ANSYS Mechanical Scripting: HOWTO Part 2

by Matt Sutton December 15, 2010 2:04 pm Leave a comment The Focus

Where We Left Off Last Time…

In the first part of this how to guide, I left you with a general description of the architecture of ANSYS Mechanical product. If you haven't read that post, I recommend that you at least skim through it to familiarize yourself with the structure of ANSYS Mechanical. One thing we established is that customizing ANSYS Mechanical is not for the faint of heart, but at the same time, it is also not an impossible task. There are two primary difficulties associated with Mechanical customization. They are:

1. ANSYS Mechanical was never truly *designed* to be customized, at not least by end users.
2. There is no documentation of the internal API, so we're left with deciphering the inner workings ourselves.

Nonetheless, the Mechanical application is architected as a collection of functional pieces written in C++ that are tied together with Javascript via a COM interface. What that means for us is that most of the functionality of ANSYS Mechanical is exposed to us through an interface that can be scripted. So, even though Mechanical may not have been explicitly designed to be scripted nor are the internals documented, yet we find that it is possible to script, and since the ANSYS developers have followed a consistent pattern, its internal structure is comprehensible by an outside agent.

A Recipe for Discovering ANSYS Mechanical Functionality

As I hinted in the introduction to this post, the ANSYS developers have followed a sound software development practice in the construction of the ANSYS Mechanical product. The practice is this: Find a particular pattern that solves a problem you currently face in the most general case. So, lets describe the problem the ANSYS developers faced and then we will describe what I think was their solution. The problem can be summarized as this: How do you create a GUI that is easily extended to later include functionality that you don't even know about today? That is the problem from 50,000 ft. Here are some more details about the problem.

- Extensible means that I (the ANSYS Developer) may need to add / subtract menu items, toolbar buttons, menu groups, button groups at will as I add in more functionality over time. I don't want to have to rewrite lots of low level code every time I need to add a button, for example.
- I already know we (ANSYS) have customers around the world who speak different languages. I don't want to maintain 10 different versions of ANSYS Mechanical for every language that I need to support. It would be nice if I could separate the language from the rest of the code.
- If I'm going to be adding buttons, menu items, etc… at will, I need to be able to describe what happens when the user presses the button, selects the menu, etc… I don't want to paint myself into a corner, so this needs to be as general as possible.

So, that's the problem and perhaps you're already seeing the solution in your head. Here is what the folks at ANSYS came up with, which is a pretty standard pattern for constructing a flexible GUI.

1. Push the actual GUI construction to the very last minute. That is, you don't hard code a GUI, rather you hard code a GUI builder engine. The GUI builder engine reads in a human readable text file at run time that describes the GUI layout and then it dynamically creates the GUI on the fly every time the program launches. Now what you have is a static executable that can dynamically morph its appearance to anything you want.
2. This is a collorary to 1. Since we want to push GUI construction to the very last minute, we therefore have to have the means to describe *at runtime* what should happen when the user presses a button or selects a menu item in the most general terms. Hence we need a dynamic scripting language that isn't interpreted until run time. Note that the only real purpose of this language is to orchestrate at a high level what should happen. The low level details of say meshing for example ca be constructed in a compiled language and contained in libraries as long as we

call into those libraries using this scripting language. So, because of 1 we end up with a scriptable interface to Mechanical.
3. We want to isolate language differences, i.e. English, French, Germany, etc… to as small a subset of files as possible. So, the best programming technique for this problem is the old “extra level of indirection.” That is, instead of using a literal string like “File” in the code that describes the menu item “File”, we use an identifier ID_File that is just a numeric constant. Then, we construct a table that maps a string value to a numeric identifier. So, maybe ID_File has a numeric value of 438. In our table there would be an entry of: 438, “File”. Likewise, anywhere in the actual GUI that we need the string “File”, we substitute the constant ID_File in its place. Now, lets say we want a French version of ANSYS Mechanical, what do we do? We simply translate the string table entries into French and create a new string table file. At run time, we read in the French string table and voilà, we have a French version of ANSYS Mechanical.

So, that is the approach the ANSYS developers took to solve the problem of creating an easily extensible GUI. The benefit for us is that we get a scripting language to boot, and if we “run the engine backwards” we get an easy algorithm for figuring out what is going on under the hood in Mechanical. So, here is the recipe for determining functionality within ANSYS Mechanical.

