Don’t Wait for LabVIEW R&D…
Implement Your Own LabVIEW Features!
Augmenting the LabVIEW Editor with G Code

Swathi Madhavan
Field Marketing Engineer, National Instruments
Outline

- VI Scripting
- Editor features with scripting hooks
  - Quick Drop
  - Right-click Plugins
  - Pull-down Menu Plugins
  - VI Analyzer Tests
  - Miscellaneous
- Rusty Nails
VI Scripting

If you’re going to implement your own LabVIEW features, you will undoubtedly use VI Scripting.
What is VI Scripting?

- VI Scripting is just VI Server with more stuff:
  - **VI Server**
    - Properties and methods for:
      - Controls/Indicators
      - VIs
      - Projects/Project Items
  - **VI Scripting**
    - Properties and methods for:
      - Controls/Indicators++
      - VIs++
      - Projects/Project Items++
      - Block diagram objects

Hidden Gem: vi.lib\Utility\traverseref.llb\TRef Find Object By Label.vi

www.ni.com/hiddengems
Enabling VI Scripting

**Tools > Options > VI Server > VI Scripting**

This gives you:

- New objects available in Quick Drop and palettes
- Terminal information in the Context Help window
Enabling VI Scripting

New entries in the VI Server object hierarchy

**Regular hierarchy**
- GObject
  - Panel
  - Pane
  - Splitter
  - Control
    - Numeric
    - Boolean
    - String
    - Tree
    - Listbox
    - ...
  - Decoration
    - Text
    - ...

**Scripting hierarchy**
- GObject
  - <all the stuff on the left>
  - Diagram
  - Wire
  - Terminal
  - Tunnel
  - Constant
  - Node
    - Function
    - SubVI
    - Structure
      - Loop
        - For Loop
        - While Loop
        - ...
        - ...
    - ...
    - ...
    - ...
Learning VI Scripting

- **Examples**
  - `[LabVIEW 20xx]\examples\Application Control\VI Scripting`
  - Each `.lvproj` in that folder lists the top-level examples

- **Help**
  - [VI Scripting Tutorial](#)

- **Browse Feature Code**
  - Quick Drop Plugins - `resource\dialog\QuickDrop\plugins`
  - Right-click Plugins - `resource\plugins\PopupMenus\edit time panel and diagram`
  - VI Analyzer tests - `project\_VI Analyzer\_tests`
Scripting Demo
Features with Hooks

Several editor features in LabVIEW were designed specifically so you could hook into them with G code.
Quick Drop Plugins

- Ctrl-Key Shortcuts pressed while Quick Drop (Ctrl-Space) is visible
  - Remove and Rewire, Create all Controls, Insert, Replace, etc.
- Useful for frequent, repetitive editor actions
- Documentation exists for writing your own
  - Getting Started with Custom Quick Drop Keyboard Shortcuts
- 8 plugins ship with LabVIEW 2018 and later
  - Including new ‘Reset to Origin’ (Ctrl-O) plugin in 2018
- Over 50 community plugins available for download online
- Join the Quick Drop Enthusiasts group at www.ni.com/quickdrop
Quick Drop Plugin Sample Code – Wire All Terminals

What is a ‘Hidden’ Terminal?

Among other things, it can be used to connect multiple parts of an object together, like all the ears on a Timed Loop.
Right-click Plugins

- G-based right-click menu entries
  - Front panel edit-time
  - Block diagram edit-time
  - Block diagram run-time (much less common)
- Useful for context-sensitive editor actions
- Documentation exists for writing your own
  - LabVIEW Help – [Understanding Shortcut Menu Plug-Ins](#)
- 16 plugins ship with LabVIEW 2018 and later
  - Including several new plugins in 2018
- Over 60 community plugins available for download online
- Join the Shortcut Menu Plug-ins group at [www.ni.com/lvmenus](http://www.ni.com/lvmenus)
  - Includes a link to a comprehensive presentation on creating your own
What is this doing here?

