NANCY GRACE ROMAN SPACE TELESCOPE

Use of Assistive Technology to Augment API Capabilities

International Conference on Environmental Systems
July 12-16, 2026

Paper ICES-2026-587

Hume Peabody
Kiet Vu

NASA-Goddard Space Flight Center
Aerodyne Industries

Agenda

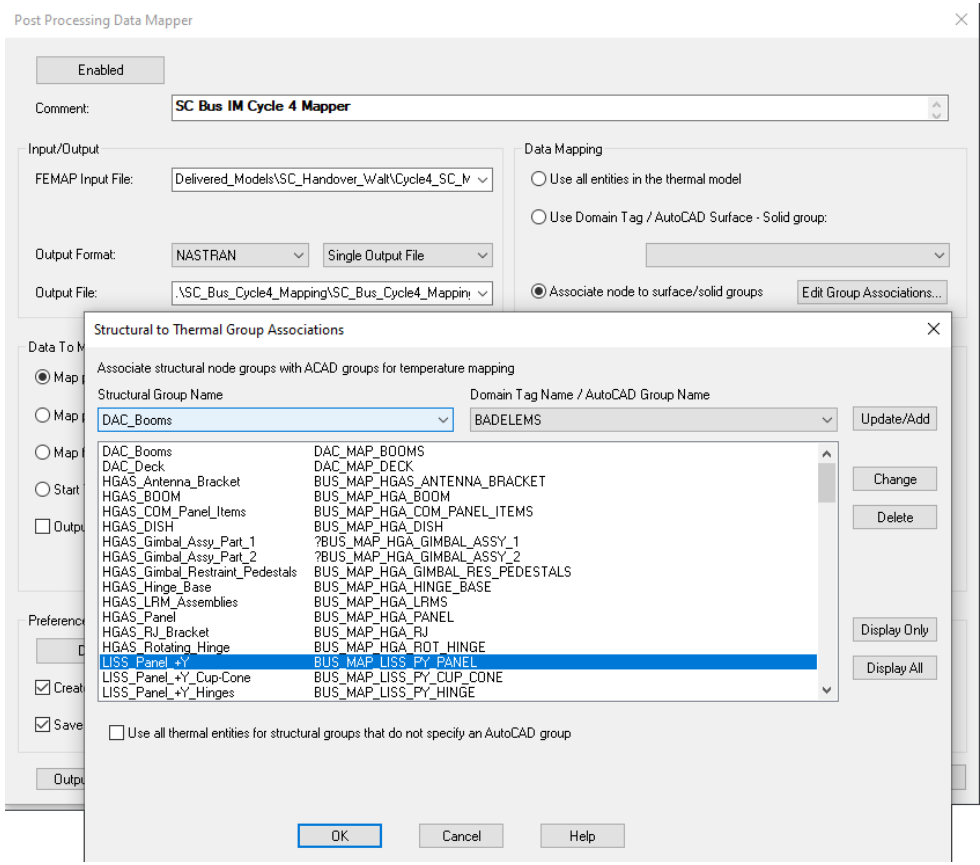
- Introduction
- Motivation
- Approach – Automation Elements
  - *Finding Elements in the GUI*
  - *Interacting with GUI Elements*
  - *Accessing Data in GUI Elements*
  - *Finding Elements more Efficiently*
- Challenges
- Conclusions

Introduction

- Application Programming Interfaces (APIs) have in recent years been introduced and expanded to allow analysts to programmatically access capabilities and data in thermal models
- But, the thermal analysis software was not developed from the start with API access in mind…
- Therefore, the capabilities and data available in the API are typically a subset of those available through the software Graphical User Interface (GUI)
- So, if a capability or data is not yet exposed to analysts through the API and the data is contained in a binary format, analysts are left with few options to access the needed data or capability
- But, there is another way to potentially access data without the API…

Motivation

- For the Roman Space Telescope project, Integrated Modeling (in particular Structural-Thermal-Optical-Performance) has been a major analysis effort throughout the program
- One of the most laborious parts of the process is verifying the accuracy of the mapping of thermal predictions to structural models for thermal distortion
- A Post-Processing Data Mapper is used in Thermal Desktop® to perform the mapping, but the verification is a tedious comparison of thermal contour plots of the thermal model compared to mapped contour plots on the FEM
- This is often done by associating selected groups of thermal objects mapped to selected groups of FEM grid points to minimize incorrect mapping at interfaces and can result in hundreds of images to manually generate and compare
- Display of these associations is accomplished using the Display Only button capability in the GUI
- Unfortunately, this capability is not currently exposed in the API…

