



ENGINEER
NEXT
MAY 22-25, 2017 | AUSTIN, TEXAS

Personalizing NI TestStand with Custom Step Types

Grant Gothing

Bloomy

ATE Product Manager

grant.gothing@bloomy.com

Bloomy Quick Facts

- NI Platinum Alliance Partner
 - Awarded “Outstanding Technical Resources” 2013 and 2014
 - NI Certifications
 - 13 NI Certified LabVIEW Architects
 - 2 NI Certified LabVIEW Embedded Systems Developers
 - 2 NI Certified TestStand Architects
 - 3 NI Certified Training Centers
 - 13 NI Certified Professional Instructors
 - Published “The LabVIEW Style Book” © 2007, Prentice Hall
- ISO 9001:2008 Certified



About Grant Gothing

- 10 years at Bloomy
- ATE Product Manager
- CLA, CTA, CPI
- EFT Module for TestStand
 - Product manager and chief architect



Quick Survey

- Heard of TestStand Custom Step Types
- Developed a sequence using Custom Step Types
- Created or modified a Custom Step Type
- Deployed Custom Step Types to multiple developers/systems



Agenda

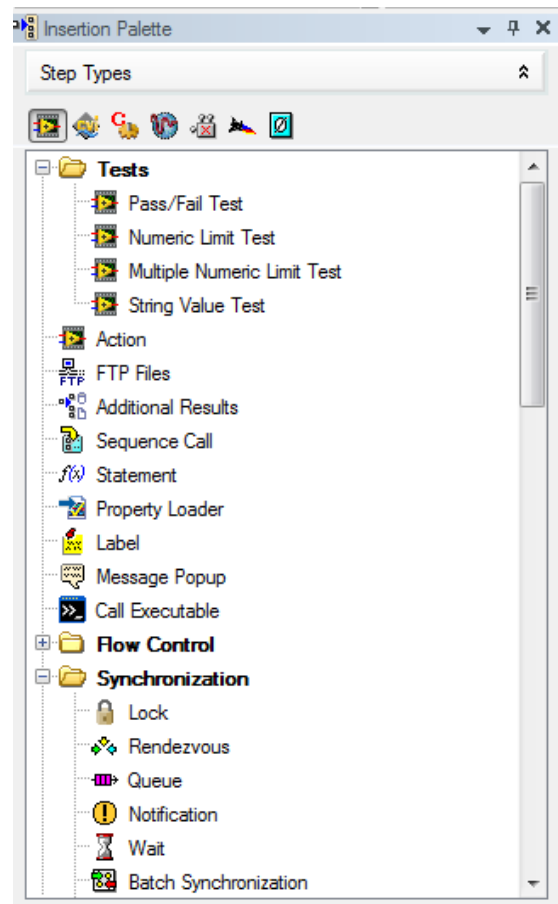
- Quick intro to TestStand Custom Step Types
- Why make a Custom Step Type?
- When should you create a Custom Step Type?
- Common applications of Custom Step Types
- How to make the most of your Step Type

Intro to Custom Step Types

Built-in Step Types

Generic interfaces

- Ship with TestStand
- Code modules & subsequences
- Statements, message popups, property loader
- Labels & comments
- Flow control & synchronization

