onestream

Guide de présentation de l'API

8.4.0 Release

Table des matières

Introduction

L'objectif du guide API est de fournir des informations détaillées sur les technologies et les interfaces de programmation d'applications à la disposition des conseillers et des fournisseurs désireux d'étendre les fonctionnalités de OneStream.

Ce document contient des informations sur les technologies utilisées dans le produit OneStream, les conventions de dénomination et les approches organisationnelles utilisées par l'équipe d'ingénieurs OneStream. Il comprend également des listes de référence détaillées pour les méthodes et les événements de l'API exposés par OneStream.

Pour les clients dans un environnement hosté par OneStream, consultez le *Guide de gestion des identités et des accès* pour obtenir des informations sur l'authentification avec OneStream et IdentityServer à l'aide de jetons d'accès personnels (PAT).

Technologies de développement

Langage de programmation

La plateforme OneStream est basée sur .Net Core. La base de code sous-jacente de OneStream est principalement assurée par des bibliothèques C# et quelques bibliothèques VB.Net C# et Visual Basic .NET sont les deux principaux langages de programmation utilisés pour coder avec .NET Core. C# et VB.NET ont des éléments de syntaxe très différents, mais Microsoft a développé ces langages simultanément dans le cadre d'une plateforme de développement .NET Core commune. C# et VB.Net sont tous deux développés, gérés et pris en charge par la même équipe de développement de langages chez Microsoft Ils se compilent dans le même langage intermédiaire *(IL)*qui s'exécute avec les mêmes bibliothèques d'exécution .NET Core. Bien que la syntaxe de programmation soit différente pour chaque langage, presque chaque commande en VB a une commande équivalente en C# et vice versa Les deux langages font référence aux mêmes classes de base .NET Core sous-jacentes pour étendre leurs fonctionnalités.

Technologie de l'interface utilisateur

L'interface utilisateur OneStream est basée sur la Windows Presentation Foundation *(WPF)* afin d'offrir à l'utilisateur final une expérience vraiment riche. WPF utilise XAML, un langage basé sur XML, pour définir et lier divers éléments d'interface. Les applications WPF peuvent être déployées en tant que programmes de bureau autonomes ou hébergées en tant qu'objet intégré dans un site Web. Le développement d'applications Windows 10 Store offre une autre possibilité de déployer des applications basées sur WPF, mais en tant qu'applications Windows uniquement.

Technologie du serveur

Tout le code OneStream est hébergé et exécuté avec Microsoft Internet Information Services *(IIS)* Cela signifie que le serveur Web *(code de service)* et le serveur d'application *(code de service)* sont exécutés au sein d'un processus hôte du pool d'applications IIS. Le code s'exécute sur le niveau du serveur d'application hébergé dans le pool d'applications IIS du serveur d'application . Il s'agit d'un concept très important à garder à l'esprit, car il y aura des moments où une règle de gestion devra interagir avec différents éléments du système . Le contexte dans lequel la règle de gestion s'exécute doit être compris afin d'établir une communication et/ou d'interagir avec ces autres éléments du système.

Technologie des bases de données

OneStream a été conçu pour fonctionner sur toutes les versions du moteur de base de données relationnelle Microsoft SQL Server *(Express, Standard, Data Center, Enterprise et Azure Database as a Service).* Pour les grandes entreprises, l'édition SQL Server Enterprise est recommandée car OneStreamelle assure le partitionnement des tables. Cela permet d'obtenir un débit maximal lors d'opérations fortement multithéoriques telles que la transformation et la consolidation des données. L'équipe OneStream d'ingénieurs s'engage à utiliser pleinement les capacités des versions les plus récentes de SQL Server et à maintenir la OneStream plateforme optimisée pour les nouvelles versions de SQL Server au fur et à mesure qu'elles sont disponibles.

Principes de base pour les développeurs

VB.Net et C#

La plateforme OneStream est entièrement basée sur .Net Core, tout comme le moteur de règles métier. Par conséquent, VB.Net et C# sont le choix logique pour la syntaxe des règles métier. Au moment de l'exécution, toutes les règles métier sont compilées à la demande et mises en cache pour une exécution rapide et fiable. L'écriture d'une règle métier en VB.Net ou C# offre à l'utilisateur final de nombreux avantages par rapport aux anciens produits basés sur VBScript. Les rédacteurs de règles métier peuvent s'attendre à des performances de code exceptionnelles, à une meilleure messagerie d'erreur et à une meilleure gestion des erreurs, car VB.Net et C# sont des langages de programmation complets. Au final, ces capacités se traduisent par un code de règles métier plus fiable.

> **NOTE:** Il existe deux grandes catégories de règles métier : Les règles métier partagées et les règles métier spécifiques à un élément. Les règles métier partagées peuvent être écrites en VB.NET ou en C#, tandis que les règles métier spécifiques aux éléments ne peuvent être écrites qu'en VB.NET.

Documentation en solution

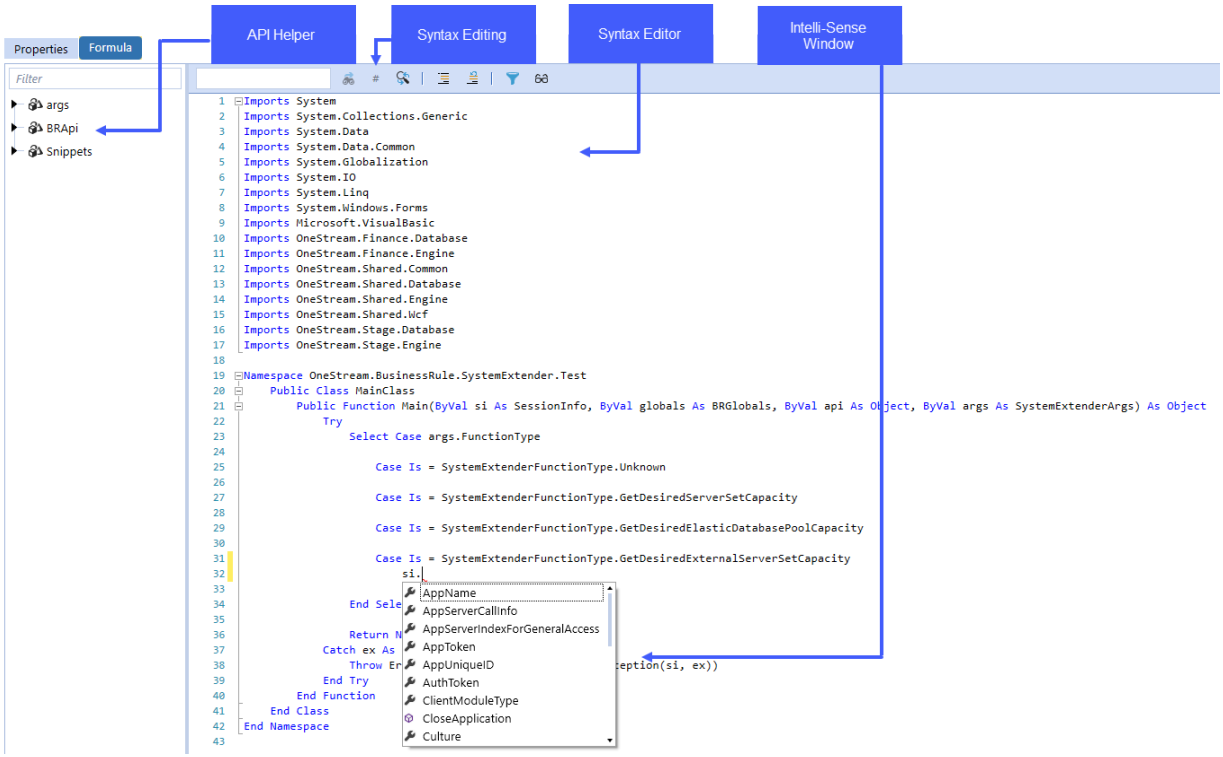
L'éditeur de règles de gestion comprend une aide contextuelle pour les propriétés et les méthodes de l'API, ainsi que des extraits *(exemples de code)*. La documentation dans la solution rend le processus d'écriture d'une règle métier plus efficace car la documentation

API, les objets et les échantillons sont présentés dans la fenêtre de l'éditeur de règles des gestion. En outre, les exemples de codage utiles accumulés par les équipes d'ingénieurs et de consultants OneStream sont également présentés de manière contextuelle dans l'éditeur de règles de gestion. Les entreprises et les partenaires peuvent créer leurs propres fragment et les inclure dans leur application en tant qu'extension des snippets prédéfinis OneStream (*solution d'édition de MarketPlace fragments requise*).

Présentation de l’éditeur Business Rules

L’éditeur Business Rules est un puissant écran intégré à la solution qui permet une aide contextuelle intégrée à l’API, une édition syntaxique avec intelli-sense et des capacités de mise en forme complètes. Le contenu de la syntaxe et la structure des règles de métier seront examinés en détail dans les sections suivantes du présent document.

L’image ci-dessous explique les principales régions et éléments de l’éditeur Business Rules.



Ressources utiles

VB.Net

VB.Net est l'un des langages de programmation les plus populaires actuellement. Ce langage est particulièrement populaire parmi les utilisateurs professionnels car sa syntaxe est perçue comme plus lisible et plus conviviale que celle des autres langages de programmation. VB.Net partage encore de nombreux éléments syntaxiques des anciens dialectes VB tels que VB6, VBA et VBScript. Cela signifie que les utilisateurs qui ont écrit des macros dans Microsoft Excel ou utilisé VBScript pour écrire des règles métier dans les solutions CPM de première génération devraient se sentir à l'aise avec les éléments de syntaxe de base de VB.Net. La principale difficulté d'apprentissage à laquelle les utilisateurs professionnels sont confrontés lorsqu'ils migrent vers VB.Net est de

comprendre la nature orientée objet du langage. Par rapport à VBScript, VB.Net offre des possibilités de codage plus élégantes. De nombreuses instructions et processus sont créés manuellement dans VBScript, mais dans VB.Net, ils sont encapsulés dans des bibliothèques d'objets que les utilisateurs peuvent simplement appeler.

Apprentissage de Microsoft VB.Net

Pour être à l'aise avec VB.Net, il faut connaître les bibliothèques et les objets de base fournis par .Net Core. Le lien ci-dessous renvoie à quelques ressources que les utilisateurs professionnels peuvent trouver utiles au cours du processus d'apprentissage de VB.Net.

Microsoft Visual Basic

https://msdn.microsoft.com/en-us/library/2x7h1hfk.aspx

C#

C# (prononcé « See Sharp ») est un langage de programmation moderne, orienté objet et sécurisé. Ce langage est particulièrement populaire parmi les développeurs car il leur permet de créer de nombreux types d'applications sûres et robustes qui s'exécutent dans .NET. C# a ses racines dans la famille des langages C et sera immédiatement familier aux programmeurs C, C++, Java et JavaScript.

Apprentissage de Microsoft C#

Le lien ci-dessous renvoie à quelques ressources que les utilisateurs professionnels peuvent trouver utiles au cours du processus d'apprentissage du langage C#.

https://docs.microsoft.com/en-us/dotnet/csharp/

Moteurs de la plateforme

La plateforme se compose de plusieurs moteurs de traitement. Ces moteurs ont des responsabilités distinctes en ce qui concerne le traitement du système et, par conséquent, ils exposent différentes interfaces API aux règles métier qu'ils appellent. La présente section donne un bref aperçu de chaque moteur de la plateforme et décrit ses principales responsabilités.

Moteur de flux de travail

Le moteur de flux de travail est considéré comme le moteur de contrôle ou le marionnettiste. La principale responsabilité de ce moteur est de contrôler et de suivre l'état des processus de métier définis dans les hiérarchies de flux de travail. Ce moteur est principalement accessible via le BRApi et peut être appelé à partir d'autres moteurs afin de vérifier l'état du flux de travail pendant l'exécution du processus. Le moteur de flux de travail fournit un modèle d'événement très riche permettant d'évaluer chaque processus de flux de travail et de le renforcer avec une logique métier spécifique au client si nécessaire (*voir Annexe 2 : Liste d'événements*).

Moteur d'étape

Le moteur d'étape a pour tâche de rechercher et de transformer les données externes en points de données analytiques valides. La principale responsabilité de ce moteur est de lire les données sources *(fichiers ou systèmes)* et d'analyser les informations dans un format tabulaire. Cela permet de transformer les données ou de les mettre en correspondance avec des membres valides définis par le moteur financier. Le moteur d'étape est un moteur multithread en mémoire qui offre la possibilité d'interagir avec les

données sources au fur et à mesure qu'elles sont analysées et transformées. Outre l'analyse et la transformation des données, le moteur d'étape dispose également d'un calcul sophistiqué qui permet de dériver et d'évaluer les données sur la base des données source entrantes. Le moteur d'étape fournit des services de qualité aux données sources en validant, en mappant et en exécutant les règles de contrôle des dérivés.

Moteur financier

Le moteur financier est un moteur d'analyse financière en mémoire. La principale responsabilité de ce moteur est d'enrichir et d'agréger les cellules de données de base en informations multidimensionnelles consolidées. Le moteur financier offre la possibilité de définir des calculs financiers sophistiqués par le biais de règles de gestion centralisées ainsi que de règles de gestion spécifiques aux membres *(Formules des membres)* . Il travaille en parallèle avec le moteur d'étape pour valider les intersections entrantes et avec le moteur de qualité des données pour exécuter les règles de confirmation utilisées pour valider les valeurs des données analytiques.

Moteur de qualité des données

Le moteur de qualité des données est chargé de contrôler les processus de confirmation et de certification des données. Ce moteur de confirmation est utilisé pour définir et contrôler la séquence des vérifications de la valeur des données nécessaires pour affirmer que les informations soumises par un système source sont correctes. Le moteur de certification est chargé de gérer les certifications des utilisateurs et de déterminer l'état d'achèvement des dépendants du flux de travail. Ce moteur est principalement accessible via le BRApi et peut être appelé à partir d'autres moteurs afin de vérifier l'état de la qualité des données pendant l'exécution du processus.

Moteur de gestion des données

Le moteur de gestion des données fournit des services d'automatisation des tâches à la plateforme. Ce moteur exécute des lots de commandes organisées en séquences contenant des étapes. Les étapes représentent des points d'entrée ou des mécanismes permettant d'exécuter des fonctions d'autres moteurs. Par exemple, l'étape « Clear Data » utilise les services du moteur financier. En outre, le moteur de gestion des données a la capacité d'exécuter une étape de règle métier qui exécute une règle métier personnalisée dans le cadre d'une séquence de gestion des données. Il s'agit d'une capacité incroyablement puissante, car elle permet d'enchaîner n'importe quel agencement d'étapes de traitement prédéfinies avec des étapes de règles de gestion personnalisées.

