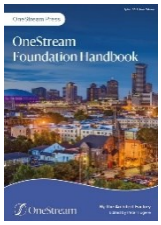

OneStream Foundation Handbook Errata

Chapter 3 | Design and Build | page 65; section Paired Cubes | amended in PDF copies from April 14 2021

clarification:

Since Dimensions can only exist in one Cube, there needs to be a common Entity Dimension that has all Entities in every Cube. > Since **Dimension members** can only exist…

Chapter 3 | Design and Build | page 59 | Data Unit Statistics section.

update:

See Figure 3.2, not Figure 3.1

Chapter 4 | Consolidation | page 86; section Stored Share | amended in PDF copies from July 7 2021

should be:

Stored Share

The Stored Share Consolidation Algorithm Type stores the amounts for Share rather than calculating them dynamically as in Standard. However, the eliminations under Stored Share are still calculated as in Standard, using OneStream's built-in algorithms.

This Consolidation Algorithm Type may be used when you have the need for different logic in calculating Share than is performed in Standard (defined above). An example of this would be if you had a minority interest calculation, where the Share contribution cannot be driven from the percent Consolidation.

When the Stored Share Consolidation Algorithm Type is used, the rules to calculate the value need to be written under the Finance Function Type `Calculate` (or `CustomCalculate`).

```
Case Is = FinanceFunctionType.Calculate
```

```
            Select Case api.FunctionType
                Case Is = FinanceFunctionType.Calculate

                    If api.Pov.Cons.Name.XFEqualsIgnoreCase("Share") Then
                        CalcMinInt(si, api, args)
                    End If

            End Select

            Return Nothing
        Catch ex As Exception
            Throw ErrorHandler.LogWrite(si, New XFException(si, ex))
        End Try
    End Function

#Region "Share Rules"
    Private Sub CalcMinInt(ByVal si As SessionInfo, ByVal api As FinanceRulesApi, ByVal args As FinanceRulesArgs)
        Try
            'Constants used in calcs
            Const All_None = ":F#EndBal:I#None:U1#Loc_000:U2#None:U3#Input:U4#Input:U5#None"
            Const All_Top = ":F#Top:I#Top:U1#Loc_ALL:U2#AllDepts:U3#USGAAP_PreEquity:U4#wAlloc:U5#None"

            Dim entity As String = api.Pov.Entity.Name
            Dim entityId As Integer = api.Pov.Entity.MemberId

            'Grabs the current calculated entity's ownership % and type
            Dim pOwn As Decimal = api.Entity.PercentOwnership(entityId)
            Dim ownType As Integer = api.Entity.OwnershipType(entityId)

            'Performs the calculation based on the ownership type of the entity
            Select Case ownType     'Ownership Type:
                Case Is = 1000      'Full Consolidation
                    api.ExecuteDefaultShare()
                Case Is = 2000      'Holding
                    api.ExecuteDefaultShare()
                Case Is = 4000      'Equity
                    api.ExecuteDefaultShare()
                Case Is = 7000      'Non-controlling Interest
                    api.Data.Calculate("A#808000:O#Elimination:C#Share:V#Periodic" & All_None & " = " & _
                    "A#NETINC:O#BeforeElim:C#Local:V#Periodic" & All_Top & " * ((100 - " & pOwn & ") * .01)")

                    api.Data.Calculate("A#310000:O#Elimination:C#Share:V#YTD" & All_None & " = " & _
                    "A#310000:O#Elimination:C#Share:V#YTD:T#POVPriorYearM12" & All_None & " " & _
                    "+ A#808000:O#Elimination:C#Share:V#YTD" & All_None & "")

                    api.Data.Calculate("A#CYNI:O#Elimination:C#Share:V#YTD" & All_None & " = " & _
                    "- A#310000:O#Elimination:C#Share:V#YTD" & All_None)

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

Chapter 4 | Consolidation | page 95; section Equity Pickup | amended in PDF copies from January 18 2022

should be:

```
If (Not api.Entity.HasChildren) And (api.Cons.IsLocalCurrencyForEntity) Then 'Base Entities and Local Currency
    Dim destinationInfo As ExpressionDestinationInfo = api.Data.GetExpressionDestinationInfo("")
    'Get a data buffer of all ownership percentages
    Dim pShOwn As DataBuffer = api.Data.GetDataBufferUsingFormula("RemoveZeros" & _
    "(Cb#Golfstream:S#Actual:A#PShOwn:V#YTD:O#BeforeAdj:U1#None:U2#None:U3#None:U4#None:U5#None:U6#None:U7#None:U8#None)",,False)