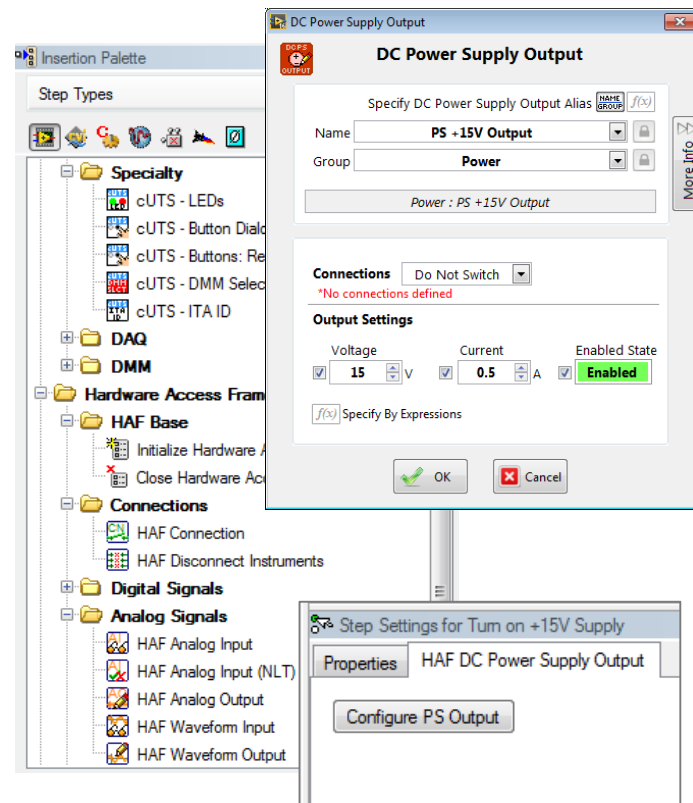


Intro to Custom Step Types

Custom Step Types

Application/architecture-specific interfaces

- User-Developed
- Wizard-like configuration dialogs
- May call code modules
- Hardware & file interfaces
- Custom data & reporting



Anatomy of a Custom Step Type

Step Data (variables)

- Configuration
- Results
- Custom data types

Step Properties

- Name, Description, Icon
- Run Options, Post-actions
- Default Expressions
- Default adapter and code module
- Disable Properties

Sub-Steps

- Edit step code modules
- Pre-Step code modules
- Post-Step code modules

Module

- Adapter
- Code Modules

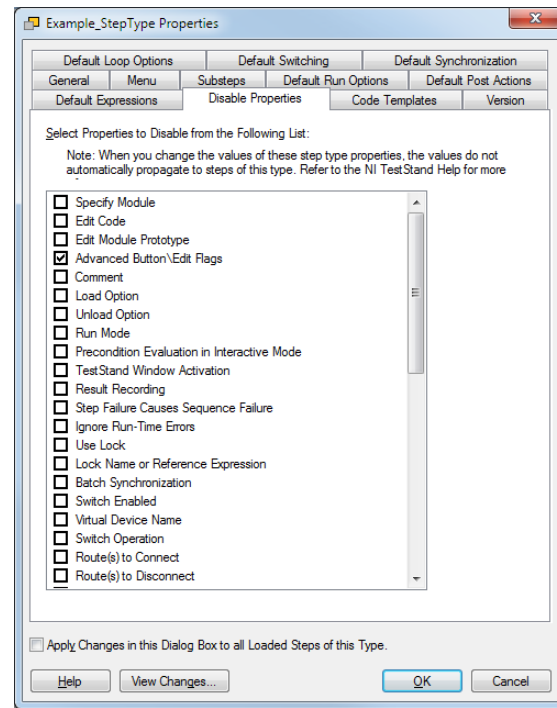
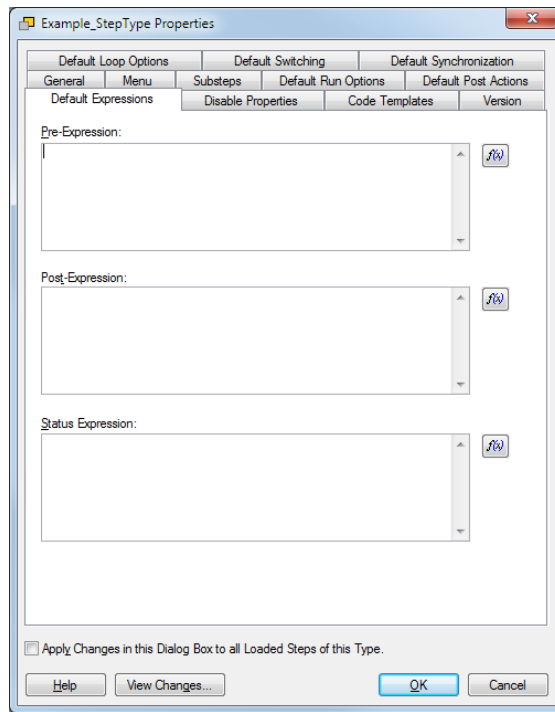
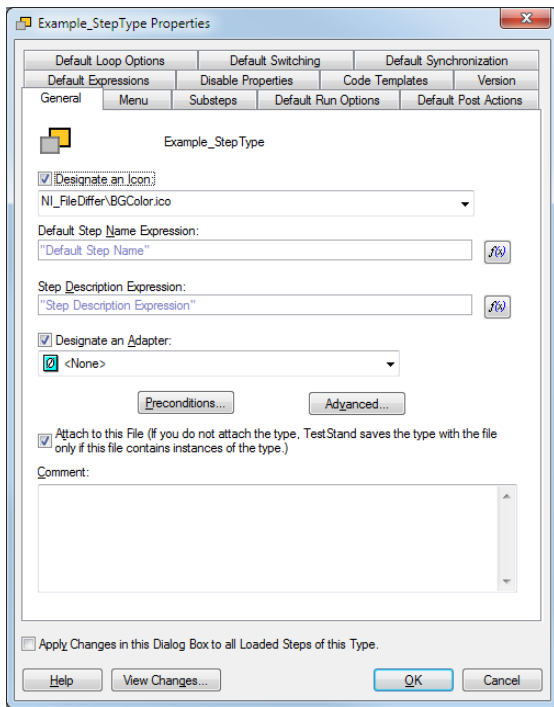
Anatomy of a Custom Step Type

