One at a time, please

Introduction

One of the problems with giving users of Hyperion Planning the ability to run calculations is opening up the possibility for all of them to run the same calculation at the same time. This can cause a range of issues, from slower performance, to calculations never finishing due to locked blocks, to crashing the server.

Prior to Planning, I created VB applications to monitor what was calculated to make sure multiple calculations were not executed at the same time. Initiating a calculation through a web portal allowed us to notify the user that the calculation request was ignored because a calculation was already running.

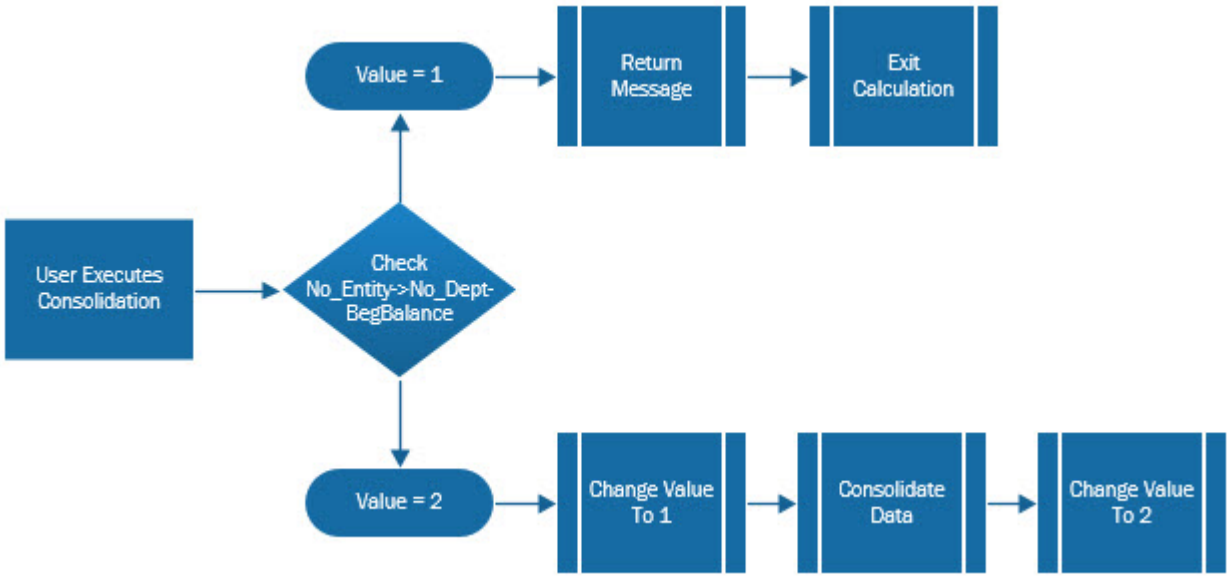
Both Essbase and Planning have come a long way since the 90s. With the introduction of the @RETURN function, developers can interact with users and create a break in a calculation (business rule) so it doesn't proceed. The message is still reactive, but with some creativity, there are some really awesome things you can achieve. Controlling what calculations are executed simultaneously is one of those things.

The Goal

Assume an application has a global consolidation calculation that is required to be executed for reporting requirements. Since the administrators don't want to be bothered at all hours of day and night, they want to enable the users to run the calculation and ensure it isn't run more than one time during the calculation window.

This assumes the 6 required dimensions in Planning, plus a Department dimension.

The Method



Make a predefined placeholder where an indicator can be saved – a 1 or a 2. When the calculation is executed, the value will be set to a 1. When the calculation is finished, the value will be set to 2. When the calculation is initiated, it will check that value. If it is a 2, the calculation will execute. If it is a 1, it assumes a calculations is already running so it will abort and notify the users. This ensures that the calculation will never run twice at the same time.

Note: I prefer the use of 1 and 2 over 0 and 1. Many times a process is implemented to eliminate zeros and restructure the application periodically. Not using a zero can eliminate errors in some situations.

Example

```
FIX("No                              Entity","No_Dept","No
Account","Budget","FY15","BegBalance")    SET CREATEBLOCKONEQ
ON;    "Working"(      /* Check to see if a calculation is
running        If the flat is a 1, return a message and stop
the calculation        If the flag is a 2, continue */
```

```
IF("Working" == 1)         @RETURN ("This calculation is
already running.  Please come back at a
                        later time and try again.", ERROR);
     ELSE          "Working" = 1;       ENDIF)     SET
CREATEBLOCKONEQ OFF;  ENDFIX     /* Aggregate the database */
FIX("Working","Budget","FY15")     AGG("Entity","Department");
ENDFIX       /* Set the flag back to 2 */   FIX("No
Entity","No_Dept","No  Account","Budget","FY15","BegBalance")
   "Working" = 2;    ENDFIX
```

Summary

This method could be used in a variety of situations, not just a global calculation. If this inspires you to use the @RETURN in other ways, please share them with the In2Hyperion and we can make your solution available to everybody.