Main Ingredients

1. Good Text Search Tool (I use Slickedit. An added benefit is if it can search entire directories recursively)
2. Patience

Procedure

1. Locate the desired functionality in the GUI itself.
2. Search the file dsstringtable.xml for the menu string, or toolbar button description string
3. Note the corresponding ID_*** string ID within the string table file.
4. Search for the ID_*** in dscontextmenu.xml or similar GUI structure file.
5. When you find the ID_***, note the do*** command listed in the section associated with that GUI element’s action callback.
6. Recursively search the aisol, or DesignSpace directory for the corresponding do*** command.

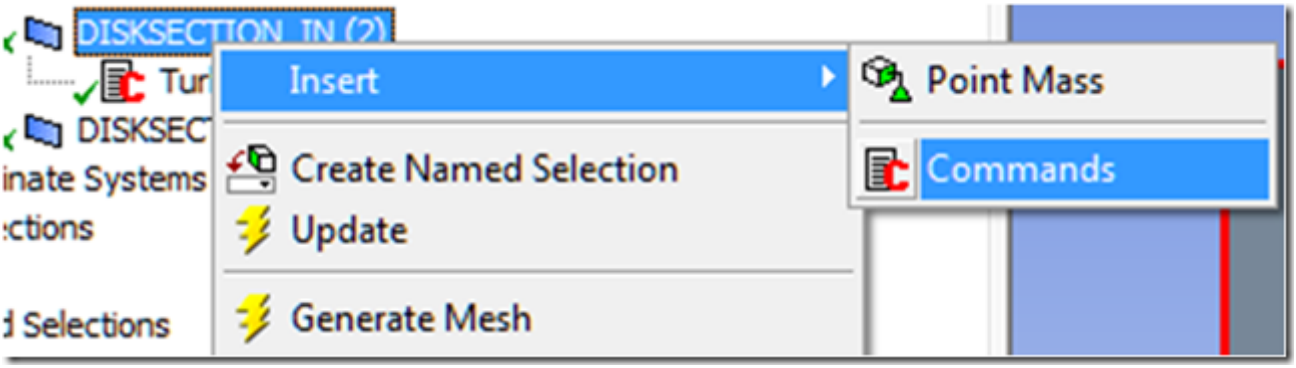
Could it possibly be simpler?

Example Use of the Above Recipe

I want to add a command object to a particular body in the tree, and eventually I want to do it programmatically.

Step 1: Locate the Functionality in the GUI Itself

Below is a screen shot of the context menu hierarchy one navigates through to insert a command object on a body in the tree. So, we see that the string we are looking for is “Commands”



Step 2: Search dsstringtable.xml for the Menu String: “Commands”

Note that the string tables are located underneath a language directory structure. If you haven’t currently guessed based on the content of this blog post, we’re using the string table located under Language\en-us\xml. When we perform the search we find the following:

```
<string id="ID_CollapseToModel">Collapse To Model</string>
<string id="ID_Collinear">Collinear</string>
<string id="ID_CommandEditorDefaultName">Commands</string>
<string id="ID_CommandEditorFileFilter">Commands Files (*.txt;
<string id="ID_CommandEditorImportError1">Active tree item is
<string id="ID_CommandEditorImportError2">Cannot insert commar
```

Step 3: Note the Corresponding ID_*** in the String Table

You can see from the image above that the ID associated with the string Commands is ID_CommandEditorDefaultName. Sounds plausible.

Step 4: Search for ID_CommandEditorDefaultName in dscontextmenu.xml

Note, that the structure of the ANSYS Mechanical GUI is described in a series of xml files as mentioned in the sections above. The file dscontextmenu.xml is one such file that shockingly describes the layout of the various context menus within the program. So, lets search for the string ID_CommandEditorDefaultName in that file and see what we find. Here are the results of that search:

```
<entry stringid="ID_CommandEditorDefaultName" icon="command">
  <visibilityFilter name="idAlbionSimAddin"/>
  <actionCallback objectId="1" methodName="doPrototypeInsertCommandEditor"/>
  <actionCallback objectId="1" methodName="FireFinished"/>
  <visibilityCallback objectId="17" methodName="CanInsertPrototypeCommandEditor"/>
</entry>
```

Note the structure of this file. There are various entries that represent individual menu items within a given context menu. For each entry there are a series of action callbacks that have an associated methodName attribute. By deduction we can assume that when the user clicks on this menu item, the listed javascript functions are called to perform the menu item's action. You'll note too that there is a visibilityCallback as well. Those wacky ANSYS developers thought of everything…

Now, one of the action callbacks is called "doPrototypeInsertCommandEditor". If that doesn't sound like a function that will insert a command editor on a prototype, I don't know what does. Let's see if we can find it.