Step Data (variables)

[-] Example_Step Type		Step Type, Type ...	0.0.0.0 (Modified)	MyTypes.ini
[-] Result		Container		
+ Error		Error (Container)		
ABC Status	""	String		
ABC Report Text	""	String		
+ Common		CommonResults (...)		
<Right click to insert Field>				
[-] Configuration		Container		
123 Parameter1	0	Number		
<Right click to insert Field>				
<Right click to insert Field>				

Anatomy of a Custom Step Type

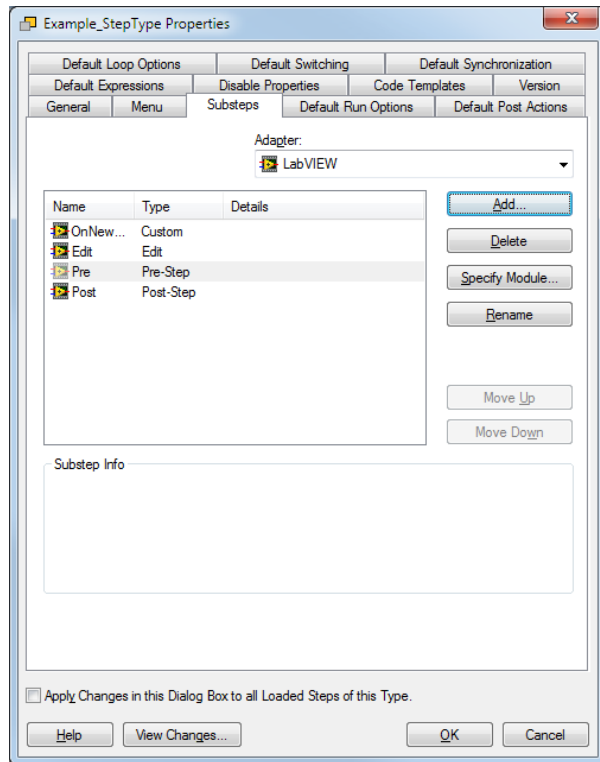
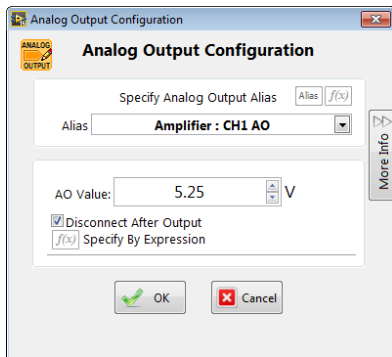
Step Properties



Anatomy of a Custom Step Type

Sub Steps & Code Module

Edit Substep(s)