Moteur de présentation

Le moteur de présentation fournit à la plateforme des services étendus de visualisation des données. Le moteur de présentation est assuré par les moteurs suivants : Cube View Engine, Dashboard Engine, Parameter Engine, Book Engine et Extensible Document Engine. Le moteur de présentation est chargé de gérer et de fournir le contenu à l'utilisateur final, ainsi que de fournir un environnement de développement pour les éléments d'interface utilisateur personnalisés. Ce moteur permet OneStreamMarketPlace des capacités de développement d'applications et continue d'évoluer avec chaque version du produit. Comme le moteur de gestion des données, le moteur de présentation interagit avec tous les autres moteurs du produit et peut faire appel à leurs services.

BRApi

Le BRApi est commun à toutes les règles métier, à tous les moteurs et à toutes les API en cours d'exécution; il ne s'agit donc pas d'un moteur à proprement parler. Une fonction BRApi s'exécute en dehors des autres moteurs et peut orchestrer certaines fonctions à partir d'autres moteurs. En d'autres termes, une fonction BRApi peut être exécutée à partir d'un moteur (par exemple, Parser) pour indiquer à d'autres moteurs (par exemple, Finance) d'exécuter leurs propres API (par exemple, API.Data.GetDataCellUsingMemberScript). Autre exemple, si la fonction API.Data.GetDataCell est disponible dans le moteur Finance, une BRApi similaire appelée GetDataCellUsingMemberScript peut être exécutée à partir de n'importe quel moteur si les arguments appropriés lui sont fournis. Une utilisation courante est BRApi.ErrorLog.LogMessage à partir de n'importe quel moteur.

Structure et organisation de l'API

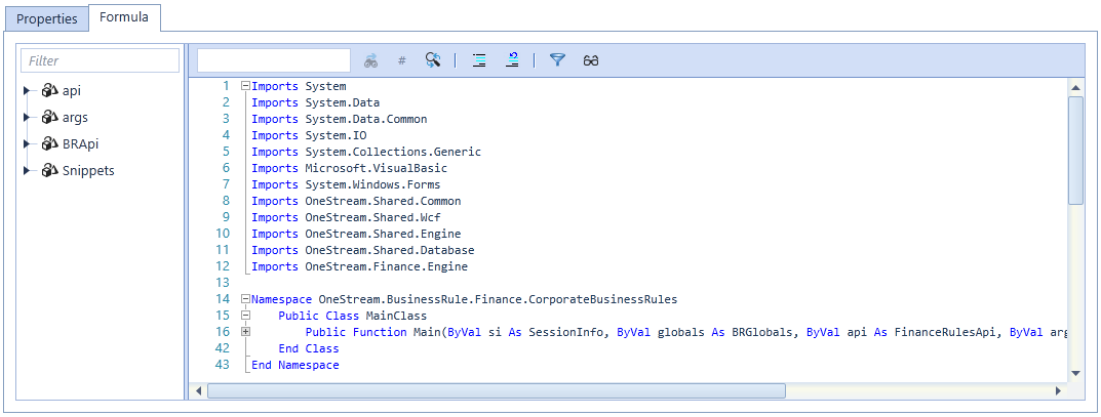
Espaces de nommage

.Net Core organise les bibliothèques de code en domaines appelés espaces de nommage. Le processus commence par l'identification d'espaces de nommage *(bibliothèques)* nécessaires à la procédure en cours de création. Les espaces de nommage fournissent une distinction aux objets et aux méthodes qui existent dans une bibliothèque de codes. En règle générale, les espaces de nommage commencent par le nom de l'entreprise qui a créé la bibliothèque de codes. Cela permet d'éviter les conflits de dénomination pour les objets qui partagent un nom commun, mais qui ont été créés par des fournisseurs de logiciels différents.

Dans le but de maintenir une syntaxe de codage aussi concise que possible, .Net Core permet à l'utilisateur de spécifier des espaces de nommage communs à utiliser en tête d'une règles métier. Ces lignes sont précédées du mot clé *« importations »*. L'ajout de déclarations d'importation évite d'avoir à taper le nom complet d'un objet dans un espace de nommage.

Toutes les règles métier sont pré-remplies avec les les espaces de nommage Microsoft couramment utilisés ainsi qu'avec les espaces de nommage spécifiques. Par exemple, l'ajout de la déclaration d'importation *System.Math* à une règles métier permet d'accéder aux objets d'espaces de nommage *System.Math*. Au lieu de taper *System.Math.Round (100.05,0)*, tapez *Round(100.05,0)*.

L'exemple ci-dessous montre les références de l'espaces de nommage utilisées dans une règle d'extensibilité standard.



Définition des espaces de nommage

OneStream est une plateforme logicielle vaste et sophistiquée et, par conséquent, beaucoup d'efforts ont été déployés pour organiser la base de code en un ensemble hiérarchique d'espaces de nommage (Namespaces). Cette section définit la hiérarchie des espaces de nommage et explique l'objectif principal des bibliothèques de code dans chaque espace de nommage. Il est important de comprendre la structure et la signification des espaces de nommage de la plateforme car la plupart des méthodes de l'API acceptent et renvoient des objets définis dans des espaces de nommage spécifiques. En comprenant la structure de la hiérarchie des espaces de nommage, les développeurs peuvent rechercher des objets à l'aide de l'intelli-sense dans l'éditeur de syntaxe.

Hiérarchie des espaces de nommage

La hiérarchie ci-dessous indique les espaces de nommage de la plateforme et les bibliothèques d'objets qu'ils contiennent. Cette hiérarchie est explorée à partir de l'éditeur de syntaxe des règles métier en tapant *OneStream* et en naviguant dans les listes déroulantes d'intelli-sense. Cette technique permet de trouver des objets à passer

dans une fonction d'API, des objets renvoyés par une fonction d'API ou des classes d'aide courantes disponibles dans la plateforme.

```
OneStream (Espace de nommage racine)

OneStream.BusinessRule

OneStream.BusinessRule.Finance

OneStream.BusinessRule.Parser

OneStream.BusinessRule.Connector

OneStream.BusinessRule.ConditionalRule

OneStream.BusinessRule.DerivativeRule

OneStream.BusinessRule.DashboardDataSet

OneStream.BusinessRule.DashboardExtender

OneStream.BusinessRule.DashboardStringFunction

OneStream.BusinessRule.Extender

OneStream.Client

OneStream.Client.SharedUI

OneStream.Client.SharedUI.FinanceMsgStrings

OneStream.Client.SharedUI.FinanceUIStrings

OneStream.Client.SharedUI.GeneralMsgStrings

OneStream.Client.SharedUI.GeneralUIStrings

OneStream.Client.SharedUI.StageMsgStrings
```

OneStream.Client.SharedUI.StageUIStrings

OneStream.Client.SharedUI.StringResourceFileType

OneStream.Client.SharedUI.StringResourceHelper

OneStream.Client.SharedUI.XFStrings

OneStream.Finance

OneStream.Finance.Engine

OneStream.Finance.Engine.DataApi

OneStream.Finance.Engine.EvalDataBufferDelegate

OneStream.Finance.Engine.FinanceRulesApi

OneStream.Finance.Engine.IAccountApi

OneStream.Finance.Engine.ICalcStatusApi

OneStream.Finance.Engine.IConsApi

OneStream.Finance.Engine.ICubesApi

OneStream.Finance.Engine.IDimensionsApi

OneStream.Finance.Engine.IEntityApi

OneStream.Finance.Engine.IFlowApi

OneStream.Finance.Engine.IFunctionsApi

OneStream.Finance.Engine.IFxRatesApi

OneStream.Finance.Engine.IMembersApi

```
OneStream.Finance.Engine.IPovApi

OneStream.Finance.Engine.IScenarioApi

OneStream.Finance.Engine.ITimeApi

OneStream.Finance.Engine.IUDApi

OneStream.Finance.Engine.IViewApi

OneStream.Finance.Engine.IWorkflowApi

OneStream.Stage

OneStream.Stage.Engine

OneStream.Stage.Engine.Parser

OneStream.Stage.Engine.ParserDimension

OneStream.Stage.Engine.TransformerDataCache

OneStream.Stage.Engine.Transformer

OneStream.Stage.Engine.TransformerDimension

OneStream.Stage.Engine.TransformRuleCache

OneStream.Shared

OneStream.Shared.Engine

OneStream.Shared.Engine.ExternalWcfClient

OneStream.Shared.Engine.TaskActivityStepWrapperItem

OneStream.Shared.Database
```

`OneStream.Shared.Database.DbConnInfo`

`OneStream.Shared.Common`

`OneStream.Shared.Common (diverses constantes, classes d'aide) & Data Transfer Objects 'DTO' (objets de transfert de données)`

`OneStream.Shared.Wcf`

`OneStream.Shared.Wcf.(Diverses constantes & Objets de transfert de données « DTO »).`

Appels financiers Microsoft

Les appels financiers font partie de l'espace de nommage Microsoft.VisualBasic et peuvent être utilisés pour des calculs tels que :

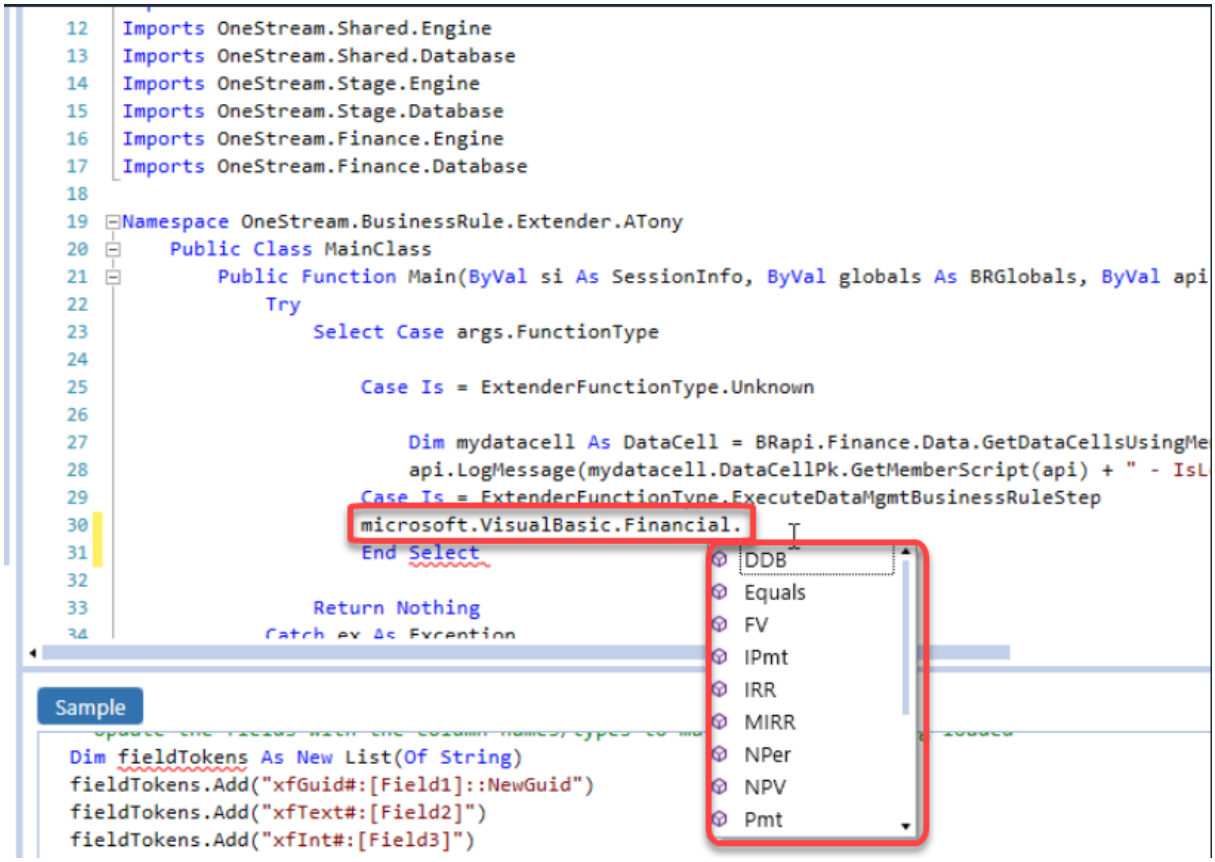
- Dépréciation
- Valeurs actuelles et futures
- Taux d'intérêt
- Taux de rendement
- Paiements

Ces fonctions sont accessibles à toute personne ayant accès aux règles métier. Elles peuvent être explorées dans l'éditeur de syntaxe des règles métier en tapant Microsoft.VisualBasic.Financial puis en naviguant dans les listes contextuelles d'intelli-sense.

Pour afficher toutes les méthodes de la classe Microsoft.Visual Basic Financial utilisées dans une règle métier :

1. Accédez à l'éditeur Business Rule :
    1. Dans l'application OneStream Software, cliquez sur l'onglet **Application**.
    2. Sous Tools, cliquez sur **Business Rules** .
    3. Développez la catégorie Business Rules appropriée ou cliquez sur **Search** dans la barre d'outils.
2. Cliquez sur l'onglet **Formula**.
3. Dans la fenêtre de l'éditeur, tapez **Microsoft.Visualbasic.Financial**.

Une liste de méthodes s'affiche.



Développement intra-solution

Le développement en cours de solution est le processus de création OneStream de règles métier pour fournir des solutions spécifiques à un domaine. Cela signifie que toutes les règles métier sont exécutées dans l'espace de traitement du serveur d'application. Le code écrit n'est exécuté que sur les serveurs d'application où OneStream est déployé .

Le développement dans l'environnement du serveur d'application permet aux développeurs de solutions de se concentrer sur le problème de l'entreprise plutôt que sur les préoccupations courantes en matière de programmation. La plateforme se charge de gérer les connexions, de déplacer les données entre les niveaux d'application et d'équilibrer la charge des activités du serveur.

Dans certains cas, le développement au sein de la solution est considéré comme une limitation parce que le développeur est restreint au codage au sein du niveau du serveur d'application. Cependant, dans la plupart des cas, l'efficacité et la qualité obtenues grâce au développement au sein de la plateforme dépassent les limites imposées par le codage au niveau du serveur d'application.

Développement personnalisé

Le développement personnalisé fait référence au développement d’applications autonomes qui interagissent avec la plateforme au niveau du serveur Web.