Step 5: Search for doPrototypeInsertCommandEditor in the aisol or Designspace Directories

Now we search inside the javascript (.js) files for a function called doPrototypeInsertCommandEditor() and see what we can find. Inside the file DSMenuScript.js we find this function definition:

```
function doPrototypeInsertCommandEditor()
{
    var ProtoObj = ds.Tree.FirstActiveObject;
    if( !ProtoObj )
    {
        // couldn't find object
        WBScript.Out( localString("ID_CannotFindObject") , true);
        return;
    }

    if(ProtoObj.Class != id_Prototype)
        return;

    var cmdEditor = ProtoObj.AddCommandEditor( );
    if( cmdEditor )
    {
        var PrototypeNode = GetNode(ProtoObj.ID);
        addNodeAndChildren(cmdEditor, PrototypeNode);
        changeActiveObject( cmdEditor.ID );
    }

    sm.Clear();
    sbUpdateSelectionPaneString();
}
```

Well, what do you know? There is the 21 line function that gets called whenever you insert a command object onto a body in tree. In the intervening time between this post and the next, spend some time staring at this code. There are some fundamental aspects of the architecture of ANSYS Mechanical that appear even in this short function. Now, I know your probably thinking to yourself, this is all fine and dandy, but what if I don't just want to insert a command object. What if I want the command object to have some APDL in it? Furthermore, what do all those other variables, etc… really mean. Lastly, how do I even call this blasted function? Stay tuned… there is much, much more to come.

Related

Overcoming Convergence Difficulties in ANSYS Workbench Mechanical, Part II: Quick Usage of Mechanical APDL to Plot Distorted Elements
October 18, 2012
In "The Focus"

Advanced Capabilities to Consider when Simulating Blow Molding in Ansys Polyflow or Discovery AIM
May 21, 2020
In "The Focus"

6 – An update on outputting results in Ansys Mechanical: 3D Printing Results
January 6, 2021
In "The Focus"

You must **log in** to post a comment.

‹ **Finding the Location of a Value in a Common Block – Again**

Happy Holidays From PADT to the ANSYS Community ›

Contact PADT
1-800-293-PADT
info@padtinc.com

SEARCH

Search

LINKS

PADT Website
Contact PADT
PADTMarket.com
Manage PADT Subscriptions

SUBSCRIBE TO OUR BLOG

Enter your email address to subscribe to this blog and receive notifications of new posts by email.

Join 1,804 other subscribers

Email Address

Subscribe

UPCOMMING EVENTS

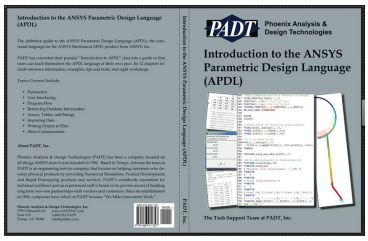
01/19/2022 - 01/21/2022
Arizona Photonics Days

04/04/2022 - 04/07/2022
37th Space Symposium
Arizona Space Industry Booth

04/12/2022 - 04/14/2022
D&M West | MD&M West

3D PRINTING GLOSSARY

INTRODUCTION TO THE ANSYS PARAMETRIC DESIGN LANGUAGE



Learn APDL with this workshop based book written by PADT's Technical Support Team. The new and improved Second Edition contains additional chapters on APDL Math and APDL in ANSYS Mechanical.

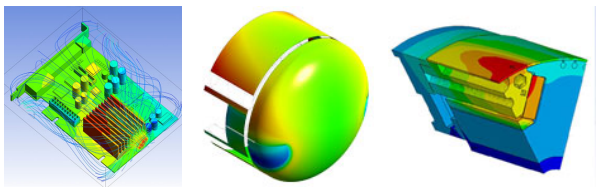
Now available on Kindle as well as paperback.

More...

SIMULATION SERVICES

PADT's simulation engineers are true experts in virtual prototyping. Trust the people you come to for ANSYS expertise to handle your simulation outsourcing needs.

Structural, Thermal, Fluid, Electromagnetic, and Systems.



LOOKING FOR ANSYS TRAINING?

Let the people who write "The Focus" train you and your team. Check out our **schedule** or **contact us** to set up a class at your place or something custom.

