Adventures In Groovy – Part 11: Accessing Metadata Properties

Introduction

Groovy opens up a lot of things above and beyond performance improvements and improving the user experience. One example is the possibility to interact with the metadata. Dimensions and members can be queried for all types of things which can be useful in many situations. Is the POV at a level 0? What is the parent of the current POV member? Does the member exist in another application? What about pushing data for specific UDAs and dynamically generating the Data Map? How about dynamically generating the Data Map to ignore dynamic calculated members? These are just some examples to get you thinking about where this could be useful.

Code Example

This article won't get into the logic to accomplish the above examples once the property is identified but will explain how to extract properties for its use. Below is an example of retrieving every property of an account named Regular_Cases. This iterates through every metadata property and writes it to the log.

```
// Get the dimension of the member in question
Dimension AccountDim = operation.application.getDimension("Account")
// Get the member
Member AccountMbr = AccountDim.getMember("Regular_Cases")
// Print the map to the log
println AccountMbr.toMap()
def memberProps = AccountMbr.toMap()
```

```
// Print the member name
println AccountMbr.toString()
// Print every property and corresponding property value
for ( e in memberProps ) {
  println "${e.key} = ${e.value}"
}
```

When this is executed, the following is sent to the log.

println AccountMbr.toMap() produces

> {Formula (rGP)=<none>, Plan Type (GP)=true, Solve Order (rGP)=0, Formula (Fin)=<none>, Data Storage (OEP_WFSC)=never share, Time Balance=flow, Formula=<none>, UDA=HSP_NOLINK, Skip Value=none, Variance Reporting=non-expense, Data Storage (GP)=never share, Essbase Name=Regular_Cases, UUID=c842d186-6d83-4b90-8d1e-49474a6a8a1d, Member=Regular_Cases, Data Storage=never share, Data Storage (rFin)=never share, Formula (rFin)=<none>, Aggregation (rWFP)=+, Formula (GP)=<none>, Data Storage (rWFP)=never share, Data Storage (OEP_REP)=never share, Data Storage (rGP)=never share, Data Type=currency, Formula (OEP_WFP)=<none>, Plan Type (rFin)=true, Aggregation (OEP_WFP)=+, Data Storage (OEP_WFP)=never share, Parent=GP_Accts, Two Pass Calculation=false, Aggregation (GP)=+, Plan Type (rGP)=true, Process Management Enabled=true, Plan Type (rWFP)=false, Source Plan Type=GP, Aggregation (OEP_WFSC)=+, Exchange Rate Type=none, Plan Type (Fin)=true, Alias: English=Regular Cases, Plan Type (OEP_WFP)=false, Aggregation (OEP_REP)=+, Solve Order (rWFP)=0, Data Storage (Fin)=never share, Hierarchy Type=dynamic, Allow Upper Level Entity Input=false, Account Type=revenue, Formula (OEP_REP)=<none>, Aggregation (Fin)=+, Aggregation (rGP)=+, Plan Type (OEP_WFSC)=false, Formula (rWFP)=<none>, Formula Description=<none>, Aggregation (rFin)=+, Solve Order (rFin)=0, Formula (OEP_WFSC)=<none>, Solve Order (OEP_REP)=0, Valid For Consolidations=false, Plan Type (OEP_REP)=false}

for (e in memberProps) {println “${e.key} = ${e.value}”}
produces

```
Regular_Cases
Formula (rGP) = <none>
Plan Type (GP) = true
Solve Order (rGP) = 0
Formula (Fin) = <none>
Data Storage (OEP_WFSC) = never share
Time Balance = flow
Formula = <none>
UDA = HSP_NOLINK
Skip Value = none
Variance Reporting = non-expense
Data Storage (GP) = never share
Essbase Name = Regular_Cases
UUID = c842d186-6d83-4b90-8d1e-49474a6a8a1d
Member = Regular_Cases
Data Storage = never share
Data Storage (rFin) = never share
Formula (rFin) = <none>
Aggregation (rWFP) = +
Formula (GP) = <none>
Data Storage (rWFP) = never share
Data Storage (OEP_REP) = never share
Data Storage (rGP) = never share
Data Type = currency
Formula (OEP_WFP) = <none>
Plan Type (rFin) = true
Aggregation (OEP_WFP) = +
Data Storage (OEP_WFP) = never share
Parent = GP_Accts
Two Pass Calculation = false
Aggregation (GP) = +
Plan Type (rGP) = true
Process Management Enabled = true
Plan Type (rWFP) = false
```

```
Source Plan Type = GP
Aggregation (OEP_WFSC) = +
Exchange Rate Type = none
Plan Type (Fin) = true
Alias: English = Regular Cases
Plan Type (OEP_WFP) = false
Aggregation (OEP_REP) = +
Solve Order (rWFP) = 0
Data Storage (Fin) = never share
Hierarchy Type = dynamic
Allow Upper Level Entity Input = false
Account Type = revenue
Formula (OEP_REP) = <none>
Aggregation (Fin) = +
Aggregation (rGP) = +
Plan Type (OEP_WFSC) = false
Formula (rWFP) = <none>
Formula Description = <none>
Aggregation (rFin) = +
Solve Order (rFin) = 0
Formula (OEP_WFSC) = <none>
Solve Order (OEP_REP) = 0
Valid For Consolidations = false
Plan Type (OEP_REP) = false
Data Storage (GP) = never share
```

Getting A Specific Property

Typically, there would not be a need to pull every property. There might be times when having access to these, however, is useful in calculations. If a currency calculation is being executed, for example, the rate applied is different if the member is a balance sheet account. Getting one value can be retrieved by building on the above script.

```
def keyProp = "Account Type"
if(memberProps[keyProp] = "Revenue"
  {do something}
```

```
elseif(memberProps[keyProp] = "Balance Sheet"
  {do something}
```

Wrap Up

This may seem a little worthless at first, but if you think about all the BSO functions (getting UDAs, Account types for VAR functions, and member relation functions) that require this information, mimicking them in Groovy requires access to the metadata properties. So, don’t underestimate its use for things like variance, currency, and other calculations, that are done outside of Essbase/Planning calculations and member formulas.