81	-1,450.94	-35.6 %
10	52130 - Entertainment	1,863.06	7,575.00	9,438.06	1,397.30	-8,040.77	-85.2 %
11	52200 - Rent						

6. If you make a change to a cell that has conditional formatting and a selection style,



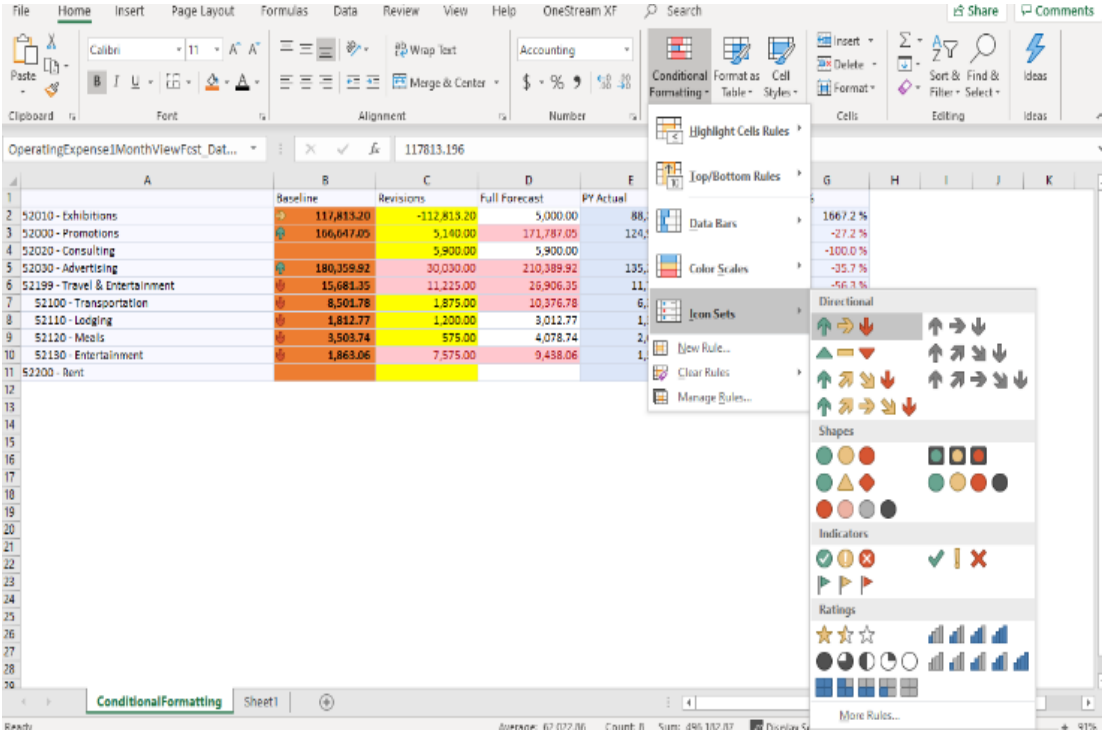
	A	B	C	D	E	F	G
1		Baseline	Revisions	Full Forecast	PY Actual	Variance	Var %
2	52010 - Exhibitions	117,813.20	-112,813.20	5,000.00	88,359.90	83,359.90	1667.2 %
3	52000 - Promotions	166,647.05	5,140.00	171,787.05	124,985.29	-46,801.76	-27.2 %
4	52020 - Consulting		5,900.00	12,000.00		-12,000.00	-100.0 %
5	52030 - Advertising	180,359.92	30,030.00	210,389.92	135,269.94	-75,119.98	-35.7 %
6	52199 - Travel & Entertainment	15,681.35	11,225.00	26,906.35	11,761.01	-15,145.34	-56.3 %
7	52100 - Transportation	8,501.78	1,875.00	10,376.78	6,376.34	-4,000.45	-38.6 %
8	52110 - Lodging	1,812.77	1,200.00	3,012.77	1,359.58	-1,653.19	-54.9 %
9	52120 - Meals	3,503.74	575.00	4,078.74	2,627.81	-1,450.94	-35.6 %
10	52130 - Entertainment	1,863.06	7,575.00	9,438.06	1,397.30	-8,040.77	-85.2 %
11	52200 - Rent						

when you submit, it will convert back to the selection style since it is no longer greater than 6000.