Développement Web personnalisé

La plateforme a la capacité d’afficher des pages web dans un tableau de bord personnalisé. Cela permet de créer des applications Web entièrement personnalisées au sein de la solution OneStream. Vous pouvez transmettre des informations sur le point de

vue de l’utilisateur et le flux de travail en tant que paramètres URL, ce qui permet à l’application web personnalisée d’agir comme un élément d’une solution intégrée.

Grâce à cette capacité, les développeurs sont libres de créer et d’intégrer toutes les solutions qu’ils peuvent imaginer.

Utilisation des outils de système

Règles de métier de système

Les règles de métier de System Extender sont utilisées en coordination avec les ensembles de serveurs Azure pour une évolutivité élastique au niveau des bases de données Azure et des ensembles de serveurs. La mise à l'échelle des serveurs et des eDTU peut être effectuée manuellement ou via les règles métier du système. Si les règles métier du système sont sélectionnées comme type de mise à l'échelle, OneStream appellera une règle métier d'extension du système définie par l'utilisateur pour déterminer si une mise à l'échelle est nécessaire. L'utilisateur est responsable de la mise en œuvre de la fonction de mise à l'échelle et du renvoi de l'objet de mise à l'échelle approprié à OneStream. Pour ce faire, il suffit d'ajouter une règle de gestion de l'extension du système et de l'affecter de manière appropriée.

Sous chaque énoncé de cas, ces règles et les Args et BRApis associés peuvent être utilisés pour vérifier la capacité actuelle de l'ensemble de serveurs, interroger les métriques sur un ensemble de serveurs ou une base de données Azure et influer sur le volume des ensembles de serveurs ou le niveau de la base de données Azure déployée.

Reportez-vous au *Guide d'installation et de configuration* sous *Paramètres de connexion aux bases de données Azure* et *Ensembles de serveurs* pour savoir où vous référer à ces règles de gestion. Exemple de point de départ d'une règle métier System Extender vide lors de sa création :

```
Namespace OneStream.BusinessRule.SystemExtender.Test
    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
```

Exemple de règle de gestion du système

Des données métriques sont transmises à cette fonction pour aider l'utilisateur à déterminer si le serveur ou la base de données doit être mis à l'échelle ou non. Selon l'objet de la mise à l'échelle, différentes données métriques sont transmises. Pour la mise à l'échelle du serveur, les métriques de l'environnement et les métriques de l'ensemble de mise à l'échelle sont transmises pour aider à déterminer la mise à l'échelle. Pour la mise à l'échelle des bases de données, les métriques d'environnement et les métriques SQL Server Elastic Pool sont transmises pour aider à déterminer la mise à l'échelle.

```
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
```

Base de données

L’écran Base de données permet aux administrateurs système de visualiser toutes les tableaux de la base de données de OneStream et fournit des outils pour gérer les données stockées et d’autres informations.

Tableaux

Cette option donne un accès en lecture seule à toutes les tableaux de la base de données et peut être utilisée pour des tâches telles que le débogage de problèmes sans avoir accès à la base de données, ou l’enregistrement des suppressions.

Outils

Les outils de base de données permettent aux administrateurs de système de gérer la base de données.

Enregistrements de données

Saisissez un filtre de membre afin de visualiser les données de l’ensemble du système.

Liste des événements

Règles de métier du gestionnaire d'événements

Gestionnaire d'événements WCF
Cette règle permet une interaction directe avec Microsoft Windows Communication Foundation, ce qui signifie qu'elle écoute les communications entre le client et le serveur web. La règle intercepte la communication, l'analyse et, si certains critères sont remplis, exécute sa logique. Cette méthode est très souple et peut être utilisée de diverses manières, par exemple pour créer, lire, supprimer et mettre à jour différents types d'objets dans le système pour les utilisateurs d'un groupe ou les changements de règles de transformation. Par exemple, une règle peut être créée pour envoyer un courriel à un auditeur à propos de chaque changement de métadonnées au fur et à mesure qu'il se produit.

Gestionnaire d'événements de transformation
Il peut être exécuté à différents moments, de l'importation au chargement. Opérations disponibles :

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

Gestionnaire d'événements pour les journaux

Il peut être exécuté avant, pendant ou après une opération de journal telle que la soumission, l'approbation ou l'enregistrement. Opérations disponibles :

SubmitJournal

ApproveJournal

RejectJournal

PostJournal

UnpostJournal

StartUpdateJournalWorkflow

EndUpdateJournalWorkflow

FinalizeUpdateJournalWorkflow

Save Data Event Handler

Cette commande permet de suivre tous les événements de sauvegarde dans une application.

Forms Event Handler

Il peut être exécuté avant, pendant ou après une opération telle que l'enregistrement d'un formulaire. Opérations disponibles :

SaveForm

CompleteForm

RevertForm

StartUpdateFormWorkflow

EndUpdateFormWorkflow

FinalizeUpdateFormWorkflow

Data Quality Event Handler

Il peut être exécuté avant, pendant ou après les événements de qualité des données tels que la confirmation et la certification. Opérations disponibles :

StartProcessCube

Calculer

Traduire

Consolider

EndProcessCube

FinalizeProcessCube

PrepareICMatch

StartICMatch

PrepareICMatchData

EndICMatch

StartConfirm

EndConfirm

FinalizeConfirm

SaveQuestionResponse

StartSetQuestionairreState

SaveQuestionairreState

EndSetQuestionairreState

StartSetCertifyState

SaveCertifyState

EndSetCertifyState

FinalizeSetCertifyState

Data Management Event Handler

Ce gestionnaire peut être exécuté avant ou après l'exécution d'une séquence ou d'une étape de gestion des données. Opérations disponibles :

StartSequence

ExecuteStep

EndSequence

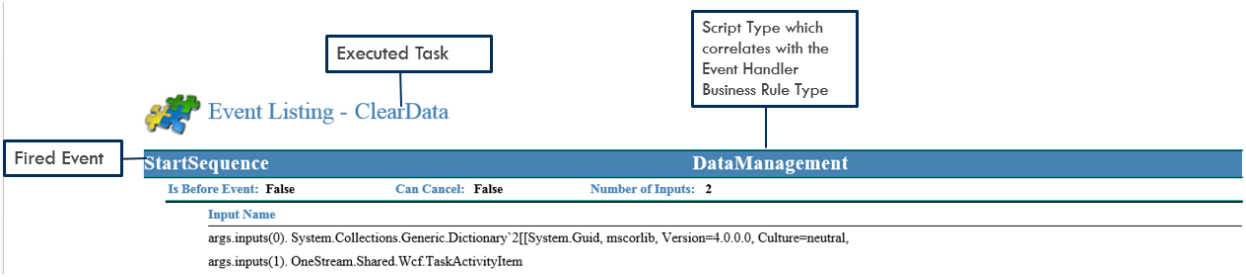
Workflow Event Handler

Cet événement peut être exécuté avant ou après une étape d'exécution d'un flux de travail. Opérations disponibles :

UpdateWorkflowStatus
WorkflowLock
WorkflowUnlock

Séquences de déclenchement des événements

OneStream déclenche une série d'événements lors de la réalisation de tâches via les règles de métier des gestionnaires d'événements. L'exemple ci-dessous explique comment lire le tableau qui fournit la séquence de déclenchement lors de l'exécution d'une tâche spécifique.



Effacer les données de Cube

StartSequence — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name

args.inputs(0). System.Collections.Generic.Dictionary`2[[System.Guid, mscorlib, Version=4.0.0.0, Culture=neutral,

args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

ExecuteStep — **DataManagement**

Is Before Event: True | Can Cancel: False | Number of Inputs: 2

Input Name

args.inputs(0). OneStream.Finance.Engine.DataMgmtStepMetadataInfo

args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

SaveCubeData — **SaveData**

Is Before Event: True | Can Cancel: True | Number of Inputs: 0

Input Name

args.inputs(0). SAVE DATA EVENT IS USED FOR DEBUG ONLY

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name

args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo

args.inputs(1). OneStream.Shared.Common.StepClassificationTypes

args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes

args.inputs(3). System.String

args.inputs(4). System.String

args.inputs(5). System.String

args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name

args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo

args.inputs(1). OneStream.Shared.Common.StepClassificationTypes

args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

ExecuteStep — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). OneStream.Finance.Engine.DataMgmtStepMetadataInfo

ExecuteStep — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

EndSequence — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Collections.Generic.Dictionary`2[[System.Guid, mscorlib, Version=4.0.0.0, Culture=neutral,
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

Effacer les données de l'étape

StartSequence — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Collections.Generic.Dictionary`2[[System.Guid, mscorlib, Version=4.0.0.0, Culture=neutral,
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

ExecuteStep — **DataManagement**

Is Before Event: True | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). OneStream.Finance.Engine.DataMgmtStepMetadataInfo
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

SaveCubeData — **SaveData**

Is Before Event: True | Can Cancel: True | Number of Inputs: 0

Input Name
args.inputs(0). SAVE DATA EVENT IS USED FOR DEBUG ONLY

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

ExecuteStep — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). OneStream.Finance.Engine.DataMgmtStepMetadataInfo

ExecuteStep — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

EndSequence — **DataManagement**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Collections.Generic.Dictionary`2[[System.Guid, mscorlib, Version=4.0.0.0, Culture=neutral,
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

Exécuter la gestion des données

StartSequence	DataManagement		
Is Before Event: False	Can Cancel: False	Number of Inputs: 2	

Input Name
args.inputs(0). System.Collections.Generic.Dictionary`2[[System.Guid, mscorlib, Version=4.0.0.0, Culture=neutral,
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

ExecuteStep	DataManagement		
Is Before Event: True	Can Cancel: False	Number of Inputs: 2	

Input Name
args.inputs(0). OneStream.Finance.Engine.DataMgmtStepMetadataInfo
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

ExecuteStep	DataManagement		
Is Before Event: False	Can Cancel: False	Number of Inputs: 2	

Input Name
args.inputs(0). OneStream.Finance.Engine.DataMgmtStepMetadataInfo
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

EndSequence	DataManagement		
Is Before Event: False	Can Cancel: False	Number of Inputs: 2	

Input Name
args.inputs(0). System.Collections.Generic.Dictionary`2[[System.Guid, mscorlib, Version=4.0.0.0, Culture=neutral,
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem

Importation de données de connexion

UpdateWorkflowStatus	Workflow		
Is Before Event: True	Can Cancel: True	Number of Inputs: 7	

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus	Workflow		
Is Before Event: False	Can Cancel: True	Number of Inputs: 7	

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

SaveCubeData	SaveData		
Is Before Event: True	Can Cancel: True	Number of Inputs: 0	

Input Name
args.inputs(0). SAVE DATA EVENT IS USED FOR DEBUG ONLY

StartLoadIntersect	Transformation		
Is Before Event: True	Can Cancel: False	Number of Inputs: 5	

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode

StartLoadIntersect — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(4). System.Guid

EndLoadIntersect — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

UpdateWorkflowStatus — Workflow

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — Workflow

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String

UpdateWorkflowStatus — Workflow

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(6). System.Guid

FinalizeLoadIntersect — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

Importer un fichier Excel

StartParseAndTransform — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

InitializeTransformer — Transformation

Is Before Event: True | Can Cancel: True | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

InitializeTransformer — Transformation

Is Before Event: False | Can Cancel: True | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ParseSourceData — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

InitializeExcelRangeLayout — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). OneStream.Stage.Engine.Parser
args.inputs(1). OneStream.Shared.Engine.StageRangeContent

InitializeExcelRangeLayout — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). OneStream.Stage.Engine.Parser
args.inputs(1). OneStream.Shared.Engine.StageRangeContent

ParseSourceData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ProcessDerivedRules — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ProcessDerivedRules — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes

ProcessDerivedRules — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(3). System.Guid

ProcessTransformRules — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ProcessTransformRules — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteData — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteData — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String

DeleteData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteRuleHistory — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteRuleHistory — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

WriteTransformedData — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

WriteTransformedData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer

WriteTransformedData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

SummarizeTransformedData — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

SummarizeTransformedData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

CreateRuleHistory — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

CreateRuleHistory — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name

CreateRuleHistory — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

EndParseAndTransform — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

UpdateWorkflowStatus — Workflow

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — Workflow

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String

UpdateWorkflowStatus — Workflow

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeParseAndTransform — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

Importer un fichier texte

StartParseAndTransform — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

InitializeTransformer — Transformation

Is Before Event: True | Can Cancel: True | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

InitializeTransformer — Transformation

Is Before Event: False | Can Cancel: True | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ParseSourceData — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ParseSourceData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ProcessDerivedRules — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ProcessDerivedRules — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ProcessTransformRules — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

ProcessTransformRules — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteData — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteRuleHistory — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

DeleteRuleHistory — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

WriteTransformedData — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

WriteTransformedData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

SummarizeTransformedData — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

SummarizeTransformedData — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

CreateRuleHistory — Transformation

Is Before Event: True | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

CreateRuleHistory — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

EndParseAndTransform — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

UpdateWorkflowStatus — Workflow

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — Workflow

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeParseAndTransform — Transformation

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Stage.Engine.Transformer
args.inputs(1). System.String
args.inputs(2). OneStream.Shared.Common.TransformLoadMethodTypes
args.inputs(3). System.Guid

Formulaire de procédure

CompleteForm	Forms
Is Before Event: True	Can Cancel: False
Number of Inputs: 4	

Input Name
args.inputs(0). OneStream.Shared.Wcf.XFFormEx
args.inputs(1). System.Boolean
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Common.WorkflowStatusTypes

CompleteForm	Forms
Is Before Event: False	Can Cancel: False
Number of Inputs: 4	

Input Name
args.inputs(0). OneStream.Shared.Wcf.XFFormEx
args.inputs(1). System.Boolean
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Common.WorkflowStatusTypes

CompleteForm	Forms
Is Before Event: True	Can Cancel: False
Number of Inputs: 4	

Input Name
args.inputs(0). OneStream.Shared.Wcf.XFFormEx
args.inputs(1). System.Boolean
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Common.WorkflowStatusTypes

CompleteForm	Forms
Is Before Event: False	Can Cancel: False
Number of Inputs: 4	

Input Name
args.inputs(0). OneStream.Shared.Wcf.XFFormEx
args.inputs(1). System.Boolean
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Common.WorkflowStatusTypes

StartUpdateFormWorkflow — **Forms**

Is Before Event: **False** Can Cancel: **False** Number of Inputs: **3**

Input Name
args.inputs(0). OneStream.Shared.Wcf.InputFormsProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean

EndUpdateFormWorkflow — **Forms**

Is Before Event: **False** Can Cancel: **False** Number of Inputs: **3**

Input Name
args.inputs(0). OneStream.Shared.Wcf.InputFormsProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean

UpdateWorkflowStatus — **Workflow**

Is Before Event: **True** Can Cancel: **True** Number of Inputs: **7**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: **False** Can Cancel: **True** Number of Inputs: **7**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String

UpdateWorkflowStatus — **Workflow**

Is Before Event: **False** Can Cancel: **True** Number of Inputs: **7**

Input Name
args.inputs(6). System.Guid

Journal de bord

SubmitJournal — **Journals**

Is Before Event: True | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Guid
args.inputs(1). OneStream.Shared.Wcf.JournalEx

SubmitJournal — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Guid
args.inputs(1). OneStream.Shared.Wcf.JournalEx

FinalizeSubmitJournal — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 1

Input Name
args.inputs(0). System.Guid

ApproveJournal — **Journals**

Is Before Event: True | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Guid
args.inputs(1). OneStream.Shared.Wcf.JournalEx

ApproveJournal — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Guid
args.inputs(1). OneStream.Shared.Wcf.JournalEx

FinalizeApproveJournal — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 1

Input Name
args.inputs(0). System.Guid

PostJournal — **Journals**

Is Before Event: True | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Guid
args.inputs(1). OneStream.Shared.Wcf.JournalEx

SaveCubeData — **SaveData**

Is Before Event: True | Can Cancel: True | Number of Inputs: 0

Input Name
args.inputs(0). SAVE DATA EVENT IS USED FOR DEBUG ONLY

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

PostJournal — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 2

Input Name
args.inputs(0). System.Guid
args.inputs(1). OneStream.Shared.Wcf.JournalEx

FinalizePostJournal — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 1

Input Name
args.inputs(0). System.Guid

StartUpdateJournalWorkflow — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 3

Input Name
args.inputs(0). OneStream.Shared.Wcf.InputJournalsProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean

EndUpdateJournalWorkflow — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Shared.Wcf.InputJournalsProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.JournalsAndTemplatesForWorkflow

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeUpdateJournalWorkflow — **Journals**

Is Before Event: False | Can Cancel: False | Number of Inputs: 3

Input Name
args.inputs(0). OneStream.Shared.Wcf.InputJournalsProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean

Flux de travail

StartValidateTransform — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Shared.Wcf.ValidationTransformationProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **True** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **False** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **True** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **False** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **False** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **True** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **False** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: **True** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid
args.inputs(4). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.DimensionValidationInfo
args.inputs(2). System.String
args.inputs(3). System.Guid

ValidateDimension — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(4). System.Guid

SetEventRules — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Shared.Wcf.ValidationTransformationProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). System.Guid

EndValidateTransform — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 4

Input Name
args.inputs(0). OneStream.Shared.Wcf.ValidationTransformationProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: **False** Can Cancel: **True** Number of Inputs: **7**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeValidateTransform — **Transformation**

Is Before Event: **False** Can Cancel: **False** Number of Inputs: **4**

Input Name
args.inputs(0). OneStream.Shared.Wcf.ValidationTransformationProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). System.Guid

StartValidateIntersect — **Transformation**

Is Before Event: **True** Can Cancel: **False** Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.ValidateIntersectionProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: **True** Can Cancel: **True** Number of Inputs: **7**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

EndValidateIntersect — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.ValidateIntersectionProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeValidateIntersect — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.ValidateIntersectionProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

SaveCubeData — **SaveData**

Is Before Event: True | Can Cancel: True | Number of Inputs: 0

Input Name
args.inputs(0). SAVE DATA EVENT IS USED FOR DEBUG ONLY

StartLoadIntersect — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

EndLoadIntersect — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeLoadIntersect — **Transformation**

Is Before Event: False | Can Cancel: False | Number of Inputs: 5

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

StartLoadIntersect — **Transformation**

Is Before Event: True | Can Cancel: False | Number of Inputs: 5

StartLoadIntersect — **Transformation**

Is Before Event: **True** | Can Cancel: **False** | Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

EndLoadIntersect — **Transformation**

Is Before Event: **False** | Can Cancel: **False** | Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: **True** | Can Cancel: **True** | Number of Inputs: **7**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — **Workflow**

Is Before Event: **False** | Can Cancel: **True** | Number of Inputs: **7**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes

UpdateWorkflowStatus — **Workflow**

Is Before Event: **False** | Can Cancel: **True** | Number of Inputs: **7**

Input Name
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeLoadIntersect — **Transformation**

Is Before Event: **False** | Can Cancel: **False** | Number of Inputs: **5**

Input Name
args.inputs(0). OneStream.Shared.Wcf.LoadCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). System.Boolean
args.inputs(3). OneStream.Shared.Wcf.LoadDataMode
args.inputs(4). System.Guid

StartProcessCube — **DataQuality**

Is Before Event: **False** | Can Cancel: **False** | Number of Inputs: **3**

Input Name
args.inputs(0). OneStream.Shared.Wcf.ProcessCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). OneStream.Shared.Wcf.TaskActivityItem

Consolidate — **DataQuality**

Is Before Event: **True** | Can Cancel: **False** | Number of Inputs: **3**

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem
args.inputs(2). OneStream.Shared.Wcf.DataUnitInfo

Consolidate — DataQuality

Is Before Event: False | Can Cancel: False | Number of Inputs: 3

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem
args.inputs(2). OneStream.Shared.Wcf.DataUnitInfo

NoCalculate — DataQuality

Is Before Event: True | Can Cancel: False | Number of Inputs: 3

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem
args.inputs(2). OneStream.Shared.Wcf.DataUnitInfo

NoCalculate — DataQuality

Is Before Event: True | Can Cancel: False | Number of Inputs: 3

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(1). OneStream.Shared.Wcf.TaskActivityItem
args.inputs(2). OneStream.Shared.Wcf.DataUnitInfo

EndProcessCube — DataQuality

Is Before Event: False | Can Cancel: False | Number of Inputs: 3

Input Name
args.inputs(0). OneStream.Shared.Wcf.ProcessCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). OneStream.Shared.Wcf.TaskActivityItem

UpdateWorkflowStatus — Workflow

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes

UpdateWorkflowStatus — Workflow

Is Before Event: True | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

UpdateWorkflowStatus — Workflow

Is Before Event: False | Can Cancel: True | Number of Inputs: 7

Input Name
args.inputs(0). OneStream.Shared.Wcf.WorkflowInfo
args.inputs(1). OneStream.Shared.Common.StepClassificationTypes
args.inputs(2). OneStream.Shared.Common.WorkflowStatusTypes
args.inputs(3). System.String
args.inputs(4). System.String
args.inputs(5). System.String
args.inputs(6). System.Guid

FinalizeProcessCube — DataQuality

Is Before Event: False | Can Cancel: False | Number of Inputs: 3

Input Name
args.inputs(0). OneStream.Shared.Wcf.ProcessCubeProcessInfo
args.inputs(1). OneStream.Shared.Wcf.WorkflowUnitPk
args.inputs(2). OneStream.Shared.Wcf.TaskActivityItem

Fonctions financières API

ID du membre

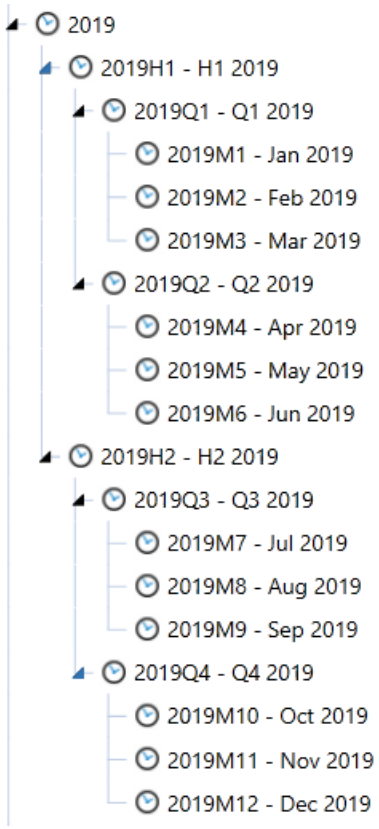
Il existe de nombreuses fonctions qui utilisent MemberID comme un nombre entier à transmettre comme propriété. Ces fonctions permettent d'obtenir le POV actuel d'un membre spécifique de la dimension afin d'effectuer une série de tâches, telles que

- Obtenir l'année en cours en fonction du POV de temps
    - Example : Api.Time.GetYearFromId(api.Pov.Time.MemberId)
- Obtenir la valeur d'un champ texte à partir du POV d'entité
    - Exemple : Api.Entity.Text(api.Pov.Entity.MemberId, 1)
- Obtenir le type de compte en fonction du POV actuel du compte
    - Example : Api.Account.GetAccountType(api.Pov.Account.MemberId)

Lorsque vous travaillez avec des formules et des calculs, il est préférable d'utiliser MemberId plutôt que le nom du membre.

Api.Pov.Time.MemberId

Le Api.Pov.Time.MemberId est obtenu à partir de l'identifiant du membre du temps pour le POV en cours d'exécution pendant le calcul. Le Time.MemberId est stocké comme un entier unique pour représenter un seul membre du temps. L'unicité est déterminée par l'agencement de l'année et de la période.



H1 = 001

Q1 = 002

M1 = 003

M2 = 004

M3 = 005

Q2 = 006

M4 = 007

M5 = 008

M6 = 009

H2 = 010

Q3 = 011

M7 = 012

M8 = 013

M9 = 014

Q4 = 015

M10 = 016

M11 = 017

M12 = 018

Le temps MemberId est construit comme suit : 2019003000

Le api.Pov.Time.MemberId est utilisé comme propriété dans de nombreuses fonctions Voici quelques-unes des fonctions les plus courantes :

- api.Time.GetYearFromId
- api.Time.GetPeriodNumFromId
- api.Time.GetNumDaysInTimePeriod
- api.Time.AddTimePeriods
- api.Time.AddYears

Api.Pov.Time.MemberId Utilisation

Exemple d'utilisation api.Pov.Time.MemberId :

```
Dim timeId As Integer = api.Pov.Time.MemberId
    BRApi.ErrorLog.LogMessage(si, "TimeId = " & timeId)
```

ErrorLog résultat :

```
TimeId = 2018003000
```

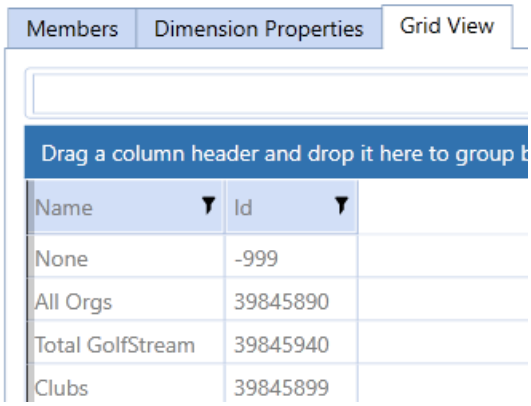
Exemple d'utilisation de api.Pov.Time.MemberId dans une formule de travail :

```
'Get Current Year as Integer Based on Current POV TimeId
Dim curYear As Integer = api.Time.GetYearFromId(api.Pov.Time.MemberId)
                         Function ITimeApi.GetYearFromId(Optional timeId As Integer) As Integer

'Execute Formula only if Current Year is Greater Than or Equal to 2018
If curYear >= 2018 Then
    'Only Run for Base Entities and at Local Currency
    If (Not api.Entity.HasChildren() And (api.Cons.IsLocalCurrencyforEntity())) Then
        api.Data.Calculate("A#CashCalc = A#10000")
    End If
End If
```

Api.Pov.Entity.MemberId

L'API Api.Pov.Entity.MemberId est obtenue à partir de l'identifiant du membre de l'entité pour le POV de l'entité en cours d'exécution pendant le calcul. L'identifiant du membre de l'entité Entity.MemberId est stocké sous la forme d'un entier unique représentant un seul membre de l'entité . L'identifiant du membre de l'entité est également trouvé en utilisant la vue en grille dans la bibliothèque Dimension de l'entité.



Le Api.Pov.Entity.MemberId est utilisé comme propriété dans de nombreuses fonctions. Voici quelques-unes des fonctions les plus courantes :

- Obtenir l'identifiant de la monnaie locale pour le POV de l'entité en cours.
    - Exemple : api.Entity.GetLocalCurrencyId(api.Pov.Entity.MemberId)
- Obtenir le nom du membre Cons de la monnaie locale pour l'entité POV actuelle.
    - Exemple :

      api.Entity.GetLocalCurrencyConsMember(api.Pov.Entity.MemberId).Name
- Obtenir la valeur du champ texte pour les membres de la dimension avant d'exécuter la formule de calcul.
    - Exemple : api.Entity.Text(api.Pov.Entity.MemberId, 1)
- Obtenir le pourcentage de consolidation pour la relation parent-enfant et spécifique à la localisation de l'utilisateur Peut également être déterminé par le type de scénario et l'heure.
    - Exemple : api.Entity.PercentConsolidation(api.Pov.Entity.MemberId, api.Pov.Parent.MemberId, api.Pov.ScenarioTypeId, api.Pov.Time.MemberId).XFToStringForFormula
- Obtenir le pourcentage de propriété pour la relation parent-enfant et spécifique à la localisation de l'utilisateur. Peut également être déterminé par le type de scénario et l'heure.
    - Exemple : api.Entity.PercentOwnership(api.Pov.Entity.MemberId, api.Pov.Parent.MemberId, api.Pov.ScenarioTypeId, api.Pov.Time.MemberId).XFToStringForFormula

Api.Pov.Entity.MemberId Utilisation

Exemple d'utilisation api.Pov.Entity.MemberId :

```
Dim entityId As Integer = api.Pov.Entity.MemberId
    BRApi.ErrorLog.LogMessage(si, "EntityId = " & entityId)
```

ErrorLog Résultat :

```
EntityId = 29360129
```

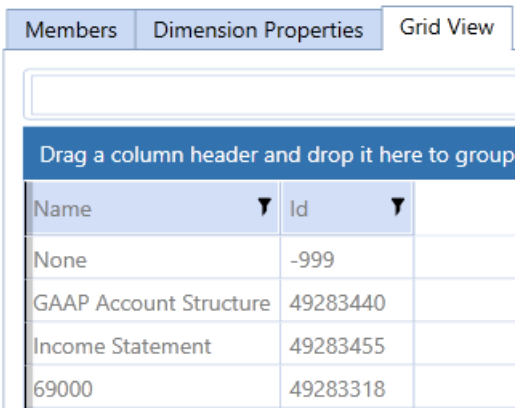
Exemple d'utilisation de api.Pov.Entity.MemberId dans une formule de travail :

```
'Get Text Value in Entity Text 1 Field for Current Entity POV
Dim entityText As String = api.Entity.Text(api.Pov.Entity.MemberId, 1)

'Only Run For Base Entities And at Local Currency
If (Not api.Entity.HasChildren() And (api.Cons.IsLocalCurrencyforEntity())) Then
    'Execute Formula if Entity has NA in the Entity Text 1 Field
    If entityText.XFEqualsIgnoreCase("NA") Then
        api.Data.Calculate("A#CashCalc = A#10000")
    End If
End If
```

Api.Pov.Account.MemberId

Api.Pov.Account.MemberId est obtenu à partir de l'identifiant du membre du compte pour la POV du compte en cours d'exécution pendant le calcul. Le Account.MemberId est stocké sous la forme d'un nombre entier unique représentant un seul membre du compte.L'identifiant du membre du compte est également trouvé à l'aide de la vue de la grille dans la bibliothèque des dimensions du compte.