PODCAST: ALL THINGS ANSYS

All Things ANSYS Podcast
Presented by PADT

ANSYS ACADEMIC PROGRAM

Ansys ACADEMIC PROGRAM
Learn about software for students and research

RSS SUBSCRIBE:

RECENT POSTS:

- Surprise – 2021 Turned out to Be a Lot Like 2020
- All Things Ansys 102: Electronics Reliability Updates in Ansys 2021 R2
- Simulating an Electro-Permanent Magnet (EPM) using Ansys Maxwell
- Ansys Pro – Premium – Enterprise Electronics Licensing Adjustments
- All Things Ansys 101: Additive & Structural Optimization Updates in Ansys 2021 R2

CATEGORIES

- Additive Manufacturing
- Ansys
- ANSYS Discovery
- ANSYS Energy Innovation Campaign
- ANSYS R 18
- Carbon
- Education
- Events
- Flownex
- Fun
- Getting To Know PADT
- News

Nimbix

PADT Medical

PADT Startup Spotlight

Podcast

Product Development

Publications

Startups

Stratasys

Stratasys Marketing

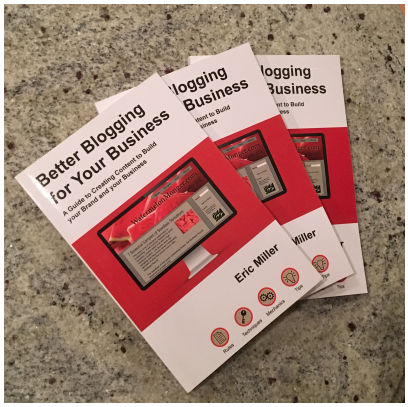
The Focus

Uncategorized

Webinar

Contact PADT
1-800-293-PADT
info@padtinc.com

THINKING ABOUT A BLOG FOR YOUR COMPANY?



Where do you start? How do you keep it going? Where do I get ideas for posts? Should I use humor?

These questions and many others are answered in the book that was inspired by the success of PADT's blog:

Better Blogging for your Business

Available now as a Kindle book or softcover on Amazon.

PADT EMAIL

Manage how PADT Emails You

Want to receive Emails from PADT? Please select any of the basic topics below to tell us what you are interested in. We promise to keep it simple sharing news, opportunities, and useful information. You can come back and change your settings whenever you need to.

* **Email**

State/Province

Company

* **Email Lists**

- [ ] PADT Additive & Advanced Manufacturing Email List
- [ ] PADT General Information Email List
- [ ] PADT Product Development and Testing Email List
- [ ] PADT Simulation Email List

By submitting this form, you are consenting to receive marketing emails from: Phoenix Analysis and Design Technologies, 7755 S. Research Dr., Suite 110, Tempe, AZ, 85284, US, http://www.padtinc.com. You can revoke your consent to receive emails at any time by using the SafeUnsubscribe® link, found at the bottom of every email. **Emails are serviced by Constant Contact.**

Sign Up!

Contact PADT
1-800-293-PADT
info@padtinc.com

SEARCH

Search

LINKS

PADT Website
Contact PADT
PADTMarket.com
Manage PADT Subscriptions

ANSYS STARTUP PROGRAM

LOOKING FOR ANSYS TRAINING?

Let the people who write "The Focus" train you and your team. Check out our **schedule** or **contact us** to set up a class at your place or something custom.

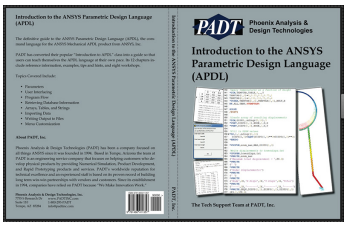
TRYING TO FIND A COMPUTER BUILT FOR SIMULATION?

Workstations, Servers, and Clusters designed by

PADT specifically for simulation users.

Learn More

INTRODUCTION TO THE ANSYS PARAMETRIC DESIGN LANGUAGE



Learn APDL with this workshop based book written by PADT's Technical Support Team. The new and improved Second Edition contains additional chapters on APDL Math and APDL in ANSYS Mechanical.

Now available on Kindle as well as paperback

More...

RECENT POSTS

Surprise – 2021 Turned out to Be a Lot Like 2020

All Things Ansys 102: Electronics Reliability Updates in Ansys 2021 R2

Simulating an Electro-Permanent Magnet (EPM) using Ansys Maxwell

Ansys Pro – Premium – Enterprise Electronics Licensing Adjustments

All Things Ansys 101: Additive & Structural Optimization Updates in Ansys 2021 R2

CATEGORIES

Additive Manufacturing

Ansys

ANSYS Discovery

ANSYS Energy Innovation Campaign

ANSYS R 18

Carbon

Education

Events

Flownex

Fun

Getting To Know PADT

News

Nimbix

PADT Medical

PADT Startup Spotlight

Podcast

Product Development

Publications

Startups

Stratasys

Stratasys Marketing

The Focus

Uncategorized

Webinar


Connect with PADT