	A	B	C	D	E	F	G
1		Baseline	Revisions	Full Forecast	PY Actual	Variance	Var %
2	52010 - Exhibitions	117,813.20	-112,813.20	5,000.00	88,359.90	83,359.90	1667.2 %
3	52000 - Promotions	166,647.05	5,140.00	171,787.05	124,985.29	-46,801.76	-27.2 %
4	52020 - Consulting		5,900.00	5,900.00		-5,900.00	-100.0 %
5	52030 - Advertising	180,359.92	30,030.00	210,389.92	135,269.94	-75,119.98	-35.7 %
6	52199 - Travel & Entertainment	15,681.35	11,225.00	26,906.35	11,761.01	-15,145.34	-56.3 %
7	52100 - Transportation	8,501.78	1,875.00	10,376.78	6,376.34	-4,000.45	-38.6 %
8	52110 - Lodging	1,812.77	1,200.00	3,012.77	1,359.58	-1,653.19	-54.9 %
9	52120 - Meals	3,503.74	575.00	4,078.74	2,627.81	-1,450.94	-35.6 %
10	52130 - Entertainment	1,863.06	7,575.00	9,438.06	1,397.30	-8,040.77	-85.2 %
11	52200 - Rent						

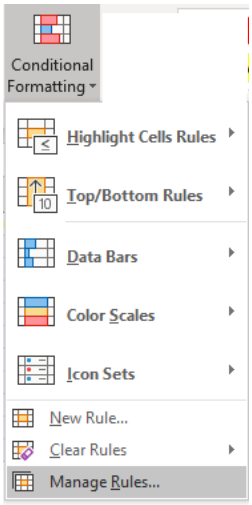
7. To add icons, go Home > Conditional Formatting > Icon Sets and select the icons to use, in this example, select the arrows.



8. The icons are part of the cube view.

	A	B	C	D	E	F	G
1		Baseline	Revisions	Full Forecast	PY Actual	Variance	Var %
2	52010 - Exhibitions	117,813.20	-112,813.20	5,000.00	88,359.90	83,359.90	1667.2 %
3	52000 - Promotions	166,647.05	5,140.00	171,787.05	124,985.29	-46,801.76	-27.2 %
4	52020 - Consulting		5,900.00	5,900.00		-5,900.00	-100.0 %
5	52030 - Advertising	180,359.92	30,030.00	210,389.92	135,269.94	-75,119.98	-35.7 %
6	52199 - Travel & Entertainment	15,681.35	11,225.00	26,906.35	11,761.01	-15,145.34	-56.3 %
7	52100 - Transportation	8,501.78	1,875.00	10,376.78	6,376.34	-4,000.45	-38.6 %
8	52110 - Lodging	1,812.77	1,200.00	3,012.77	1,359.58	-1,653.19	-54.9 %
9	52120 - Meals	3,503.74	575.00	4,078.74	2,627.81	-1,450.94	-35.6 %
10	52130 - Entertainment	1,863.06	7,575.00	9,438.06	1,397.30	-8,040.77	-85.2 %
11	52200 - Rent						
12							