Api.Pov.Account.MemberId est utilisé comme propriété dans de nombreuses fonctions. Voici quelques-unes des fonctions les plus courantes :

- Obtenir le type de compte en fonction du POV actuel du compte
  - Example : api.Account.GetAccountType(api.Pov.Account.MemberId)
- Obtenir la valeur du champ texte pour les membres de la dimension avant d'exécuter la formule de calcul
  - Exemple : api.Account.Text(api.Pov.Account.MemberId, 1)

Utilisation Api.Pov.Account.MemberId

Exemple d'utilisation api.Pov.Account.MemberId :

```
Dim accountType As AccountType = api.Account.GetAccountType(api.Pov.Account.MemberId)
    BRApi.ErrorLog.LogMessage(si, "AccountType = " & accountType.ToString)
```

ErrorLog Résultat :

```
AccountType = Revenue
```

Exemple d'utilisation de api.Pov.Account.MemberId dans une formule de travail :

```
'Get Account Type of Account and Use Specific FX Rate Type for Specific Account Types. Used in FinanceFunctionType.FXRate or Dynamic Calc
Dim accountType As String = api.Account.GetAccountType(api.Pov.Account.MemberId).ToString
Dim rateType As String = "ClosingRate"

If accountType = "Asset" Then

    Dim rate As Decimal = api.FxRates.GetCalculatedFxRate(rateType, api.Pov.Time.MemberId, args.FxRateArgs.SourceCurrencyId, args.FxRateArgs.DestCurrencyId)
    Return New FxRateResult(rate)

End If
```

Clé primaire de dimension. - DimPk

DimPk est connue sous le nom de clé primaire de dimension. Il s'agit d'une clé primaire unique qui est attribuée aux dimensions lors de leur création. Il s'agit d'un agencement de DimTypeId et de DimId.

DimPk est couramment utilisé pour identifier la dimension à utiliser lors de la vérification des membres en tant que membres de base ou descendants dans une dimension spécifique. DimPk est couramment utilisé dans les fonctions suivantes :

- Obtenir la clé primaire d'une dimension spécifique
    - Exemple : api.Dimensions.GetDim("UD1DimName").DimPk
- Vérifier s'il s'agit d'un membre de base d'un ancêtre spécifique
    - Example : api.Members.IsBase(dimPk, ancestorMemberId, baseMemberId, dimDisplayOptions)
- Obtenir les membres de base de Parent à partir de GetMember
    - Exemple : api.Members.GetBaseMembers(api.Pov.UD1Dim.DimPk, parent.MemberId, Nothing)

Utilisation DimPK

Exemple d'utilisation DimPK :

```
Dim dimPK As DimPk = api.Dimensions.GetDim("CostCenters").DimPk
    BRapi.ErrorLog.LogMessage(si, "DimPk for CostCenters = " & dimPK.ToString)
```

ErrorLog Résultat :

```
DimPk for CostCenters = DimTypeId: 9, DimId: 17
```

Exemple d'utilisation de api.Pov.UD1Dim.DimPk dans une formule de travail :

```
'Retrieve Base Members of Services in UD1 to Use in GetDataCell Loop
Dim parent As Member = api.Members.GetMember(DimType.UD1.Id, "Services")
Dim serviceNames As List(Of Member) = api.Members.GetBaseMembers(api.Pov.UD1Dim.DimPk, parent.MemberId, Nothing)

'Loop through all the Service Base Members
If Not serviceNames Is Nothing Then
    For Each serviceName As Member In serviceNames
        'GetDataCell for All Service Base Members as String and Decimal
        Dim serviceNameCellString As String = ("E#Houston:C#Local:S#Actual:T#2019M1:V#Periodic:A#Dept_Intersection:F#None:O#Forms:I#None:U1#" & serviceName.Name & ":
        Dim serviceNameCell As Decimal = api.Data.GetDataCell(serviceNameCellString).CellAmount
    Next
End If
```

Id du type de dimension

Dimension Type Id est une propriété de DimPk. Dimension Type Id est un identifiant entier unique qui est attribué à une dimension. Le DimTypeId se trouve dans la table Dim et le DimTypeId représente chaque dimension.

- Entity = 0
- Scenario = 2
- Account = 5
- Flow = 6
- UD1 = 9
- UD2 = 10
- UD3 = 11
- UD4 = 12
- UD5 = 13
- UD6 = 14
- UD7 = 15
- UD8 = 16

Le DimTypeId est utilisé dans plusieurs fonctions. DimTypeId est le plus souvent utilisé avec les fonctions GetMember ou GetMemberId où la première propriété de la fonction est DimTypeId. Dans ce cas, GetMember et GetMemberId doivent savoir quelle dimension utiliser pour le membre recherché par la fonction.

- Obtenir un membre spécifique dans une dimension spécifique
    - Exemple : api.Members.GetMember(DimType.Account.Id, "AcctMemberName")
- Obtenir l'identifiant d'un membre spécifique dans une dimension spécifique
    - Exemple : api.Members.GetMemberId(DimType.Account.Id, "AcctMemberName")

Utilisation DimTypeID

Exemple d'utilisation DimTypeId :

```
Dim dimTypeId As Integer = DimType.Account.Id
    BRApi.ErrorLog.LogMessage(si, "DimTypeID for Account = " & dimTypeId.ToString)
```

Résultat ErrorLog :

```
DimTypeID for Account = 5
```

Exemple d'utilisation de DimType.Account.Id dans une formule de travail :

```
'Get Cash Account Member and Store as a Variable to Pass into Api.Data.Calculate
Dim acctMember As Member = api.Members.GetMember(DimType.Account.Id, "10000")
api.Data.FormulaVariables.SetMemberVariable("variableAccount",acctMember)
api.Data.Calculate("A#CashCalc= A$variableAccount * 100")
```

Dimension de l'unité de données POV

Les calculs stockés s'exécutent sur la base du POV de l'unité de données. La dimension de l'unité de données se compose du cube, de l'entité, du parent, de la consolidation, du temps et du scénario.

Étant donné que les calculs stockés s'exécutent sur la base des dimensions d'unité de données, ces dimensions sont utilisées dans le cadre des instructions If pour exécuter les calculs sur les conditions. Les dimensions d'unité de données ne doivent pas être utilisées comme tampons de données de destination, ni du côté gauche de l'équation dans une formule api.Data.Calculate.

Les dimensions liées au compte, telles que le compte, le flux et les UD, ne sont pas disponibles au moment de l'exécution des calculs et ne peuvent donc pas être utilisées dans les instructions If pour les calculs stockés. En revanche, elles sont disponibles pour les calculs dynamiques.

Exécuter pour POV et vérifier les noms des membres pour les dimensions de l'unité de données avant d'exécuter le calcul :

- If api.Pov.Cube.Name.XFEqualsIgnoreCase("CubeName") Then
- If api.Pov.Entity.Name.XFEqualsIgnoreCase("EntityName") Then
- If api.Pov.Scenario.Name.XFEqualsIgnoreCase("ScenarioName") Then
- If api.Pov.Cons.Name.XFEqualsIgnoreCase("USD") Then

Unité de données Dimension POV Utilisation

Exemple d'utilisation api.Pov.Entity.Name :

```
Dim entityPovName As String = api.Pov.Entity.Name
    BRApi.ErrorLog.LogMessage(si, "Entity Pov Name = " & entityPovName)
```

ErrorLog Résultat :

```
Entity Pov Name = Houston Heights
```

Exemple d'utilisation de api.Pov.Entity.Name dans une formule de travail :

```
'Only Run Calculation For Houston Heights
If api.Pov.Entity.Name.XFEqualsIgnoreCase("Houston Heights") Then
    api.Data.Calculate("A#CashCalc = A#10000")
End If
```

```
'Only Run Calculation For Houston Heights
Dim entityPovName As String = api.Pov.Entity.Name

If entityPovName.XFEqualsIgnoreCase("Houston Heights") Then
    api.Data.Calculate("A#CashCalc = A#10000")
End If
```

Fonctions temporelles

Les API suivantes sont quelques-unes des fonctions temporelles les plus courantes :

- api.Time.GetYearFromId
- api.Time.GetPeriodNumFromId
- api.Time.GetNumDaysInTimePeriod
- api.Time.AddTimePeriods
- api.Time.AddYears

Api.Time.GetYearFromId

Cette fonction obtient l'année à partir du POV Time Id actuel. Elle évalue l'année et introduit ensuite la logique nécessaire à l'exécution de la formule.

```
'Get Current Year as Integer Based on Current POV TimeId
Dim curYear As Integer = api.Time.GetYearFromId(api.Pov.Time.MemberId)
                         Function ITimeApi.GetYearFromId(Optional timeId As Integer) As Integer

'Execute Formula only if Current Year is Greater Than or Equal to 2018
If curYear >= 2018 Then
    'Only Run for Base Entities and at Local Currency
    If (Not api.Entity.HasChildren() And (api.Cons.IsLocalCurrencyforEntity())) Then
        api.Data.Calculate("A#CashCalc = A#10000")
    End If
End If
```

Api.Time.GetPeriodNumFromId

Cette fonction permet d'obtenir le numéro de la période à partir du numéro d'identification de l'heure du POV en cours. La période est statique et est configurée avec des mois ou des semaines suivis du numéro de la période. Par exemple M1 – M12 ou W1 – W54. Il

évalue le numéro de la période et introduit ensuite la logique nécessaire à l'exécution de la formule.

Utilisation Api.Time.GetPeriodNumFromId

Exemple d'utilisation api.Time.GetPeriodNumFromId :

```
'Get Current Period As Integer Based on Current POV TimeId
Dim curPeriod As Integer = api.Time.GetPeriodNumFromId(api.Pov.Time.MemberId)
    BRApi.ErrorLog.LogMessage(si, "Period Number = " & curPeriod)
```

ErrorLog Résultat :

```
Period Number = 1
```

Exemple d'utilisation de api.Time.GetPeriodNumFromId dans une formule de travail :

```
'Get Time Member Id to Get Year and Period
Dim timeId As Integer = api.Pov.Time.MemberId

'Get Current Year As Integer Based On Current POV TimeId
Dim curYear As Integer = api.Time.GetYearFromId(api.Pov.Time.MemberId)

'Get Current Period As Integer Based on Current POV TimeId
Dim curPeriod As Integer = api.Time.GetPeriodNumFromId(api.Pov.Time.MemberId)
                           Function ITimeApi.GetPeriodNumFromId(Optional timeId As Integer) As Integer

'Execute Formula only if Current Year is Greater Than or Equal to 2018
'AND Current Period Number is Greater Than or Equal to 1
If curYear >= 2018 And curPeriod >= 1 Then
    'Only Run for Base Entities and at Local Currency
    If (Not api.Entity.HasChildren() And (api.Cons.IsLocalCurrencyforEntity())) Then
        api.Data.Calculate("A#CashCalc = A#10000")
    End If
End If
```

Api.Time.GetNumDaysInTimePeriod

Cette fonction permet d'obtenir le nombre de jours à partir de l'heure actuelle du POV. Le nombre de jours est déjà programmé en fonction du mois sélectionné. Il évalue le nombre

de jours pour une période et introduit ensuite la logique nécessaire à l'exécution de la formule.

Utilisation Api.Time.GetNumDaysInTimePeriod

Exemple d'utilisation api.Time.GetNumDaysInTimePeriod :

```
'Get Current Number of Days in Time Period
Dim numDays As Integer = api.Time.GetNumDaysInTimePeriod(api.Pov.Time.MemberId)
    BRApi.ErrorLog.LogMessage(si, "Number of Days in Period = " & numDays)
```

ErrorLog Résultat :

```
Number of Days in Period = 31
```

Exemple d'utilisation de api.Time.GetNumDaysInTimePeriod dans une formule de travail :

```
'Get Time Member Id to Get Year and Period
Dim timeId As Integer = api.Pov.Time.MemberId

'Get Current Year As Integer Based On Current POV TimeId
Dim curYear As Integer = api.Time.GetYearFromId(api.Pov.Time.MemberId)

'Get Current Period As Integer Based on Current POV TimeId
Dim curPeriod As Integer = api.Time.GetPeriodNumFromId(api.Pov.Time.MemberId)

'Get Current Number of Days in Time Period
Dim numDays As Integer = api.Time.GetNumDaysInTimePeriod(api.Pov.Time.MemberId)
                         Function ITimeApi.GetNumDaysInTimePeriod(Optional timeId As Integer) As Integer
'Execute Formula only if Current Year is Greater Than or Equal to 2018
'AND Current Period Number is Greater Than or Equal to 1
'AND Number of Days is Greater Than or Equal to 30 Days
If (curYear >= 2018 And curPeriod >= 1 And numDays >= 30) Then
    'Only Run for Base Entities and at Local Currency
    If (Not api.Entity.HasChildren() And (api.Cons.IsLocalCurrencyforEntity())) Then
        api.Data.Calculate("A#CashCalc = A#10000")
    End If
End If
```

Api.Time.AddTimePeriods

Cette fonction permet d'ajouter des périodes à l'Identité temporelle du POV en cours. Il transmet ces données à différentes fonctions telles que GetPeriodNumFromId introduit ensuite la logique nécessaire à l'exécution de la formule.