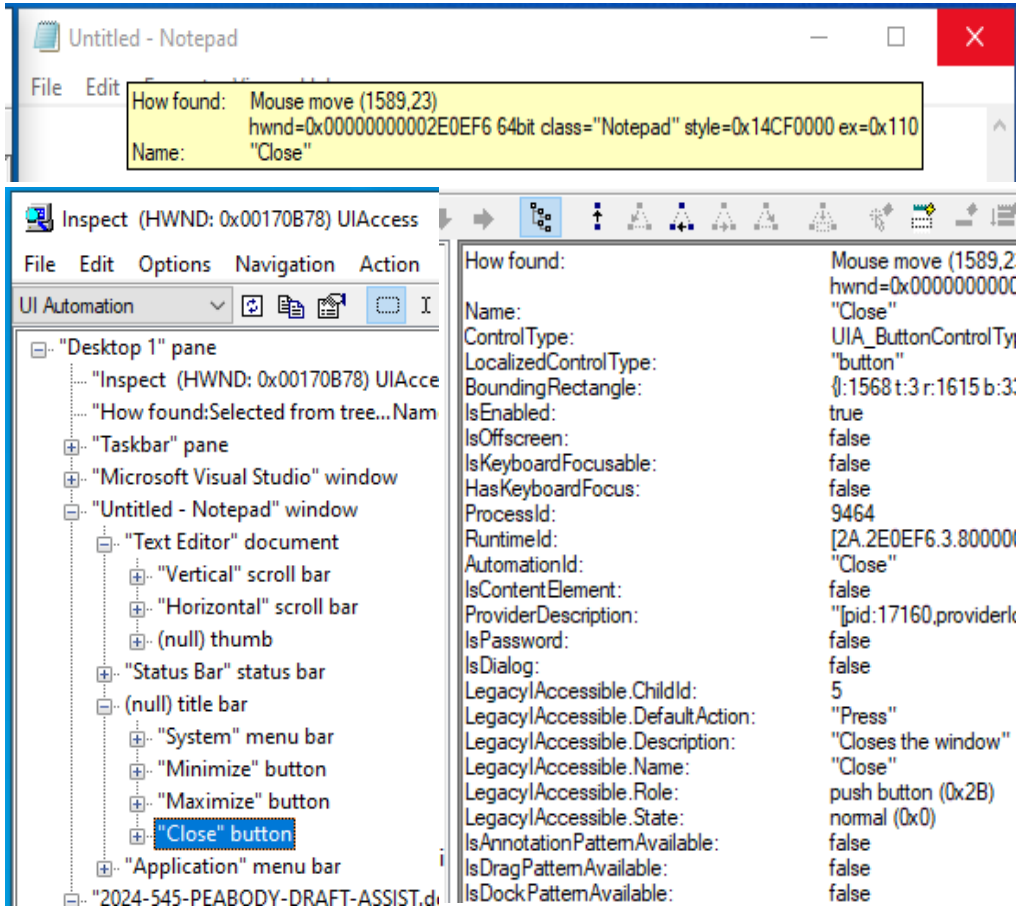


Approach – AutomationElements

- Fortunately, *Assistive Technology* allows programmers access to GUI elements in external programs and manipulate them or retrieve data
  - This is notionally used to help disabled users to access programs more easily when they were not developed with a disabled user in mind
  - For example, Assistive Technology could be used to extract text from a document and then process it to be read aloud for a blind user
- This same technology could be used to access capabilities and data in a thermal analysis program and augment what can be accomplished programmatically via the APIs
- The basis for this is called AutomationElements in the .NET framework and the process for utilizing it is as follows:
  - Navigate the Visual Elements of the Windows environment to hierarchically find the AutomationElement representing the GUI control of interest
  - Perform some action (Button Click, ListItem selection, retrieve TextBox text) using Control Patterns
- Fortunately, tools like *inspect.exe* exist that allow users to uniquely identify the controls in the GUI for programmatic manipulation

Finding Elements in the GUI

- Tools like *inspect.exe* allow a user to select GUI elements in a program and display the relevant information, such as ControlType, Name, IsEnabled, etc
- Navigation is hierarchical, beginning with the Windows Desktop (Desktop 1 Pane)
- In this instance, Windows has GUI elements for Inspect, a ToolTip on How found, the TaskBar, Visual Studio, NotePad and other applications
- Under Notepad is a TextEditor document, a Status Bar, a Title Bar, and a Menu Bar
- Further navigating the Title Bar shows a Menu Bar and Minimize, Maximize and Close Buttons
- Having hovered over the Close Button shows the Name for the control as well as the LocalizedControlType
- This Name can be used by the AutomationElements to programmatically interact with this control in the Notepad application. (Note that the Name need not be unique)