9. You can also create, edit, delete, and view all conditional formatting rules.



10. If you save the workbook, the conditional formatting is saved.

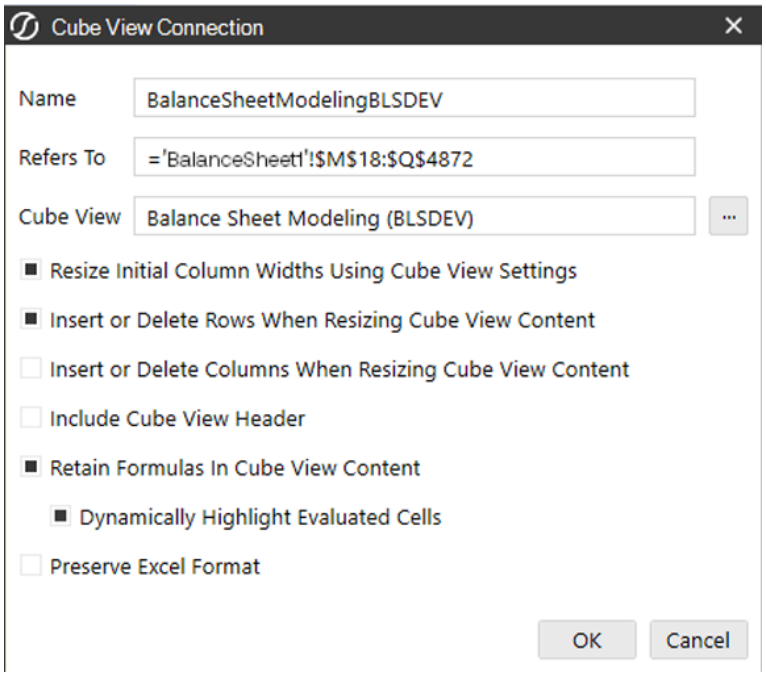
Selection Styles for Cube Views

Using Selection Styles

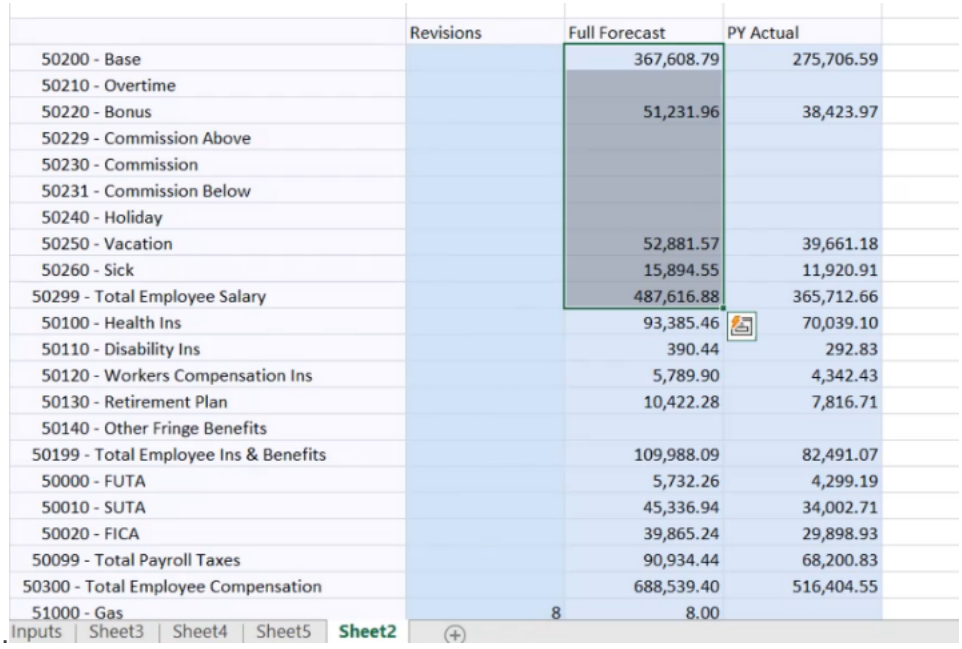
You can bring an existing cube view into Excel or Spreadsheet and format using functionality that is included in Excel, to create highly formatted reporting. You can also modify it locally as well as in spreadsheet. You can use existing cube view formatting, add new styles to apply changes to rows, columns, or cells, or a combination of existing styles with new styles, and add conditional formatting.

Creating a Selection Style

1. Log in using the OneStream menu.
2. Click **Cube Views** > **Cube View Connections**.
3. Click the **Add** button.
4. Select the cube view, click the **OK** button, then the **Close**button.



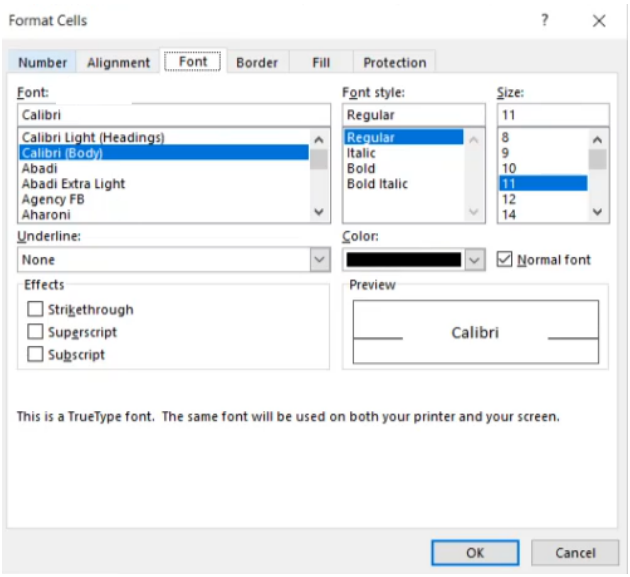
5. Select the cell, a group of cells, a column, or row to format.