Preconditions

Pre-Expression

Pre-Step substep(s)

Main module

Post-Step substep(s)

Post-Expression

Status expression

Post Actions

Common custom step types

(Or built-in steps implemented as custom step types)

- Pre-TS 2016 Property Loader
- IVI Functions
- NI TCL Steps
- ExtraPutty

The screenshot displays the NI TestStand software interface. The top window, titled 'Steps: MainSequence', shows a sequence of steps:

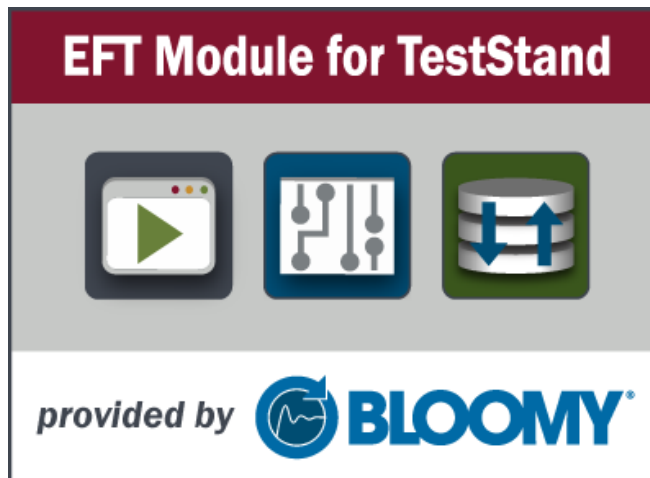
- Main (2)**
 - Initialize Connection: Initialize connection with "" target, Login...
 - Send-Received data**: Send command : "" on target : 0 with Ti...
 - <End Group>
 - Cleanup (0)**: <Insert Steps Here>

The bottom window, titled 'Step Settings for Send-Received data', shows the configuration for the selected step:

Module: ExtraPuTTY.ExtraPuTTY.dll
Function: SendRcvData_F

Parameter Name	Description	Log	Value Expression		
Return Value	void	<input type="checkbox"/>			
ConnectionId	unsigned long	<input type="checkbox"/>	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Command	const char *	<input type="checkbox"/>	""	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Title	const char *	<input type="checkbox"/>	""	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Comments	const char *	<input type="checkbox"/>	""	<input type="checkbox"/>	<input checked="" type="checkbox"/>
TimeCapture	long	<input type="checkbox"/>	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>
DataRcv	char [1000000]	<input type="checkbox"/>	Step.Result.DataReceived	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MaxSizeofData	long	<input type="checkbox"/>	1000000	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Settings	unsigned long	<input type="checkbox"/>	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>
arg9	bool &	<input type="checkbox"/>	Step.Result.PassFail	<input type="checkbox"/>	<input checked="" type="checkbox"/>
arg10	char [1024]	<input type="checkbox"/>	Step.Result.ReportText	<input type="checkbox"/>	<input checked="" type="checkbox"/>
arg11	bool &	<input type="checkbox"/>	Step.Result.Error.Occurred	<input type="checkbox"/>	<input checked="" type="checkbox"/>
arg12	long &	<input type="checkbox"/>	Step.Result.Error.Code	<input type="checkbox"/>	<input checked="" type="checkbox"/>
arg13	char [1024]	<input type="checkbox"/>	Step.Result.Error.Msg	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Bloomy EFT Module for TestStand



INNOVATIONS

SHOP

SUPPORT

COMMUNITY

[Home](#) > [Documentation](#) > [Best Practices for Custom Step Type Development](#)

Best Practices for Custom Step Type Development

Publish Date: Nov 21, 2014 | 8 Ratings | 5.00 out of 5 |  [Print](#)

Overview

TestStand includes numerous built-in step types to create steps in a test sequence. Some step types call code modules, specify how the

Best Practices for Custom Step Type Development

<http://www.ni.com/product-documentation/8300/en/>

Joe Spinozzi – Cyth Systems LLC

Why make a Custom Step Type?

