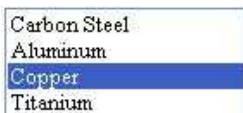


Making Custom Controls Persistent

by Jeff Henning

Mathcad Custom Controls allow you to select items from lists or fill in list boxes in order to automate worksheet operations. However, the controls aren't persistent from session to session. That is, list box states and radio button states always revert to the defaults when the file is closed and re-opened. Most of the time, you'd like to be able to save the state of these controls in a specific file as they represent a configuration of the situation being analyzed.

Select a Pipe Material

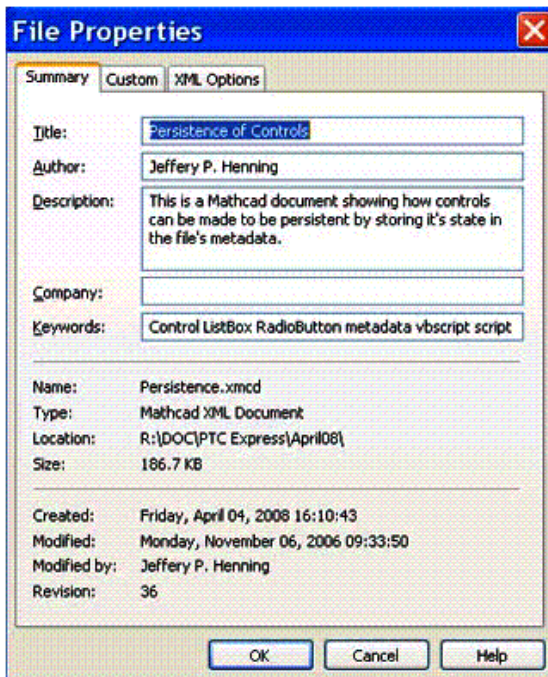


Select the Nominal Pipe Diameter:



1/2 inch Titanium pipe selected

Metadata in Mathcad 13



Using Metadata

Mathcad 13's ability to access worksheet metadata provides a way to store control states and restore them when the worksheet is opened the next time. Worksheet Metadata is nothing more than a list of named

items with values that get embedded and stored in a worksheet. Mathcad like most Windows applications has metadata that can be accessed under the **File > Properties** menu. Above is the Properties Page for this worksheet.

While any of this standard information can be accessed, the Properties Page also contains a **Custom** tab that allows you to store custom information about the worksheet. This is where you can store the states of your controls. Take a look at the list box VBScript to see how this is done. The first thing you have to do at the top of the script is initialize the list box.

```

Rem Initialize List Box
ListBox.ResetContent()

Rem Add Strings here as needed
ListBox.AddString("Carbon Steel")
ListBox.AddString("Aluminum")
ListBox.AddString("Copper")
ListBox.AddString("Titanium")

Rem Initialize Selection If desired
ListBox.CurSel = 0      'Initialize selection to 0
Set metadata = worksheet.metadata 'Interface to metadata

Set items = metadata.customitems 'collection of custom
properties
For each item in items 'Look through the list of items

    If (item.name = "ListBoxState") Then 'If we find the
state...
        ListBox.CurSel = item.value      '...set the selection
    End If      'Otherwise leave selection at 0
Next

```

Most of this is standard, except where the current selection is initialized.

1. First initialize the current selection to zero (ListBox.CurSel = 0) to initialize selection of the first item in the list. This is the default behavior.

2. Then look through all of the metadata on the Custom Properties Page to see if you have stored an item named "ListBoxState" and saved this file. If this is the first time the file has been run, say from a template, then this item won't exist, and the default selection will be active.

3. If the "ListBoxState" does exist, then you need to update the current selection to the value stored in the custom item.

That's how to initialize the list box. Now see how the metadata winds up on the Custom Properties Page in the first place. Every time a list item is selected, the Start, Exec, and Stop events occur, running the corresponding Subroutines in the VBScript. At the end of this process, during the Stop event, you'd like to update your metadata dynamically.