Interacting with Elements in the GUI

- This code snippet will navigate from the top Desktop window to find the Close button in an instance of Notepad and execute the Click to close the application
- Start at the top:

```
Dim AEWindow as AutomationElement = AutomationElement.RootElement.FindFirst( TreeScope.Children,
    New PropertyCondition(AutomationElement.LocalizedControlTypeProperty, "window")
) '// Find the first element in the Windows environment
```

- Use the TreeWalker object to find the Notepad instance:

```
Do While Not IsNothing(AEWindow)
    If AEWindow.Current.Name.ToUpper.Contains("NOTEPAD") Then Exit do
    AEWindow = TreeWalker.ControlViewWalker.GetNextSibling(AEWindow)
Loop '// Loop through all elements in the Windows environment until we find NOTEPAD in the name, then exit
```

- Find the Title Bar and then the Close Button:

```
Dim AETitle as AutomationElement = AEWindow.FindFirst(TreeScope.Children,
    New PropertyCondition(AutomationElement.ControlTypeProperty, ControlType.TitleBar)
) '// From the NOTEPAD window we found earlier, find the TitleBar control
Dim AEClose as AutomationElement = AETitle.FindFirst(TreeScope.Children,
    New PropertyCondition(AutomationElement.NameProperty, "Close")
) '// From the NOTEPAD TitleBar control we found earlier, find the Close Button
```

Interacting with Elements in the GUI

- Lastly, “Click” the Close to shut down the Notepad Application:

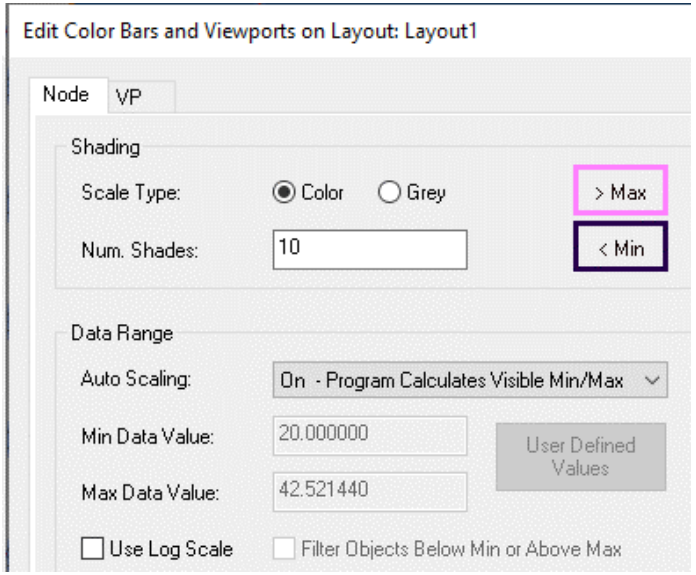
```
Dim IPClose As InvokePattern = AEClose.GetCurrentPattern(InvokePattern.Pattern)
IPClose.Invoke() '// From the Close Button of the TitleBar in NOTEPAD, execute a button click
```

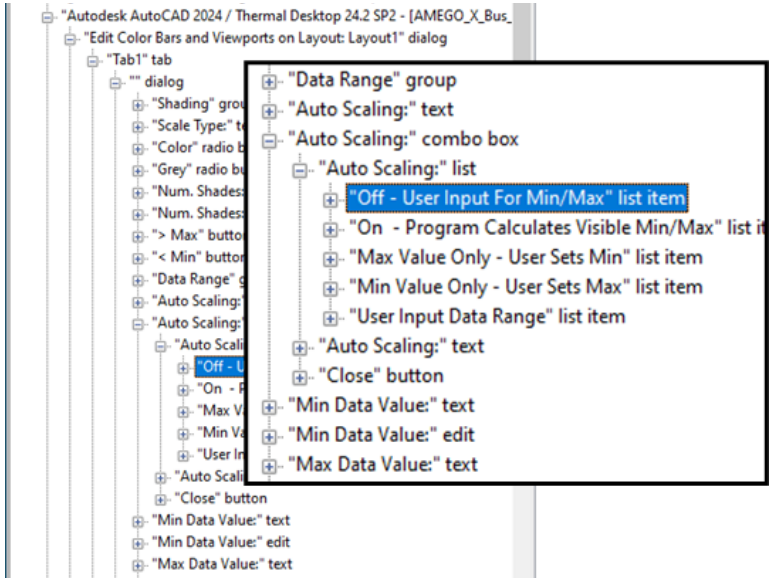
- Being able to close an application via a program may seem trivial and unnecessary. But a tool was recently developed which allowed users to output the mass for selected Domain Tag Sets in Thermal Desktop
- This tool called a method in the API which executed the command line function to calculate the mass for a given selection set and passed in the Domain Tag Set argument
- At the conclusion of that command line function, an instance of Notepad was launched with the results
- Repeating this for hundreds of Domain Tag Sets representing various constituent parts of the design would result in hundreds of Notepad instances also being opened which would be incredibly tedious to close manually
- Fortunately, AutomationElements allowed each instance to be closed without the user needing to intervene