- Simplify
- Standardize
- Speed Up



Sequence
Development

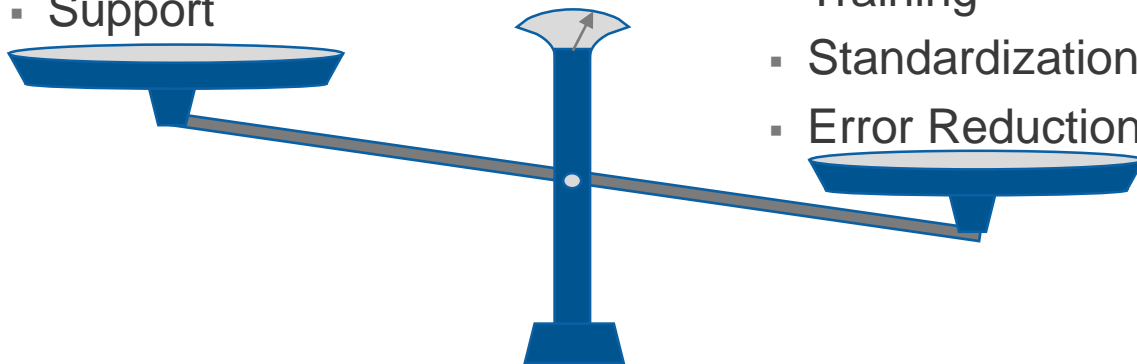
When should you make a Custom Step Type?

Step Type

- Development
- Debug
- Deployment
- Maintenance
- Support

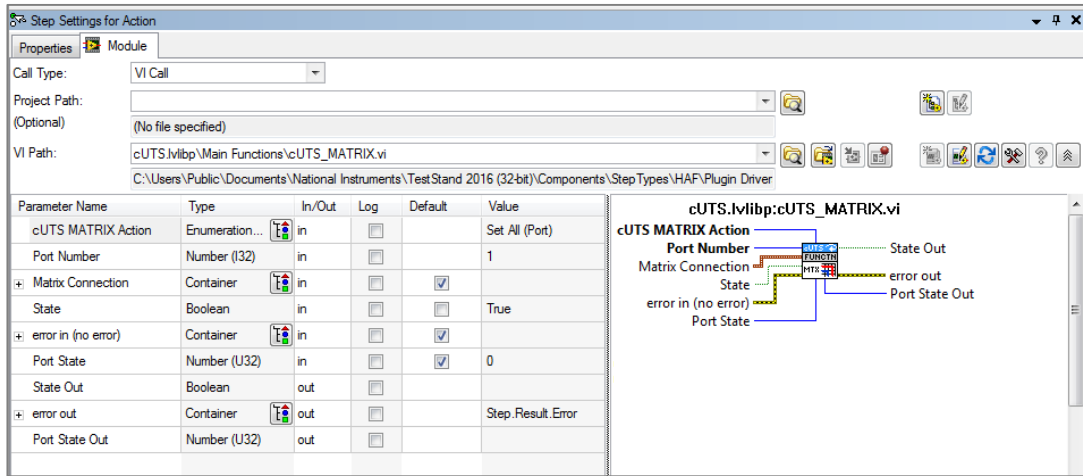
Time Savings

- Test Development
- Test Deployment
- Training
- Standardization & Reuse
- Error Reduction