The **Request Deallocation** function is necessary whenever reading the **Data Type** property in right-click menu scripting code to avoid instabilities related to reading LV Class data across application instances.
Pull-down Menu Plugins

- G-based pull-down menu entries
  - **Tools** menu
  - **Help** menu
  - **File** menu
- Documentation exists for writing your own pull-down menu entries
  - [Integrating into the LabVIEW Menus](#)
- Useful for **environment-based** editor actions
  - Usually launching a tool of some kind
- Useful for **document-based** editor actions
  - Do something regarding the current open document (project, VI, etc.)
Pull-down Menu Sample Code – Explore
Launch Windows Explorer (or LLB Manager) opened to the folder/LLB containing this VI

What is this VI?
vi.lib/VIServer/
Menu Launch VI Information.vi

Why is this ‘2’ here?
Open template VIs for editing

Why is the FP.Close necessary?
Pull-down menu plugins always show their panel when launched. You need to close the panel when the plugin is done.
VI Analyzer Tests

- Inspect and/or Modify various aspects of VIs
  - VI Properties, front panel objects, block diagram objects
- Useful for bulk operations on VIs
- Documentation exists for writing your own
  - Designing a Custom VI Analyzer Test
- 92 tests ship with VI Analyzer Toolkit 2018
  - VI Analyzer Toolkit is licensed with LV Professional/Dev Suite or available as a separate purchase
  - VI Analyzer Toolkit is required to create new tests
- Over 50 community tests available for download online
- Join the VI Analyzer Enthusiasts group at www.ni.com/vianalyzer
- Comprehensive introductory presentation to VI Analyzer available here
Be careful when saving VIs in a bulk processing tool like the VI Analyzer. Ensure your scripting code is doing what it’s supposed to, and that your code is backed up before running the analysis.
Summary

- **Quick Drop Plugins** – *Repetitive* selection-based actions performed dozens of times per day
- **Right-click Menus** – *Context-sensitive* object-based actions usually performed less often
  - Exception – repetitive actions that are more granular than the selection boundary
- **Pull-down Menus**
  - *Environment-based* actions
  - *Document-based* actions
- **VI Analyzer Tests** – *Bulk* actions on multiple files
  - Code inspection
  - Code modification
Miscellaneous Hooks

- Icon Editor
- Edit > Create SubVI
- Bookmark Manager
- Project Templates/Sample Projects
- Custom Channel Wire Types
Customizing File > New content

[LabVIEW 20xx]\templates

- Add a folder containing your .vit files
  - Add a text file with the same name as your folder to define the UI name
- Add a diagram image file alongside each .vit named *d.png
  - Scale the image to fit within about 250x250 pixels if possible
- Description and image show up in dialog
Customizing New Class VIs

[LabVIEW 20xx]\resource\Framework\Providers\LVClassLibrary

- Add custom scripting after new accessors created
  - Search for any VI named Custom User Scripting for *.vi

- Change template new member VIs
  - Modify appropriate [Read or Write]Template*.vit
  - Conpane pattern, error case structure, location of bundle/unbundle, etc.

- Change default names of new member VIs
  - Modify strings in CLSUIP_LocalizedStrings.vi
  - Useful if you prefer Get/Set nomenclature to Read/Write

- Miscellaneous other changes
  - Dig around in the LVClassLibrary folder and see what you can find
A quick note about Custom Probes

- Help topic – Creating Custom Probes
- Lots of great custom probe libraries already available
  - The ones that ship with LabVIEW
  - History Probes
  - Statistics Probes
  - LabVIEW Class-specific probes
- Data Agnostic Smart Probes ([nugget post](nugget post))
  - LabVIEW 2016 and later
  - Probes that are not tied to a specific data type
  - Useful for gathering timing information
  - Useful for manipulating VI properties during execution
  - Useful for ???
Rusty Nails

Do not enter.
Stay away. Stay far away.
There will be no google search terms.
There will be no links.
You have been warned.
AVOID lv_* “callback” VIs

- Located in [LabVIEW 20xx]\resource folder
- Define new behavior for various editor actions
  - Ctrl-N for new VI
  - Help > About dialog
  - Ctrl-Q to exit LabVIEW
  - …and many more

- Issues
  - Framework existed before LabVIEW 8.0 (before projects)
  - Framework was not modified to handle projects
  - Most “callbacks” are not project-aware
AVOID Project Providers

- Plug in to the following with G code:
  - Pull-down menus
  - Project right-click menus
  - Project toolbar
  - New project item types
  - New icon overlays in project window

- Issues
  - Documentation
  - Stability
  - Corner Cases
  - Weird editor tricks to workaround hangs/crashes
  - Mostly untested outside NI-specific scenarios
AVOID XNodes

- Create New Diagram objects
- Define object behavior that is not possible with regular subVIs
  - Growability
  - Dynamic creation/removal of terminals
  - Custom response to type propagation
  - …and many more

- Issues
  - Documentation
  - Corner Cases
  - Somewhat untested outside NI-specific scenarios
  - Features hidden behind licensing
    - Hacks required to get around this restriction
Thanks for attending!

Any questions, please use the chat pod.