Utilisation Api.Time.AddTimePeriods

Exemple d'utilisation api.Time.AddTimePeriods :

```
'Get Current Time Member Id, Add 2 Periods, and Ok to Span Years
'Example:  Current Time Member Id = 2018003000.  Add 2 Periods, Then Member Id = 2018005000
Dim addTime As Integer = api.Time.AddTimePeriods(api.Pov.Time.MemberId, 2, True)
    BRApi.ErrorLog.LogMessage(si, "Add Time Periods = " & addTime)
```

ErrorLog Résultat :

```
Add Time Periods = 2018005000
```

Exemple d'utilisation de api.Time.AddTimePeriods dans une formule de travail :

```
'Get Time Member Id to Get Year and Period
Dim timeId As Integer = api.Pov.Time.MemberId

'Get Current Time Member Id, Add 2 Periods, and Ok to Span Years
'Example:  Current Time Member Id = 2018003000.  Add 2 Periods, Then Member Id = 2018005000
Dim addTime As Integer = api.Time.AddTimePeriods(api.Pov.Time.MemberId, 2, True)
                         Function ITimeApi.AddTimePeriods(timeId As Integer, numTimePeriodsToAdd As Integer, okToSpanYears As Boolean) As Integer

'Get Period from Add Time Period and Pass in GetPeriodNumFromId
Dim periodNum As Integer = api.Time.GetPeriodNumFromId(addTime)

'Execute Formula Only in Mar Period
If periodNum = 3 Then
    'Only Run for Base Entities and at Local Currency
    If (Not api.Entity.HasChildren() And (api.Cons.IsLocalCurrencyforEntity())) Then
        api.Data.Calculate("A#CashCalc = A#10000")
    End If
End If
```

Api.Time.AddYears

Cette fonction permet d'ajouter des années à l'identité temporelle actuelle du POV. Il transmet ces données à différentes fonctions telles que GetYearFromId ou GetPeriodNumFromId et introduit ensuite la logique nécessaire à l'exécution de la formule.

Utilisation Api.Time.AddYears

Exemple d'utilisation api.Time.AddYears :

```
'Get Current Time Member Id and Add 2 Years
'Example:  Current Time Member Id = 2018003000.  Add 2 Years, Then Member Id = 2020003000
Dim addYears As Integer = api.Time.AddYears(api.Pov.Time.MemberId, 2)
    BRApi.ErrorLog.LogMessage(si, "Added 2 Years To Current Time POV = " & addYears)
```

ErrorLog Résultat :

```
Added 2 Years To Current Time POV = 2020003000
```

Exemple d'utilisation de api.Time.AddYears dans une formule de travail :

```
'Get Current Time Member Id and Add 2 Years
'Example:  Current Time Member Id = 2018003000.  Add 2 Years, Then Member Id = 2020003000
Dim addYears As Integer = api.Time.AddYears(api.Pov.Time.MemberId, 2)
                          Function ITimeApi.AddYears(timeId As Integer, numYearsToAdd As Integer) As Integer

'Get Year from addYears and Pass it in for GetYearFromId function
Dim futureYear As Integer = api.Time.GetYearFromId(addYears)

'Execute Formula Only in Year 2020
If futureYear = 2020 Then
    'Only Run for Base Entities and at Local Currency
    If (Not api.Entity.HasChildren() And (api.Cons.IsLocalCurrencyforEntity())) Then
        api.Data.Calculate("A#CashCalc = A#10000")
    End If
End If
```

Utilisation des fonctions membres pour les calculs

Les fonctions de calcul des membres sont couramment utilisées par l'intermédiaire de l'API Finance pour accéder aux informations générales relatives à tout membre spécifié au sein d'une dimension. Les fonctions Membre permettent à un rédacteur de règles d'identifier les membres, d'obtenir des informations sur les membres et d'identifier les membres de base et les membres parents à exécuter dans les formules et les règles de gestion des membres.

Vous trouverez ci-dessous quelques-unes des fonctions les plus courantes des membres pour les calculs :

- GetMember
- GetMemberID
- GetBaseMembers

GetMember

Cette fonction permet d'obtenir un membre de dimension spécifique. Il est utilisé pour différentes fonctions telles que la fonction api.Data.FormulaVariables GetBaseMembers, les listes de membres personnalisées et lors de l'utilisation des identifiants de membres dans les tampons de données pour des processus tels que la consolidation personnalisée.

Utilisation GetMember

Exemple d'utilisation GetMember :

```
Dim getMember As Member = api.Members.GetMember(DimType.Account.Id, "10000")
    BRapi.ErrorLog.LogMessage(si, "Member Property Info = " & getMember.ToString)
```

ErrorLog Résultat :

```
Member Property Info = DimTypeId: 5, MemberId: 39845888,
Name: 10000, Description: Petty Cash, DimId: 38
```

Exemple d'utilisation de GetMember dans une formule de travail :

```
'Get Cash Account Member and Store as a Variable to Pass into Api.Data.Calculate
Dim acctMember As Member = api.Members.GetMember(DimType.Account.Id, "10000")
api.Data.FormulaVariables.SetMemberVariable("variableAccount",acctMember)
api.Data.Calculate("A#CashCalc= A$variableAccount * 100")
```

GetMemberId

Cette fonction permet d'obtenir l'identifiant d'un membre d'une dimension spécifique. Cette technique est couramment utilisée lorsque vous travaillez avec des tampons de données source et que les cellules d'un identifiant de membre spécifique doivent être modifiées.

Utilisation GetMemberID

Exemple d'utilisation GetMemberId :

```
Dim getMemberId As Integer = api.Members.GetMemberId(DimType.Account.Id, "10000")
    BRapi.ErrorLog.LogMessage(si, "Member Id for 10000 = " & getMemberId.ToString)
```

ErrorLog Résultat :

```
Member Id for 10000 = 39845888
```

Exemple d'utilisation de GetMemberId dans une formule de travail :

```
'Get Member Id for CashCalc Account
Dim cashCalcId As Integer = api.Members.GetMemberId(DimType.Account.Id, "CashCalc")

'Create a data buffer with the cells from S#Actual:A#10000 and copy the cells to S#ActualCopy:A#CashCalc
Dim destinationInfo As ExpressionDestinationInfo = api.Data.GetExpressionDestinationInfo("S#ActualCopy")
Dim sourceDataBuffer As DataBuffer = api.Data.GetDataBuffer(DataApiScriptMethodType.Calculate, "S#Actual:A#10000", destinationInfo)
'Check that the source Data Buffer exists
If Not sourceDataBuffer Is Nothing Then
    'Create a new result data buffer for data cells
    Dim resultDataBuffer As DataBuffer = New DataBuffer()
    'Loop through source data cells from the source data buffer
    For Each sourceCell As DataBufferCell In sourceDataBuffer.DataBufferCells.Values
        'Only get source cells that have data
        If (Not sourceCell.CellStatus.IsNoData) Then

            'Copy the cell from 10000 - Petty Cash to CashCalc Account in ActualCopy Scenario
            'The source data buffer contains source data cells with 10000 - Petty Cash AccountId
            'Change the source Account Id for 10000 - Petty Cash with the CashCalc Account Id
            Dim resultCell As New DataBufferCell(sourceCell)
            resultCell.DataBufferCellPk.AccountId = cashCalcId
            resultDataBuffer.SetCell(api.DbConnApp.SI, resultCell)

        End If
    Next
    'Set Destination Data Buffer with new Data Buffer with new cells including the CashCalc AccountId
    api.Data.SetDataBuffer(resultDataBuffer, destinationInfo)

End If
```

GetBaseMembers

Cette fonction permet d'obtenir les membres de base d'un membre parent spécifique. Il est couramment utilisé lorsque vous travaillez avec des listes de membres dans le cadre de FinanceFunctionType.MemberList, ou pour faire en sorte que les membres de la base passent en boucle par des dimensions spécifiques pour api.Data.GetDataCell.

Utilisation GetBaseMembers

Exemple d'utilisation GetBaseMembers :

```
'Retrieve Base Members of Services in UD1 to Use in GetDataCell Loop
Dim parent As Member = api.Members.GetMember(DimType.UD1.Id, "Services")
Dim serviceNames As List(Of Member) = api.Members.GetBaseMembers(api.Pov.UD1Dim.DimPk, parent.MemberId, Nothing)

'Loop through all the Service Base Members
If Not serviceNames Is Nothing Then
    For Each serviceName As Member In serviceNames
        BRapi.ErrorLog.LogMessage(si, "List of Base Members = " & serviceName.ToString)
```

ErrorLog Résultat :

List of Base Members = DimTypeId: 9, MemberId: 17825805, Name: GroundsMaint, Description: Ground Maintenance, DimId: 17
List of Base Members = DimTypeId: 9, MemberId: 17825797, Name: EquipMaint, Description: Equipment Maintenance, DimId: 17
List of Base Members = DimTypeId: 9, MemberId: 17825804, Name: GolfPros, Description: Golf Pro Staff, DimId: 17
List of Base Members = DimTypeId: 9, MemberId: 17825814, Name: ProShop, Description: ProShop Retail, DimId: 17

Exemple d'utilisation de GetBaseMembers dans une formule de travail :

```
'Retrieve Base Members of Services in UD1 to Use in GetDataCell Loop
Dim parent As Member = api.Members.GetMember(DimType.UD1.Id, "Services")
Dim serviceNames As List(Of Member) = api.Members.GetBaseMembers(api.Pov.UD1Dim.DimPk, parent.MemberId, Nothing)

'Loop through all the Service Base Members
If Not serviceNames Is Nothing Then
    For Each serviceName As Member In serviceNames
'       'GetDataCell for All Service Base Members as String, Decimal, and for International Rule Writing
        Dim serviceNameCellString As String = ("E#Houston:C#Local:S#Actual:T#2019M1:V#Periodic:A#Dept_Intersection:F#None:O#Forms:I#None:U1#" & serviceName.Name & ":U2#UD1Default:
        Dim serviceNameCell As Decimal = api.Data.GetDataCell(serviceNameCellString).CellAmount
        Dim serviceNameCellText As String = serviceNameCell.ToString("G", CultureInfo.InvariantCulture)

        'Check cell amount for intersection and then introduce logic based on cell amount
        'Use Data Buffer logic or api.Data.Calculate with SetDataBufferVariable function when in loop
    Next
End If
```

Écriture de calculs stockés

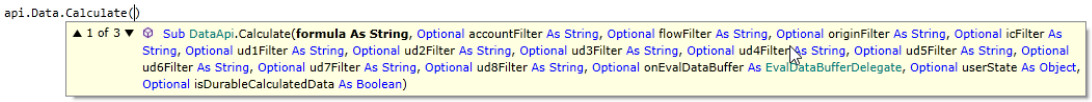
Lors de l'écriture d'une formule membre ou d'une règle de gestion pour un calcul stocké, les nouveaux nombres calculés stockent les données pour cet agencement Cube, Entité, Parent, Cons, Scénario et Temps. Par exemple, une unité de données.

Au lieu d'être renvoyés, de nombreux nombres sont calculés et stockés. Au lieu d'être renvoyés, de nombreux nombres sont calculés et stockés. Lors de l'exécution d'un calcul, d'une conversion ou d'une consolidation, la formule membre est appelée une seule fois pour une unité de données entière OneStream et ne précise pas avec quel compte, flux ou défini par l'utilisateur les nombres sont enregistrés.

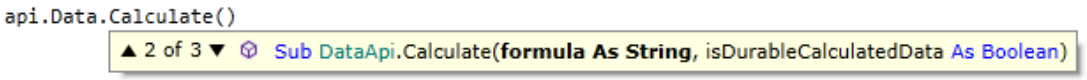
Au départ, cela peut prêter à confusion car les formules de membre sont souvent écrites dans la propriété Formula d'un compte, et les administrateurs pensent que OneStream n'autorisera que cette formule de membre spécifique à écrire dans ce compte spécifique. Cependant, l'insertion d'une formule de membre dans la propriété Formula d'un compte n'a qu'un but organisationnel. Lorsque OneStream appelle cette formule, elle est en train de calculer une unité de données et initialisera le moteur de l'API avec uniquement les dimensions de l'unité de données.

Les formules de base stockées sont généralement utilisées via la fonction Api.Data.Calculate de l'API , qui est utilisée de trois façons différentes :

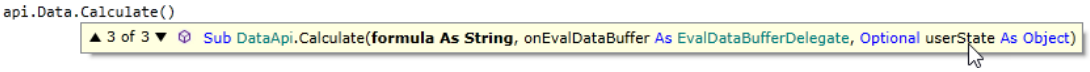
- Api.Data.Calculate en utilisant la formule en tant que chaîne, les fonctions de surcharge, la fonction Eval, et IsDurableCalculatedData



- Api.Data.Calculate en utilisant la formule comme chaîne et IsDurableCalculatedData



- Api.Data.Calculate en utilisant la formule comme chaîne de caractères et la fonction Eval



Fonction de surcharge

La fonction la plus courante est la fonction Api.Data.Calculate, qui définit la valeur d'une ou plusieurs valeurs de dimension (côté gauche de la formule) comme étant égale à une autre (côté droit). Les arguments finaux (facultatifs) sont ajoutés à la formule pour les fonctions de surcharge, les évaluations et les données durables.

La fonction Api.Data.Calculate doit respecter les règles d'explosion des données, ce qui signifie que les côtés gauche et droit des formules sont équilibrés avec les mêmes valeurs de dimension des deux côtés. Si un membre est spécifié pour une dimension n'importe où dans le côté droit de l'équation, vous devez explicitement spécifier quelque chose pour cette dimension dans le côté gauche de l'équation.

Cette variante de la fonction Api.Data.Calculate fournit des filtres de membre (tous facultatifs) qui peuvent être utilisés pour filtrer les résultats avant de les enregistrer dans la cible ou la destination. Cette fonction est la plus puissante des fonctions Api.Data.Calculate car elle vous permet de filtrer les intersections. En outre, la fonction Eval ajoute la possibilité de filtrer le nombre de cellules de données individuelles traitées par des attributs de cellule de données tels que CellAmount ou CellStatus.

Cette fonction est couramment utilisée pour filtrer la mémoire tampon des données source par membres de base d'une dimension liée au compte. Par exemple, le chiffre d'affaires A peut être la mémoire tampon des données source, mais il n'est pas nécessaire d'avoir tous les produits pour le calcul. Au lieu de cela, le chiffre d'affaires A peut devoir être calculé par les membres de base des clubs. En utilisant Clubs.Base pour le chiffre d'affaires A, la mémoire tampon des données du chiffre d'affaires A a été réduite pour n'inclure que Clubs.Base. .

