Adventures in Groovy – Part 26: Is the POV a level 0 member?

One of the more surprisingly useful things that can be done in a Groovy calculation is querying metadata. This was discussed in Part 11: Accessing Metadata Properties, but I recently had a situation where users could load data at parent levels and have it allocated down to the bottom of the hierarchies. Knowing if users selected a parent member in the hierarchy was critical because the business process was different based on whether it was a parent or a level 0 member. This can obviously be checked in a business rule, but I had the need to alter several functions in the process if the selection was a parent. Smart Pushes and data synchronizations, as well as the business rules that were executed, were dependent on the level of the hierarchy selected. This is possible with Groovy.

The Code Explained

Using a combination of methods, the children of any member can be returned in a list. That list can be used for all kinds of things. This focuses on getting the size of the list to identify if it has children.

```
// Normally getting the POV Selected, but this is hard coded for the example
def povProduct = '"' + "Tents" + '"'
// Setup a connection to the cube
Cube cube = operation.application.getCube("GP")
// Create a dimension object
Dimension productDim = operation.application.getDimension("Product", cube)
// Create a member list - ILev0 of the member passed
def MemberList =
```

```
productDim.getEvaluatedMembers("ILvl0Descendants($povProduct)"
, cube)

println MemberList
println MemberList.size() == 1 // will print true or false
```

The MemberList variable now holds a list of all the members (level 0 members below tents), including the member passed in the function. This will always have at least one item in the list, assuming the member exists in the dimension. If there is more than 1, we know it has children, and therefore, is not a level 0 member.

```
if(MemberList.size() == 1)
  {
  //actions taken if the POV is a level 0 selection
  }
  else
  {
  //actions taken if the POV is NOT a level 0 selection
  }
```

Summary

In applications where top down planning is needed, different logic runs when data is entered at a parent verses when it is entered at a leaf (lev0) member. This can be handled in an Essbase calculation using @ISLEV0, but with Groovy, the fix statement can be altered so that unnecessary logic isn’t executed and data pushes are limited to only what is impacted.

Do you have an idea of how you might benefit from the results of metadata queries? Come up with your own and share with the group by commenting below.