Accessing Data in the GUI Elements

- The image on the right shows the Edit Color Bars from in the Thermal Desktop GUI along with the results from *inspect.exe*
- Following a similar approach to earlier, the Min Data Value text can be found, but note that this control is currently disabled
- To enable it, the AutoScaling listbox needs to be set to “User Input for Min/Max”
- This requires navigating the hierarchy to find and invoke the Select Pattern on the list item
  - Desktop → Autocad → ColorBarForm → ColorBarTab → ColorBarCombo → ColorBarComboList → ColorBarComboListItems

```
Dim AEColorBarComboListItems As AutomationElementCollection =
  AEColorBarComboList.FindAll(TreeScope.Children,
  New PropertyCondition(AutomationElement.LocalizedControlTypeProperty, "list item",
        PropertyConditionFlags.IgnoreCase)
) ' // Get ColorBarComboListItems collection from ColorBarComboListBox
Dim SIPColorBarComboListItems As SelectionItemPattern =
  AEColorBarComboListItems(0).GetCurrentPattern(SelectionItemPattern.Pattern
) ' // Set SelectedItem pattern for first ComboBox list item
SIPColorBarComboListItems.Select() ' // Invoke selection
```

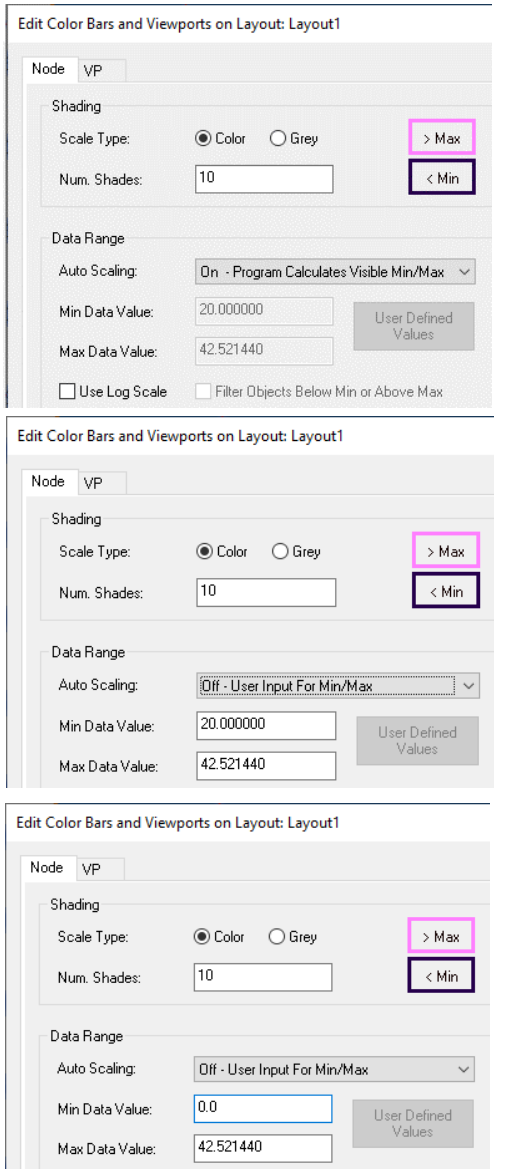




Accessing Data in the GUI Elements

- The image on the right shows the Edit Color Bars from in the Thermal Desktop GUI at three stages
  - Original
  - After Selecting Off – User Input for Min/Max
  - After Updating the Min Data Value text box
- Following a similar approach to earlier, the AutomationElement for the Min Data Value text box (AEMinValue) can be found navigating the hierarchy
- Using a TextPattern allows a programmer to access the data in the Text Box
- Similarly, using a ValuePattern allows the programmer to set the data in the TextBox
  - It is incumbent on the developer to ensure that controls are enabled and can accept data
- The following code snippet retrieves the first 1000 characters in the text box and then sets the value to 0.0

```
Dim TPMinValue As TextPattern = AEMinValue.GetCurrentPattern(TextPattern.Pattern)
Dim StrMinValue as String = TPMinValue.DocumentRange.GetText(1000) '// Get the Min Data Value
Dim VPMinValue As ValuePattern = AEMinValue.GetCurrentPattern(ValuePattern.Pattern)
VPMinValue.SetValue("0.0") ' // Set the Min Data Value
```



Finding Elements more Efficiently

- Code snippets so far have shown a methodical, step-by-step navigation of the hierarchy
- But using a recursive argument (SubTree) instead of a next level down argument (Children) allows developers to find controls much faster by navigating the tree during the FindFirst method

```
Dim AEColorBarComboListItem As AutomationElement = AEAcad.FindFirst(TreeScope.Children,
    New PropertyCondition(AutomationElement.NameProperty, "Off – User Input For Min/Max")
) ‘ // Search next level down for elements with a name of Off – User Input for Min/Max
Dim AEColorBarComboListItem As AutomationElement = AEAcad.FindFirst(TreeScope.Subtree,
    New PropertyCondition(AutomationElement.NameProperty, "Off – User Input For Min/Max")
) ‘ // Search hierarchy for elements with a name of Off – User Input for Min/Max
```

- However, this approach does assume that the *first* control with the *specified name* is the intended target. Recall, that Names not be unique…