	Revisions	Full Forecast	PY Actual
50200 - Base		367,608.79	275,706.59
50210 - Overtime			
50220 - Bonus		51,231.96	38,423.97
50229 - Commission Above			
50230 - Commission			
50231 - Commission Below			
50240 - Holiday			
50250 - Vacation		52,881.57	39,661.18
50260 - Sick		15,894.55	11,920.91
50299 - Total Employee Salary		487,616.88	365,712.66
50100 - Health Ins		93,385.46	70,039.10
50110 - Disability Ins		390.44	292.83
50120 - Workers Compensation Ins		5,789.90	4,342.43
50130 - Retirement Plan		10,422.28	7,816.71
50140 - Other Fringe Benefits			
50199 - Total Employee Ins & Benefits		109,988.09	82,491.07
50000 - FUTA		5,732.26	4,299.19
50010 - SUTA		45,336.94	34,002.71
50020 - FICA		39,865.24	29,898.93
50099 - Total Payroll Taxes		90,934.44	68,200.83
50300 - Total Employee Compensation		688,539.40	516,404.55
51000 - Gas	8	8.00	

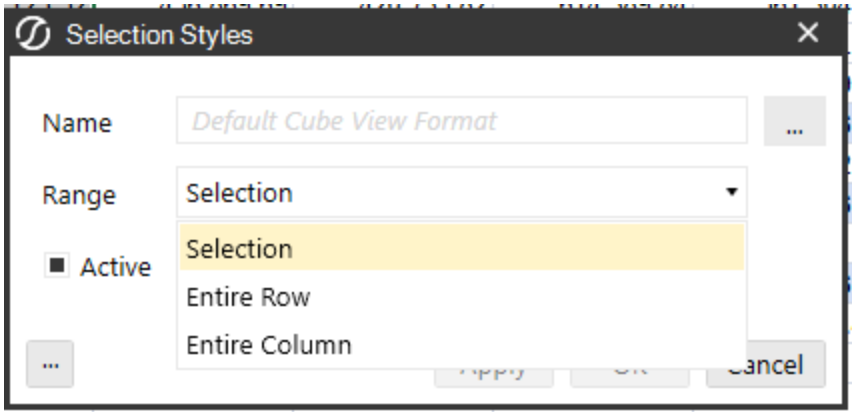
Inputs | Sheet3 | Sheet4 | Sheet5 | Sheet2

6. Right-click to open Format Cells or use the Home menu to choose formatting.



7. After formatting, click **OneStream** > **Cube Views** > **Selection Styles**.

8. Enter a Name and Range to apply to the current Selection, Entire Row, or Entire Column. Then click **OK**.



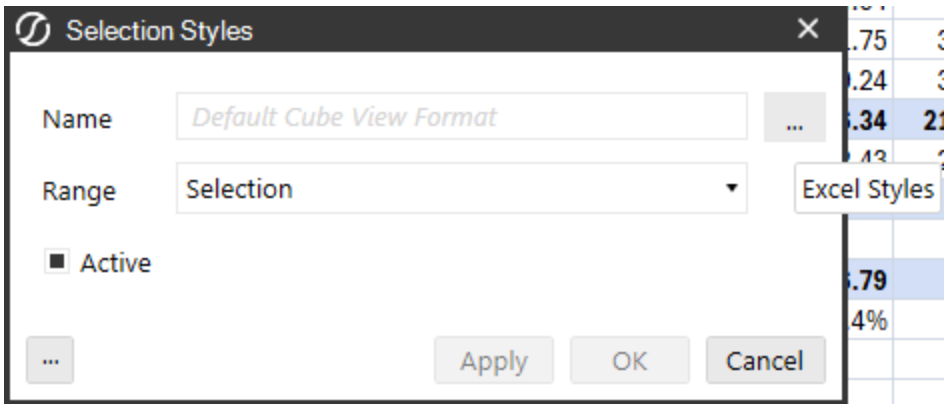
9. The formatting is applied to cube view.