Api.Data.Calculate Utilisation

Exemple d'utilisation de la fonction de surcharge dans une formule de travail :

```
'Add a Formula and use API.Data.Calculate with a filter on UD2 (product) so that
'A#[ClubsSalesCalc] = the A#60000 account (Operating Sales) For just the base products under UD2#Clubs
'Hint: api.Data.Calculate("A#[ClubsSalesCalc] = A#60000",,,,,,"UD2 MEMBER FILTER GOES HERE")
'Formula will run at the base and parent levels

If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then
    api.Data.Calculate("A#ClubsSalesCalc = A#60000",,,,,,"U2#Clubs.Base")
End If
```

▲ 1 of 3 ▼ Sub DataApi.Calculate(formula As String, Optional accountFilter As String, Optional flowFilter As String, Optional originFilter As String, Optional icFilter As String, Optional ud1Filter As String, **Optional ud2Filter As String**, Optional ud3Filter As String, Optional ud4Filter As String, Optional ud5Filter As String, Optional ud6Filter As String, Optional ud7Filter As String, Optional ud8Filter As String, Optional onEvalDataBuffer As EvalDataBufferDelegate, Optional userState As Object, Optional isDurableCalculatedData As Boolean)

IsDurableCalculatedData

Cette variante de Api.Data.Calculate vous permet de définir si les données sont durables ou non. Les données durables ne sont pas effacées automatiquement lorsqu'une unité de données est recalculée. Elles ne peuvent être effacées qu'en appelant api.Data.ClearCalculatedData avec la propriété booléenne clearDurableCalculatedData définie sur Vrai. Dans le cadre de la séquence de calcul standard qui s'exécute lors d'un calcul ou d'une consolidation, les données durables sont ignorées lors du traitement de l'effacement, à moins que l'effacement ne soit spécifiquement défini dans la règle de gestion ou la formule de membre.

La raison la plus courante pour laquelle IsDurableCalculatedData est fixé à Vrai est l'ensemencement. Dans le cadre du premier ensemencement, l'objectif peut être

d'ensemencer d'un scénario à un autre une seule fois et de ne plus jamais l'ensemencer. Dans ce cas, les données ensemencées ne doivent être effacées à aucun moment du processus de calcul ou de consolidation. Cette technique est couramment utilisée dans les processus de budget ou de prévision lorsque vous exécutez l'ensemencement par le biais d'un tableau de bord. La formule peut être appliquée en tant que FinanceFunctionType.CustomCalculate ou FinanceFunctionType.Calculer dans une règle de gestion.

Utilisation IsCurableCalculatedData

Exemple d'utilisation de IsDurableCalculatedData dans une formule de travail :

```
Case Is = FinanceFunctionType.CustomCalculate

    'Define a unique Function Name that will be passed into Custom Calculate process
    If args.CustomCalculateArgs.FunctionName.XFEqualsIgnoreCase("CopyScenario") Then

        'Declare variables that will be passed into api.Data.Calculate.
        'Selected values from parameters will be passed into api.Data.Calculate formula
        Dim selectedTime As String = args.CustomCalculateArgs.NameValuePairs("SelectedTime")
        Dim sourceScenario As String = args.CustomCalculateArgs.NameValuePairs("SourceScenario")
        Dim targetScenario As String = args.CustomCalculateArgs.NameValuePairs("TargetScenario")

        'Only run for base entities and local currency
        If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then
            'Using api.Data.Calculate function with formula and IsDurableCalculatedData set to TRUE As Boolean.
            'Can use filters as well. Use RemoveNoData function or EVAL to eliminate processing data cells with NODATA
            api.Data.Calculate("S#[" & targetScenario & "]:T#[" & selectedTime & "] = RemoveNoData(S#[" & sourceScenario & "]:T#[" & selectedTime & "])",True)
        End If

    End If
```

Fonction Eval

Eval est une fonction avancée qui vous permet d'accéder aux cellules de données individuelles dans toute unité de données créée lors du traitement d'un script api.Data.Calculate. Elle permet d'envelopper Eval() autour d'un sous-ensemble de mathématiques de la formule afin d'évaluer la mémoire tampon de données qui vient d'être créée par l'exécution de ces mathématiques.

Avant la version 5.0 et l'introduction de la fonction RemoveNoData, Eval était couramment utilisé pour évaluer des cellules de données individuelles dans un tampon

de données source à traiter en fonction de la quantité ou de l'état de la cellule. L'évaluation du nombre de cellules sans données pour une unité de données est un facteur important pour la performance et l'efficacité des calculs.

Eval était initialement une fonction importante pour évaluer les cellules de données individuelles, mais elle a été remplacée par des techniques plus récentes telles que GetDataBuffer et GetDataBufferUsingFormula, et le bouclage à travers les cellules dans le tampon de données, ainsi que les fonctions Remove.

Utilisation de la fonction Eval

Exemple d'utilisation de la fonction Eval dans une formule de travail :

```
Formula Footer...
Helper Functions Header...
Private Sub OnEvalDataBuffer (ByVal api As FinanceRulesApi, ByVal evalName As String, ByVal eventArgs As EvalDataBufferEventArgs)
    Try

        'Start with and empty list of result cells.
        'Good practice - Clear out DataBufferResults before executing

        eventArgs.DataBufferResult.DataBufferCells.Clear()

        'Loop over the source cells and assign a bonus % based on level
        For Each sourceCell As DataBufferCell In eventArgs.DataBuffer1.DataBufferCells.Values

            'Only get source cells that have data and are greater than or equal to 0
            If (Not sourceCell.CellStatus.IsNoData) And (sourceCell.CellAmount >= 0.00) Then
                'Create new data buffer cells with the filtered data cells
                Dim resultCell As New DataBufferCell(sourceCell)
                    'Condition if Level is greater than or equal to 1 and less than 2
                    If (sourceCell.CellAmount >= 1.00) And (sourceCell.CellAmount < 2.00) Then
                        'Return 10% to multiply by Salary or A#50200
                        resultCell.CellAmount = 0.10

                        'Condition if Level is greater than or equal to 2 and less than 3
                    Else If (sourceCell.CellAmount >= 2.00) And (sourceCell.CellAmount < 3.00) Then
                        'Return 20% to multiply by Salary or A#50200
                        resultCell.CellAmount = 0.20

                        'Condition if Level is greater than or equal to 3 and less than 4
                    Else If (sourceCell.CellAmount >= 3.00) And (sourceCell.CellAmount < 4.00) Then
                        'Return 30% to multiply by Salary or A#50200
                        resultCell.CellAmount = 0.30

                    Else 'All other conditions
                        'Return 5% to multiply by Salary or A#50200
                        resultCell.CellAmount = 0.05

                    End If
                    'Set the final results of the data cells for the Data Buffer
                    eventArgs.DataBufferResult.SetCell(api.SI, resultcell, False)

            End If
        Next

        Catch ex As Exception
        Throw ErrorHandler.LogWrite(api.SI, New XFException(api.SI, ex))
    End Try
End Sub

Helper Functions Footer...
```

Résumé

La fonction Api.Data.Calculate est le moyen le plus facile et le plus simple d'écrire une formule en tant que formule de membre ou règle de gestion. La construction d'une formule Api.Data.Calculate doit être équilibrée de chaque côté de la formule avec les dimensions appropriées afin d'éviter l'explosion des données. Il existe trois façons différentes d'utiliser la fonction Api.Data.Calculate : Formule avec surcharge, Formule avec IsDurableCalculatedData et Formule avec évaluation.

Du point de vue des performances :

1. N'utilisez jamais la fonction Api.Data.Calculate dans une boucle lorsque vous utilisez des variables.
2. Utilisez les fonctions Remove dans la mesure du possible, en particulier pour les modèles de données éparses comportant de nombreuses cellules NODATA.
3. Les fonctions GetDataBuffer et GetDataBufferUsingFormula peuvent avoir un meilleur impact sur les performances Essayez de remplacer Api.Data.Calculate par GetDataBuffer lorsque vous effectuez des calculs Dans certains cas, les performances sont meilleures en utilisant les fonctions GetDataBuffer à la place de Api.Data.Calculate.

Fonctions Remove

Les fonctions Remove ont été introduites dans la version 5.0 . Elles ont remplacé les raisons d'utiliser la fonction Eval. Le besoin fondamental de la fonction Eval était d'évaluer les cellules de données individuelles dans un tampon de données source afin d'appliquer une logique de traitement. Dans de nombreux cas, OneStream ne voulait pas traiter les cellules de données dans les tampons de données source dont l'état de la cellule était NODATA ou le montant de la cellule = 0. Avec la version 5.0, les fonctions permettent de le faire sans qu'il soit nécessaire d'écrire une logique supplémentaire.

Les fonctions **RemoveNoData** et **RemoveZeros** permettent de ne pas traiter des cellules de données individuelles dans un tampon de données source. Elles enveloppent la fonction Remove() autour d'un sous-ensemble de la formule afin d'empêcher le traitement de cellules de données individuelles à l'intérieur d'une mémoire tampon de données source Les fonctions Remove sont utilisées dans les formules de membre ou les règles de gestion.

Les fonctions Remove sont utilisées pour des raisons de performance Les unités de données peuvent contenir un grand nombre de cellules de données NODATA ou de cellules de données de valeur 0. Ces cellules peuvent être traitées inutilement pendant l'exécution du calcul si ces fonctions ne sont pas utilisées dans une formule Api.Data.Calculate.

RemoveZeros

RemoveZeros permet de supprimer de la mémoire tampon des données source les cellules de données dont la quantité est égale à 0. En outre, cette fonction permet de supprimer de la mémoire tampon des données source les cellules de données dont l'état

est NODATA. Il est important d'évaluer si les 0 sont nécessaires à la formule Api.Data.Calculate lors de l'exécution du calcul.

RemoveNoData

La fonction RemoveNoData supprime du tampon de données source les cellules de données dont l'état est NODATA SEULEMENT. Contrairement à la fonction RemoveZeros, cette fonction ne supprime pas les cellules de données dont l'état est 0.

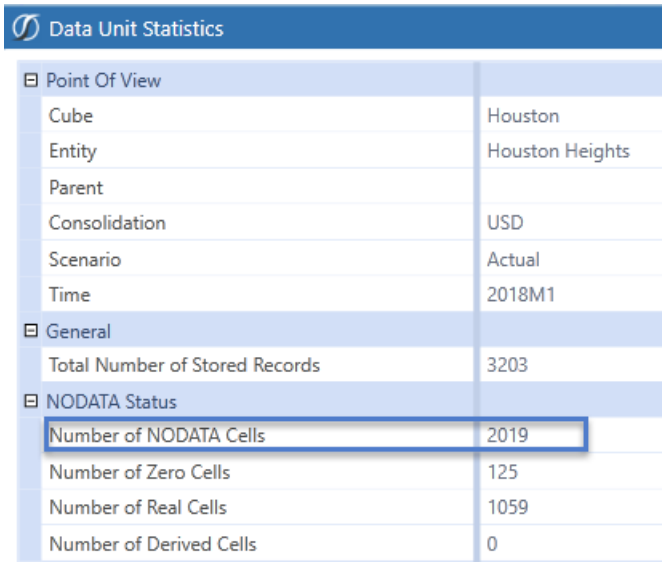
Les cellules NODATA et les cellules 0 peuvent être trouvées en utilisant les méthodes suivantes :

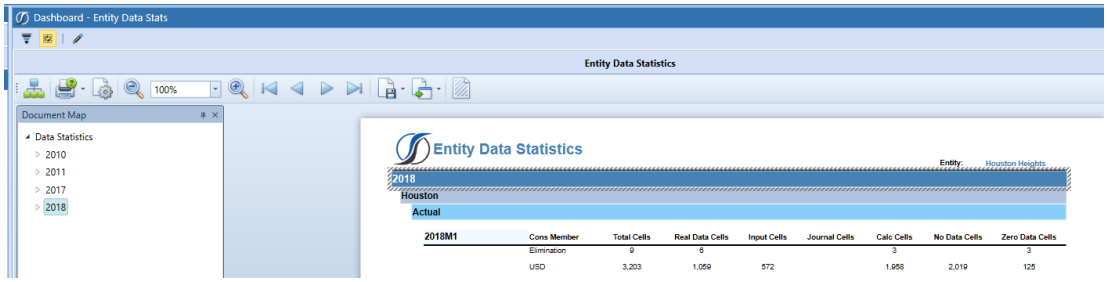
1. Examinez les statistiques de l'unité de données lorsque vous cliquez avec le bouton droit de la souris sur une cellule dans la vue en cube.
2. Consultez le tableau de bord de l'analyse des applications et vérifiez le rapport sur les statistiques des données d'entité.

Ce rapport est basé sur les statistiques de l'unité de données et des données de l'entité. Il peut y avoir de nombreuses formules de membres et règles de gestion qui conduisent à la création de données. Par conséquent, toutes les formules doivent être évaluées pour déterminer si ces fonctions de suppression sont utilisées Plus le pourcentage de cellules NODATA par rapport au nombre total d'enregistrements stockés est élevé, plus il est important d'utiliser ces fonctions de suppression.

Exemple = 3 203 enregistrements stockés dont 2 019 sont des cellules NODATA Près de 65 % de l'unité de données a des cellules NODATA à traiter, ce qui entraîne un temps de calcul supplémentaire.

Les fonctions Review se trouvent dans les fonctions clés, sous la rubrique « Snippets ».