- Returning a Collection of *all* controls that meet the specified criteria using the FindAll method would return two elements: a text box (edit) and a label (text):

```
Dim MinValues As AutomationElementCollection = AEAcad.FindAll(TreeScope.Subtree,
    New PropertyCondition(AutomationElement.NameProperty, "Min Data Value:")
) ‘ // Search hierarchy for elements with a name of Min Data Value:
```

Finding Elements more Efficiently

- The specification of the desired element could also be further refined using an AndCondition

```
Dim MinValues As AutomationElementCollection = AEAcad.FindAll(TreeScope.Subtree,
    New AndCondition(
      New PropertyCondition(AutomationElement.NameProperty, "Min Data Value:"),
      New PropertyCondition(AutomationElement.LocalizedControlTypeProperty, "edit")
    )
  ) ‘ // Search hierarchy for elements with a name of Min Data value: that are also a TextBox
```

- The code snippet above would return a collection of all Text Boxes (edit) that have a name of “Min Data Value:” from any hierarchy path starting from the AutoCAD window (which *should* just be a single element)
- That said, the *.Count* property should always be checked for a Collection to ensure the code is working as intended

Challenges

- Now for the bad news…This approach does have its share of caveats and pitfalls
  - Since this approach is working with the GUI of external programs, a user moving focus to another application or control may cause disruption in the code execution. This approach is best left for fire-and-forget type applications that can be running undisturbed for their duration
  - This approach does require robust error checking to ensure that control are enabled and able to be interacted with by the code. Furthermore, it does require very precise narrowing of the intended targets. It is quite possible to be operating on a wrong control in a second instance of an application that may also be open.
  - Developers are not in control of an external program's GUI. So, it is possible that forms may change, breaking the code. That said, *major* GUI changes in legacy programs are not all that common (developers are more often interested in adding to capabilities and not removing them…)
- That said, these limitations are manageable with careful development and use by an analyst

Conclusions

- APIs have expanded the abilities of analysts to further develop capabilities to interface with thermal analysis tools
- However, if a particular capability is available in the GUI, but not exposed in the API, another approach must be used
- AutomationElements allow users to programmatically interact with the GUI, giving analysts another tool in their toolbox for utility development
- The process consists of:
  - Navigating the GUI hierarchy to find the control of interest
  - Executing some action (Click, Retrieve Text, Set Value, Select List Item, etc) on the control of interest
- This approach was recently used on the Roman Space Telescope project to develop a utility that:
  - Took an input file of Thermal Domain Tag Sets, Structural FEM group names, and View information
  - Looped through and generated thermal model contour plots and FEM mapped contour plot images
  - And pasted the images into an output PowerPoint file
- This utility saved many hours of manual labor and turned the process into a turn-key, overnight effort