	Revisions	Full Forecast
50200 - Base		$ 367,608.79
50210 - Overtime		
50220 - Bonus		$ 51,231.96
50229 - Commission Above		
50230 - Commission		
50231 - Commission Below		
50240 - Holiday		
50250 - Vacation		$ 52,881.57
50260 - Sick		$ 15,894.55
50299 - Total Employee Salary		$ 487,616.88
50100 - Health Ins		93,385.46
50110 - Disability Ins		390.44
50120 - Workers Compensation Ins		5,789.90
50130 - Retirement Plan		10,422.28

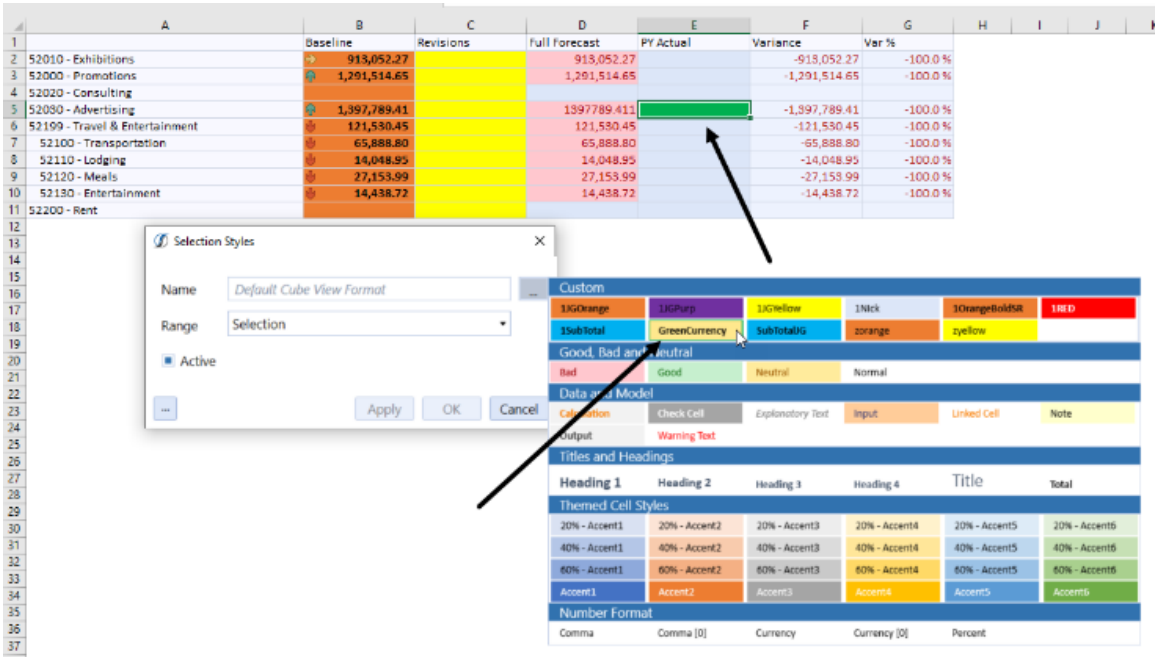
10. Save the file to save the formatting to the cube view.

Using an Existing Style

1. Select the cell, a group of cells, a column, or row to format, then click **Cube Views** > **Selection Styles**.
2. Click Excel Styles to select a style. If a style is not selected the Default Cube View Format is used.



3. On the style palette, hover to view the styles.



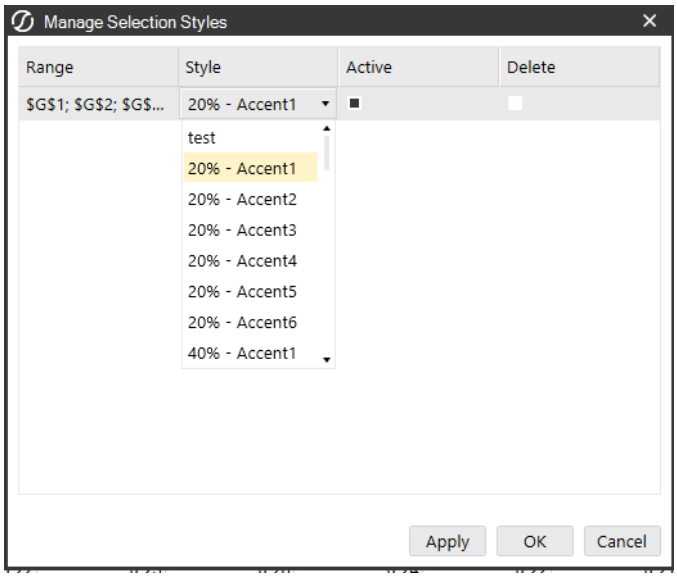
4. Click the style to use and then Apply or OK.
5. Save the file to save the cube view formatting.