The coding for the Stop event (ListBoxEvent_Stop()) is shown below.

```

Sub ListBoxEvent_Stop()
    'Define a metadata object
    Set myMetadata = CreateObject ("Mathcad.CustomMetadataItem")

```

```

' Set Type, Name, and Value of the Metadata
myMetadata.Type = mcCMTNumber
'Number Type
myMetadata.Name = "ListBoxState"
'Name for saved state
myMetadata.Value = ListBox.CurSel
'CurrentSelection

' Create the Custom Item Worksheet.Metadata.AddCustomItem
myMetadata

' Note: Overwrites the Custom Item if it already exists

End Sub

```

To put information into the Custom metadata, create a new CustomMetadataItem object and populate it with the necessary information before storing it in the worksheet's metadata.

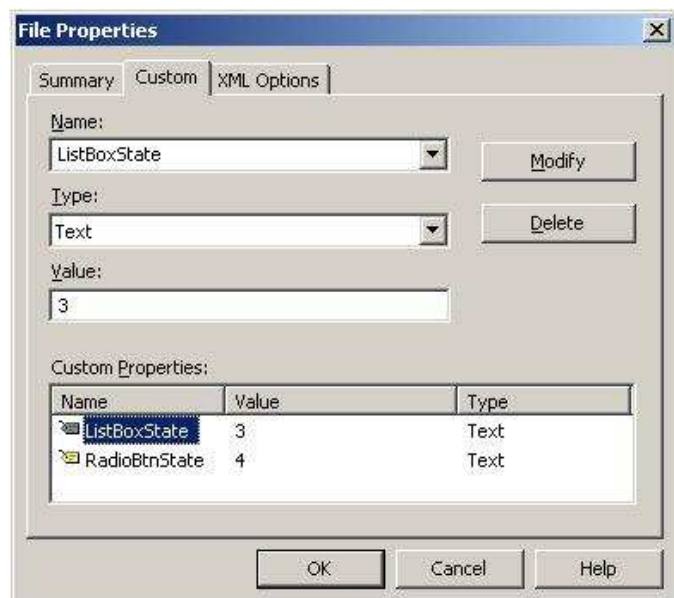
The object is created with **CreateObject("Mathcad.CustomMetadataItem")**. This object has several members that you need to set:

Name = The text string name of our Custom Item (**"ListBoxState"**)

Type = the type of data stored (mcCMTText, mcCMTDate, **mcCMTNumber**, mcCMTYesNo)

Value = the value (of **Type**) to be stored (**ListBox.CurSel**)

Once you have populated your object with the proper data, you need to add this item to the list of custom metadata. Do this with the AddCustomItem member function of the Metadata collection. Once you've made a selection, the Custom Properties page looks like the one shown below.



The state of the four radio buttons can also be made persistent. Check the Custom item "RadioBtnState," which you created, during initialization of each button.

```

Rem Initialize the Proper Button
Set metadata = worksheet.metadata 'Interface

```

```

Set items = metadata.customitems 'collection of custom
properties
For each item in items
    If item.name = "RadioBtnState") Then

        RadioBtn.Check = (item.Value =
CStr(RadioBtn.ButtonID))

    End If

Next

```

and then update the Custom item when a button is clicked thereby changing the state.

```

Sub RadioBtn_Click()

    RadioBtn.Recalculate()
    ' Define a metadata object

    Set myMetadata =
CreateObject("Mathcad.CustomMetadataItem")

    ' Set Type, Name, and Value of the Metadata

    myMetadata.Type = mcCMTNumber          'Number Type

    myMetadata.Name = "RadioBtnState"      'Our name for
saved state

    myMetadata.Value = RadioBtn.SelectedButton() 'Current
Selection

    ' Create the Custom Item

    Worksheet.Metadata.AddCustomItem myMetadata

    ' Note: Overwrites the Custom Item if it already
exists

End Sub

```

The other control types can also be made persistent in a similar manner.

Right-click, choose **Save Target As**, and change the extension to **XMCDZ** and File Type to **All** to **download Mathcad file**. (Mathcad 13)

Was this article helpful? [Let us know.](#)

[\[PRINTER FRIENDLY VERSION\]](#)

[Contact PTC](#) | [Privacy Policy](#) | [PTC Express Archive](#) | [Subscribe](#) | [Edit Profile](#)

PTC, 140 Kendrick Street, Needham, MA 02494 USA