User Guide – SwapModels Capability

Sometimes, models need a user interface capable of transforming to offer alternate parameter values and calculations based on a user selection. An example might be dramatic switches such as switching a business simuator from “retail” to “online” or a manufacturing simulation from “batch” to “continuous.” In such cases, numerous input parameters and calculations relevant to one selection may be irrelevant and a user distraction for the other.

In Microsoft Excel, the ExcelSteps add-in contains capability for swapping one ExcelSteps Scenario Model[[1]](#footnote-1) for another to do such “personality changes.” The picture shows a mockup configuration with a useful user interface for such swaps.

Model with Swappable SMdlType2/SMdlType1 Scenario Model based on Plant Selector

A screenshot of a computer

Description automatically generated

The selector cell, N5, has a dropdown validation list with options for selecting type of manufacturing plant. This is within a “dashboard” Scenario Model ($I$5 home cell) at the top. It is created with a fixed number of rows -- 6. The dashboard model is not affected by the swap capability. Rather, changes to its selector cell trigger the swap.

The swappable region sits directly below this. It is a second Scenario Model ($I$10 home cell) configured to allow variable row number to handle lots of parameters and calculations. In the ExcelSteps Scenario Model object nomenclature, both models are custom, “Lite” models, which can be positioned where needed on a sheet and which have just four, unlabeled header columns to the left of the single values column, N. The four header columns are Group headings (Column I), Variable Description (J), variable name and units (K and L). In the Lite configuration, an ExcelSteps admin sheet contains the variables’ number formatting and calculation formulas such as for the T\_start\_f variable. The currently-viewable model can be refreshed from this, but the admin sheet is hidden from the user. In general, ExcelSteps Scenario Models can be multicolumn, but the SwapModels capability is only enabled for “Calculator” models having a single values column such as this example’s column N. That is a good configuration for displaying single-valued parameters to users either for their information or to allow them to make parameter input changes affecting other areas of the project.

When not displayed as a Scenario Model, swappable models are stored in rows/columns format on the tblImport admin sheet. This should also be kept hidden from the user. To keep the user interface uncluttered, xlVeryHidden VBA status is recommended for these admin sheets to make the sheets invisible in the Sheet Hide/Unhide user-facing menus).

The tblImport sheet example below shows two, models in rows/columns format. Note that, the SMdlType2 rows would not be there if swapped as the previous Scenario Model, but this shows the rows/columns version for comparison.

The Model column, A, is the key column. The variables’ Scenario Model order and spacing is determined by tblImport sheet row order and by inserting optional “<blank>” tags in the Variable Name column E to specify inserting a blank row in the Scenario Model. The Number Fmt and Formula/Row Type columns are formatted as Excel Text format to store number format strings and formula strings, respectively. Formula strings rely on range names that are set automatically when a Scenario Model is refreshed. The Formula/Row Type column can also be used to specify adding a dropdown validation list for a variable –not shown in this example.

For wayfinding, swappable models are required to contain the mdl\_name variable shown in the example in Rows 2 and 6. This can be in any variable order within the model. As shown, the mdl\_name Column J Value should be the model’s name/Column A key value. This provides the program with the needed Model key value for transferring to the tblImport sheet from the Scenario Model region.

A screenshot of a computer

Description automatically generated

The code example shows the Worksheet\_Change event to trigger the swap based on the selector cell.

* A SetApplEnvir subroutine (not shown) toggles off screen updating, application events and automatic calculation during the swap.
* The code uses an ExcelSteps.xlam New\_mdl() (shown below) subroutine to instance the ExcelSteps.xlam mdlDest Class called from the model’s workbook.

Example event-based VBA code that swaps models based on the $N$5 selection

Private Sub Worksheet\_Change(ByVal Target As Range)

Dim rngSelector As Range, mdlDest As Object

'Set selector cell range and exit if selector cell didn't change

Set rngSelector = ThisWorkbook.Sheets("SMdl").Cells(5, 14)

If Intersect(Target, rngSelector) Is Nothing Then Exit Sub

SetApplEnvir False, False, xlCalculationManual

'Instance the mdlDest Scenario model and set mdlDest.SwapModels() ModelNew param

Set mdlDest = ExcelSteps.New\_mdl

If rngSelector = "Batch Plant" Then

ModelNew = "SMdlType1"

ElseIf rngSelector = "Other Plant" Then

ModelNew = "SMdlType2"

End If

'Swap to specified model

If Not mdlDest.SwapModels(ThisWorkbook, ModelNew:=ModelNew, ModelDefnDest:=defn\_dest) Then MsgBox "Error”

'Re-enable events and automatic calculation

SetApplEnvir True, True, xlCalculationAutomatic

End Sub

ExcelSteps.xlam sub for instancing a mdlScenario Class from the workbook containing the event-based code

Public Function New\_mdl() As mdlScenario

Set New\_mdl = New mdlScenario

End Function

Scenario Model definition strings for the top “dashboard” and swappable model, mdlDest, regions

Public Const defn\_dash As String = "SMdl:4,9:6:T:T:T:T:T:SMdlDash"

Public Const defn\_dest As String = "SMdl:10,9:0:T:T:T:F:T:SMdlDest"

' defn Booleans: IsCalc, IsSuppHeader, IsRngNames, IsMdlNmPrefix, IsLiteModel

To configure the models, these are parsed as follows, with fixed\_n\_rows=0 used to signal lack of a limit on number of rows:

Sheet\_locn:CellHome\_row:CellHome\_column:fixed\_n\_rows:<Boolean flags>:model\_name

1. See separate documentation on Scenario Models which are a configurable VBA Class/object within ExcelSteps. Scenario Models can be placed where needed on a sheet. Class methods take care of formatting, cell naming and refreshing of custom formulas. [↑](#footnote-ref-1)