Adding a Selection Styles Shortcut

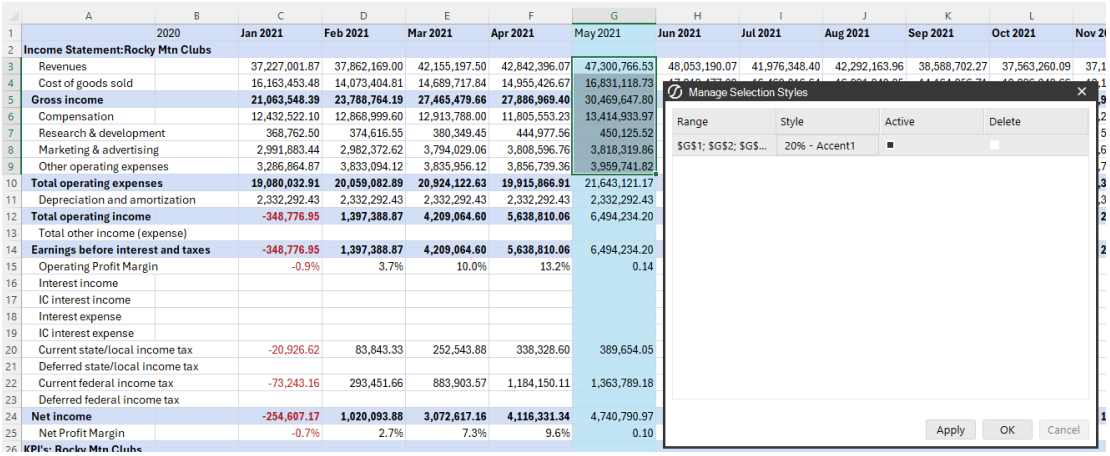
In Excel, right-click Selection Styles then click Add to Quick Access Toolbar. This adds a shortcut to your toolbar that launches the Selection Styles window.

Reviewing Styles and Ranges

1. Click **Cube Views** > **Manage Selection Styles**.
2. Review your styles that are applied to the cube view.
3. Modify selected styles.

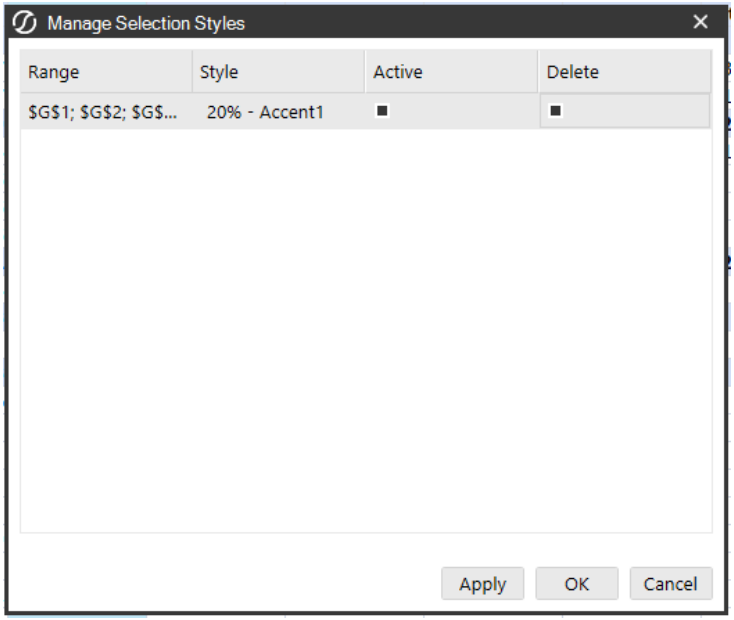


4. Enable or disable Active styles as needed and click **OK**.
5. Click **Apply** to preview the changes, or **OK** to apply the changes.