Supprimer l'utilisation des fonctions

Exemple d'utilisation de RemoveZeros dans une formule de travail :

```
'Declare variable To Get period number From the current time period
Dim curMonth As Integer = api.Time.GetPeriodNumFromId(api.Pov.Time.MemberId)
'Run for Entity Base Members Only
If (Not api.Entity.HasChildren()) Then
    'Check to see if current month is M1.
    'If so, pull Ending Balances From M12 prior year.  We are using F#None for this exercise
    'If M2 - M12, pull Ending Balances or F#None from prior period in current year
    'Only run the calculation for Balance Sheet base accounts
    'Remove data cells with cell amount of 0 and cell status of NoData
    If curMonth = 1 Then
        api.Data.Calculate("F#BegBalCalcRemove= RemoveZeros(F#None:T#PovPriorYearM12)","A#[Balance Sheet].Base")
    Else
        api.Data.Calculate("F#BegBalCalcRemove = RemoveZeros(F#BegBalCalc:T#PovPrior1)","A#[Balance Sheet].Base")
    End If
End If
```

Exemple d'utilisation de RemoveNoData dans une formule de travail :

```
'Declare variable to get period number from the current time period
Dim curMonth As Integer = api.Time.GetPeriodNumFromId(api.Pov.Time.MemberId)
'Run for Entity Base Members Only
If (Not api.Entity.HasChildren()) Then
    'Check to see if current month is M1.
    'If so, pull Ending Balances From M12 prior year.  We are using F#None for this exercise
    'If M2 - M12, pull Ending Balances or F#None from prior period in current year
    'Only run the calculation for Balance Sheet base accounts
    'Remove data cells with cell status of NoData ONLY
    If curMonth = 1 Then
        api.Data.Calculate("F#BegBalCalcRemove= RemoveNoData(F#None:T#PovPriorYearM12)","A#[Balance Sheet].Base")
    Else
        api.Data.Calculate("F#BegBalCalcRemove = RemoveNoData(F#BegBalCalc:T#PovPrior1)","A#[Balance Sheet].Base")
    End If
End If
```

Fonctions GetDataBuffer

Il se peut qu'un script membre ne soit pas défini pour la fonction Api.Data.Calculate parce que plusieurs cellules de données, qui ne semblent avoir aucun rapport entre elles, sont traitées et qu'aucun des membres de la dimension n'est constant Dans ce cas, utilisez les fonctions GetDataBuffer et SetDataBuffer.

Les fonctions GetDataBuffer et SetDataBuffer sont plus fondamentales que l'utilisation d'une fonction Eval. Elles vous permettent de lire des nombres à l'aide d'un script membre, de traiter ou de modifier chaque cellule du résultat, puis d'enregistrer les modifications. Les fonctions GetDataBuffer courantes sont les suivantes :

- GetDataBuffer
- GetDataBufferForCustomShareCalculation
- GetDataBufferForCustomElimCalculation
- GetDataBufferUsingFormula
- SetDataBuffer

Lors de l'utilisation des fonctions api.Data.Calculate, il est important de savoir à quel membre une formule est rattachée. Par exemple, si la formule commence par Api.Data.Calculate(« A#Vente1 = ... »), placez la formule dans le paramètre Formule du membre du compte Account Reconciliations.

Cependant, lors de l'utilisation des fonctions GetDataBuffer, la formule peut ne pas être écrite dans un membre spécifique. Chaque cellule de données sauvegardée peut être écrite dans un membre de dimension différente. Dans ce cas, la logique peut être développée dans une règle de métier et peut être créée en tant que sous-programme à exécuter dans les règles de métier de finance.

GetDataBuffer Fonction

GetDataBuffer récupère les valeurs d'une unité de données au cours d'une consolidation, d'un calcul ou d'une conversion particulière. Lorsque vous l'utilisez GetDataBuffer, cela équivaut au tampon de données source ou au côté droit de l'équation pour Api.Data.Calculate. Selon la fonction GetDataBuffer que vous utilisez, trois ou quatre propriétés peuvent être utilisées.

Pour la fonction de base GetDataBuffer, trois propriétés sont utilisées :

- ScriptMethodType As DataApiScriptMethodType
- SourceDataBufferScript As String
- ExpressionDestinationInfo As ExpressionDestinationInfo

La propriété scriptMethodType utilise généralement l'option Calculate de la propriété DataApiScriptMethodType.

Le sourceDataBufferScript est équivalent au côté droit de l'équation pour le Api.Data.Calculate.

Le expressionDestinationInfo est équivalent au côté gauche de l'équation pour le Api.Data.Calculate. Fréquemment, ceci est manipulé en utilisant l'Id de la dimension, en passant dans l'Id du membre de la dimension pour la clé primaire du tampon de données.

Le GetDataBuffer peut être utilisé de différentes manières, et n'est pas limité à ce qui suit :

1. Utiliser les Data Buffers pour effectuer des calculs dans les tampons de données. Dans certains cas, cela peut être plus performant qu'un Api.Data.Calculate.

2. Utiliser GetDataBuffer à la place de Api.Data.Calculate pour les routines secondaires qui exécutent le code et les instructions, sont stockés dans la mémoire et sont utilisés dans les fonctions des règles de métier de finance.

Utilisation GetDataBuffer

Exemple d'utilisation de GetDataBuffer avec Data Buffer Math dans une formule de travail :

```
'Alternate way to api.Data.Calculate("A#DataBufferMath:UD2#None = A#60999:UD2#Top - A#54500:UD2#Top").  May have better performance impact.

'Run only for Local Currency and Base Entities
If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then

    'Declare Variable for Destination Buffer
    Dim destinationInfo As ExpressionDestinationInfo = api.Data.GetExpressionDestinationInfo("A#DataBufferMath:UD2#None")

    'Get Source Data Buffer for Net Sales
    Dim netSales As DataBuffer = api.Data.GetDataBuffer(DataApiScriptMethodType.Calculate,"RemoveNoData(A#60999:UD2#Top)", destinationInfo)

    'Get Source Data Buffer for Operating Expenses
    Dim operatingExpenses As DataBuffer = api.Data.GetDataBuffer(DataApiScriptMethodType.Calculate,"RemoveNoData(A#54500:UD2#Top)", destinationInfo)

    'Create New Data Buffer With the End Result of Net Sales - Operating Expenses
    Dim dataBufferExample As DataBuffer = (netSales - operatingExpenses)

    'Set the Destination Data Buffer
    api.Data.SetDataBuffer(dataBufferExample, destinationInfo)

End If
```

Exemple d'utilisation de GetDataBuffer et de SetDataBuffer dans Business Rule Using Sub Routine dans une formule de travail :

```
    Case Is = FinanceFunctionType.Calculate

        'Execute Sub Routine in the Function to Return Results
        Me.CalculateBonusUsingGetDataBuffer(api)
```

```
Private Sub CalculateBonusUsingGetDataBuffer(ByVal api As FinanceRulesApi)
    Try
        'Define Destination Data Buffer or left side of the equation
        'Will copy to A#Bonus while processing the data buffer in memory
        Dim destinationInfo As ExpressionDestinationInfo = api.Data.GetExpressionDestinationInfo("")
        'Read the numbers for A#Salary into a source Data Buffer
        Dim sourceDataBuffer As DataBuffer = api.Data.GetDataBuffer(DataApiScriptMethodType.Calculate, "A#Salary", destinationInfo)

        'Check to make sure the source Data Buffer exists
        If Not sourceDataBuffer Is Nothing Then
            'Create a new data buffer for the result cells
            Dim resultDataBuffer As DataBuffer = New DataBuffer()

            'Loop over the source cells in the source Data Buffer
            For Each sourceCell As DataBufferCell In sourceDataBuffer.DataBufferCells.Values
                'Only process cells that have data and cell amount that is greater than 0
                If ((Not sourceCell.CellStatus.IsNoData) And (sourceCell.CellAmount > 0.00)) Then
                    'Create new data buffer cells from the filtered source cells from source Data Buffer
                    Dim resultCell As New DataBufferCell(sourceCell)

                    'Define A#Bonus as the target account to copy to
                    'Multiply data cell amounts by 5%
                    'Set the manipulated data cells to the data buffer
                    resultCell.DataBufferCellPk.AccountId = api.Members.GetMemberId(DimType.Account.Id, "Bonus")
                    resultCell.CellAmount = sourceCell.CellAmount * 0.05
                    resultDataBuffer.SetCell(api.SI, resultCell)

                End If
            Next

            'Save the results to the destination data buffer
            api.Data.SetDataBuffer(resultDataBuffer, destinationInfo)

        End If

            Catch ex As Exception
        Throw ErrorHandler.LogWrite(api.si, New XFException(api.si, ex))
    End Try
End Sub
```

Fonctions mathématiques non équilibrées

Fonctions mathématiques non équilibrées

Les fonctions mathématiques non équilibrées sont nécessaires pour effectuer des calculs avec deux tampons de données, lorsque le deuxième tampon de données doit spécifier une dimensionnalité supplémentaire. Le terme « déséquilibré » est utilisé parce que le script du deuxième tampon de données peut représenter un ensemble de dimensions différent de celui de l'autre tampon de données dans le texte api.Data.Calculate. Ces fonctions empêchent l'explosion des données. Les quatre fonctions mathématiques déséquilibrées sont les suivantes :

- AddUnbalanced
    - Exemple : api.Data.Calculate(« A#TargetAccount = AddUnbalanced (A#OperatingSales, A#DriverAccount:U2#Global, U2#Global) »)
- SubtractUnbalanced
    - Exemple : api.Data.Calculate(« A#TargetAccount = SubtractUnbalanced (A#OperatingSales, A#DriverAccount:U2#Global, U2#Global) »)
- MultiplyUnbalanced
    - Exemple : api.Data.Calculate(« A#TargetAccount = MultiplyUnbalanced (A#OperatingSales, A#DriverAccount:U2#Global, U2#Global) »)
- DivideUnbalanced

- Exemple : api.Data.Calculate(« A#TargetAccount =DivideUnbalanced (A#OperatingSales, A#DriverAccount:U2#Global, U2#Global) »)

Lors de l'utilisation de fonctions mathématiques non équilibrées, les deux premiers paramètres représentent les premier et deuxième tampons de données sur lesquels la fonction doit être exécutée. Le troisième paramètre représente les membres à utiliser à partir du deuxième tampon de données lors de l'exécution des mathématiques avec chaque intersection dans le premier tampon de données. Les calculs privilégient les intersections du premier tampon de données sans créer d'intersections supplémentaires.

Il est important que la dimensionnalité de la cible (côté gauche de l'équation) corresponde à la dimensionnalité du premier tampon de données du côté droit de l'équation (argument 1).

Ces fonctions sont souvent utilisées lorsqu'un tampon de données source effectue des calculs avec une intersection de cellules de données spécifique. Il peut s'agir d'un taux, d'un conducteur ou d'une entrée de cellule de données.

Utilisation des fonctions mathématiques non équilibrées

Exemple d'utilisation de MultiplyUnbalanced dans une formule de travail :

```
'Calculate Salary (A#50200) times Bonus Percent to create Bonus number.  Use MultiplyUnbalanced formula to calculate.
'Use a Technique to Not Process No Data Cells and 0 Data Cells for Salary account
'1st property is the data buffer with the least dimensions and matches dimensionality of destination data buffer. Follow Data Explosion rules
'2nd Property is the data buffer with the most dimensions
'3rd Property is the list of account related dimensions that make it unbalanced

'Run for only Base Entities and Local Currency
If ((Not api.Entity.HasChildren()) And (api.Cons.IsLocalCurrencyforEntity())) Then
    api.Data.Calculate("A#BonusUnbalanced = MultiplyUnbalanced(RemoveZeros(A#50200), A#BonusPercent:F#None:O#Forms:I#None:U2#None:U3#None:U4#None:U5#None:U6#None:U7#None:U8#None, F#None:O#Forms:I#None:U2#None:
End If
```

Fonction GetDataBufferUsingFormula

La fonction GetDataBufferUsingFormula utilise une expression mathématique entière pour calculer un tampon de données final. GetDataBufferUsingFormula peut effectuer les mêmes calculs de tampon de données que Api.Data.Calculate, mais le résultat est affecté à une variable, alors que Api.Data.Calculate enregistre réellement les données calculées.

GetDataBufferUsingFormula calcule d'abord plusieurs tampons de données sources , puis le résultat des calculs est stocké en mémoire à l'aide d'une variable de formule. Enfin, la variable de formule est utilisée n'importe où dans la formule de membre ou la règle de gestion. Cette fonction est couramment utilisée lors de la rédaction des règles pour les règles de métier de la planification utilisant MultiplyUnbalanced, DivideUnbalanced, les fonctions de suivi telles que les 12 derniers mois et les allocations.

Lors de l'utilisation de GetDataBufferUsingFormula, FilterMembers et RemoveMembers sont utilisés conjointement pour réduire les membres dimensionnels dans le tampon de données source.

FilterMembers

FilterMembers modifie un tampon de données et n'inclut que les numéros des dimensions spécifiées. Le premier paramètre est le tampon de données de départ. Il peut s'agir d'un nom de variable ou d'une équation mathématique entière entre parenthèses. Il peut y avoir autant de paramètres que nécessaire pour spécifier les filtres de membres et différents filtres de membres peuvent être utilisés pour plusieurs types de dimensions. Le tampon de données filtré résultant ne contiendra que les nombres qui correspondent aux membres dans les filtres.

GetDataBufferUsingFormula Utilisation

Exemple d'utilisation de GetDataBufferUsingFormula dans une formule de travail :

```
'Alternate way to api.Data.Calculate("A#DataBufferMathUsingFormula:UD2#None = A#60999:UD2#Top - A#54500:UD2#Top").  May have better performance impact using
'GetDataBufferUsingFormula

    'Standard GetDataBufferUsingFormula formula
    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#60999:UD2#Top) - RemoveNoData(A#54500:UD2#Top)")
        '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 $
        'Can pass this variable to other api.Data.Calculate, GetDataBufferUsingFormula, or Sub routines
        api.Data.Calculate("A#DataBufferMathUsingFormula:UD2#None = $dataBufferExample")

    End If
```

Exemple d'utilisation de GetDataBufferUsingFormula avec FilterMembers et MultipleUnbalanced dans une formule de travail :

```
'Use Data Buffer Using Formula to filter specific members
'1st argument inside () is the starting data buffer
'2nd argument is the filter based on the starting data buffer
'Can continue to add filters separated by a comma
Dim salesExp As DataBuffer = api.Data.GetDataBufferUsingFormula("RemoveZeros(FilterMembers(A#All,A#TotalExp.Base))")

'Set Data Buffer Variable to pass salesExp to any other formula
api.Data.FormulaVariables.SetDataBufferVariable("salesExp", salesExp, False)

'Use MultiplyUnbalanced to multiply all Expense Accounts by a specific data cell intersection and divide by 12
'1st argument is Formula Variable multiplied by 2nd argument which is an individual data cell intersection
'3rd argument is the dimensions that make it unbalanced
Dim result As DataBuffer = api.Data.GetDataBufferUsingFormula("MultiplyUnbalanced($salesExp, (E#Global:V#YTD:A#RateAccount:C#USD:F#None:O#BeforeAdj:I#None:U1#None:U2#None:U3#None:U4#None/12), E#Global:V#YTD:C#USD:F#None:O#BeforeAdj

'Set Data Buffer Variable to pass result to any other member formula
api.Data.FormulaVariables.SetDataBufferVariable("result", result, True)

'Calculate using Data Buffer Variable.  Can do additional math inside api.Data.Calculate
api.Data.Calculate("V#Periodic = $result")
```