    If pShOwn.DataBufferCells.Count > 0 Then
        Dim resultDataBuffer As DataBuffer = New DataBuffer()
        For Each cell As DataBufferCell In pShOwn.DataBufferCells.Values

            'in Golfstream:     account 17800 = investment account on balance sheet
                            '   account 65000 = income from subsidiary account on P&L
                            '   account 69000 = net income

            'Get the investees NI
            Dim investeeNI As Decimal = api.Data.GetDataCell("E#[" & cell.DataBufferCellPk.GetICName(api) & "]" & _
            ":A#69000:V#YTD:O#Top:I#Top:U1#Top:U2#Top:U3#Top:U4#Top:U5#DataSource:U6#None:U7#None:U8#None").CellAmount

            If investeeNI <> 0.00 Then  'If the investees NI is not 0
                Dim resultCell17800 As New DataBufferCell(cell)
                resultCell17800.DataBufferCellPk.AccountId = api.Members.GetMember(DimType.Account.Id, "17800").MemberId
                resultCell17800.DataBufferCellPk.OriginId = DimConstants.Import
                resultCell17800.DataBufferCellPk.FlowId = api.Members.GetMember(DimType.Flow.Id, "EndBal").MemberId
                resultCell17800.DataBufferCellPk.ICId = cell.DataBufferCellPk.ICId
                resultCell17800.DataBufferCellPk.UD5Id = api.Members.GetMember(DimType.UD5.Id, "CalcEPU").MemberId

                'Calcualte the change in investment as the ownership % * the investee NI
                resultCell17800.CellAmount = cell.CellAmount * investeeNI
                resultDataBuffer.SetCell(si, resultCell17800, True)

                Dim resultCell65000 As New DataBufferCell(cell)
                resultCell65000.DataBufferCellPk.AccountId = api.Members.GetMember(DimType.Account.Id, "65000").MemberId
                resultCell65000.DataBufferCellPk.OriginId = DimConstants.Import
                resultCell65000.DataBufferCellPk.FlowId = api.Members.GetMember(DimType.Flow.Id, "None").MemberId
                resultCell65000.DataBufferCellPk.ICId = cell.DataBufferCellPk.ICId
                resultCell65000.DataBufferCellPk.UD5Id = api.Members.GetMember(DimType.UD5.Id, "CalcEPU").MemberId

                'Calculate the investor NI as the ownership % * the investee NI
                resultCell65000.CellAmount = cell.CellAmount * investeeNI
                resultDataBuffer.SetCell(si, resultCell65000, True)
            End If
        Next

        api.Data.SetDataBuffer(resultDataBuffer,destinationInfo)

    End If
End If
```

Chapter 5 | Planning | page 150 [Second-last paragraph] | amended in PDF copies from March 31 2021 | amended in all new print copies from April 13 2021

should be:

The Growth Factor can then be applied to all Accounts via an `Api.Data.Calculate` function in the Scenario formula.

```
Api.Data.Calculate("S#FactorBasedForecast = MultiplyUnbalanced(S#Actual *
S#POV:T#POVYearM1:A#GrowthFactor, A#GrowthFactor")
```

Chapter 5 | Planning | page 152

Update to

Location of Seeding Rules

In general, Seeding Rules should run as a Custom Calculate Finance Rule. A few sections ago, I described the differences between Custom Calculate and Data Unit Calculations Sequence Calculations. Since Seeding Rules typically copy data that does not change, they will only need to be run once, or when the source data has changed which will likely not be often. Seeding rules that run in the DUCS will run every time a calculation and consolidation is executed, resulting in unnecessary clearing and re-copying of the same data and suboptimal performance. The Custom Calculate rule can be linked to a Dashboard and executed centrally by an admin or by each user for

their respective set of entities. Remember to use the DurableData = True and always include a ClearCalculateData script!

Chapter 6 | Data Integration | page 175 [Bottom paragraph; Line 1] | amended in PDF copies from March 31 2021 | amended in all new print copies from April 13 2021

should be: Data reloading can be separated by different Source IDs in one Workflow and have multiple channels.

Chapter 10 | Reporting | page 292 [Paragraph 5; Line 1] | amended in PDF copies from March 29 2021 | amended in all new print copies from April 13 2021

should be: These ‘simplistic’ layout tabbed Dashboards can be easily constructed; however, the amount of data in each tab could cause issues when running the Dashboard or when trying to navigate from tab to tab.