6. In Selection Styles, you will see styles that are no longer Active. Click **Activate** to enable them.

7. You can delete a style from the cube view but it will be available in the current workbook.



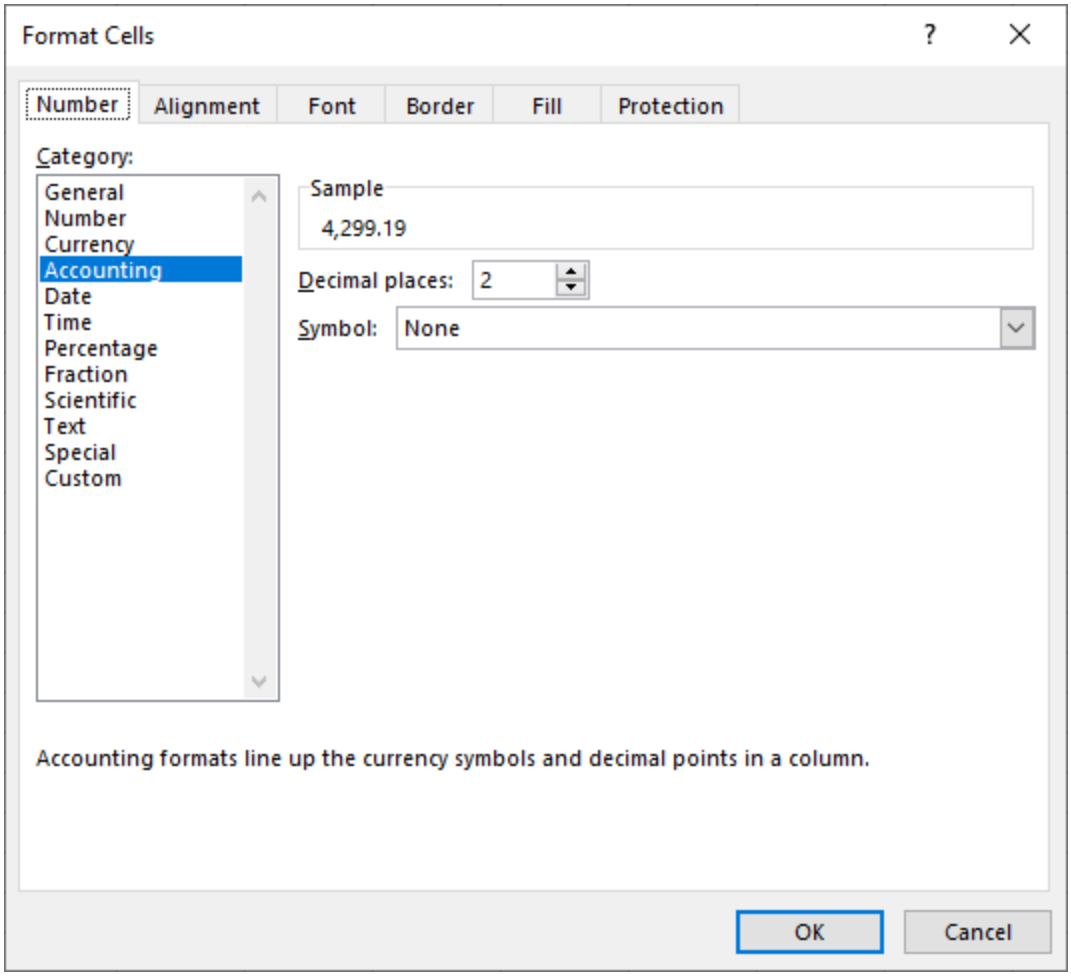
8. Save the file to save the cube view formatting.

Using Right-click Menu Options

You can manage selection styles by right-clicking in a cube view and selecting **OneStream** > **Cube Views** > **Cube View Connections** > **Selection Styles** > **Manage Selection Styles**.

Modifying and Duplicate Styles

1. In the Home menu, click **Styles** and right-click a style.
2. Click **Modify** or **Duplicate** and then **Format**.
3. Make any necessary changes and click **OK**.

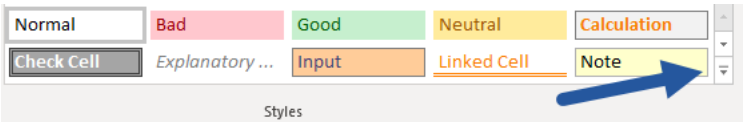


4. Save the file.

Merging Styles

You can use styles created in other workbooks in Excel only.

1. Open a new workbook.
2. Add a cube view.
3. Click **Home** > **Styles** then the **More** arrow.



4. Click **Merge Styles**, select the style, and click **OK**.

The style is now available in the workbook.