Common applications of Custom Step Types

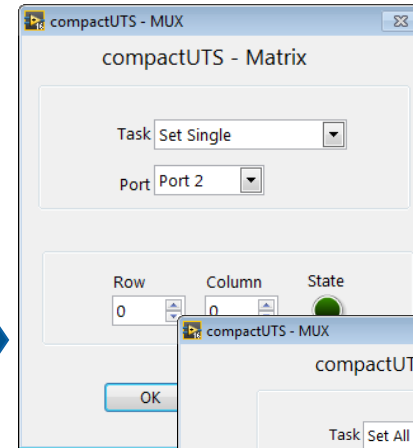
Action engine front-end



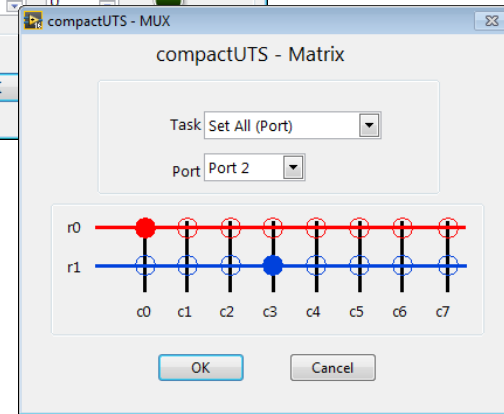
The screenshot shows the 'Step Settings for Action' dialog box. It has a 'Properties' tab and a 'Module' tab. Under 'Properties', 'Call Type' is set to 'VI Call', 'Project Path' is '(No file specified)', and 'VI Path' is 'c:\Users\Public\Documents\National Instruments\TestStand 2016 (32-bit)\Components\Step Types\HAF\Plugin Driver'. The 'Module' tab shows a table of parameters for the 'cUTS MATRIX Action'.

Parameter Name	Type	In/Out	Log	Default	Value
cUTS MATRIX Action	Enumeration...	in	<input type="checkbox"/>		Set All (Port)
Port Number	Number (I32)	in	<input type="checkbox"/>		1
Matrix Connection	Container	in	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
State	Boolean	in	<input type="checkbox"/>	<input type="checkbox"/>	True
error in (no error)	Container	in	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Port State	Number (U32)	in	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0
State Out	Boolean	out	<input type="checkbox"/>		
error out	Container	out	<input type="checkbox"/>		Step.Result.Error
Port State Out	Number (U32)	out	<input type="checkbox"/>		

Below the table is a block diagram for 'cUTS.lvlib:cUTS_MATRIX.vi' showing inputs like 'Port Number', 'Matrix Connection', 'State', 'error in (no error)', and 'Port State', and outputs like 'State Out', 'error out', and 'Port State Out'.



The screenshot shows the 'compactUTS - MUX' dialog box. The 'Task' is set to 'Set Single' and the 'Port' is 'Port 2'. Below, there are fields for 'Row' (0) and 'Column' (0), and a 'State' indicator (a green circle).



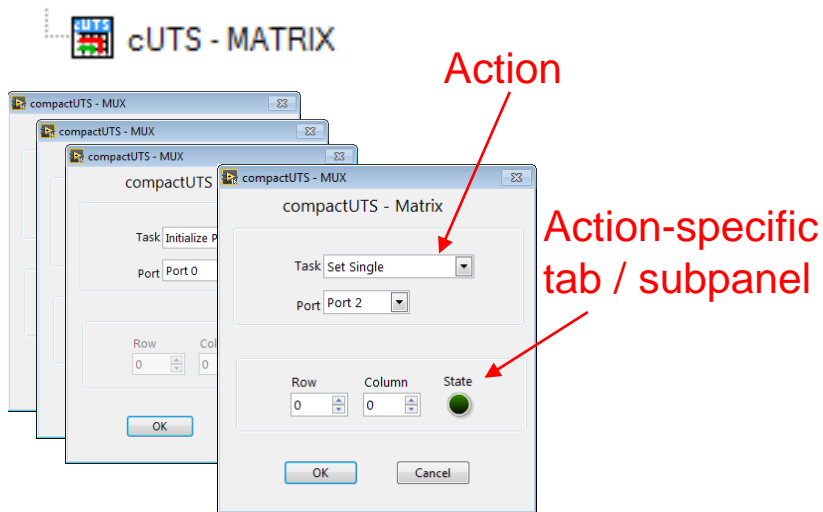
The screenshot shows the 'compactUTS - MUX' dialog box. The 'Task' is set to 'Set All (Port)' and the 'Port' is 'Port 2'. Below, there is a matrix of 2 rows (r0, r1) and 8 columns (c0-c7). The r0 row has a red dot at c0, and the r1 row has a blue dot at c3. 'OK' and 'Cancel' buttons are at the bottom.

- Custom view for each action
- Show only relevant fields
- Reduce ambiguity
- Instrument & UUT Interfaces
- Custom file IO

Common applications of Custom Step Types

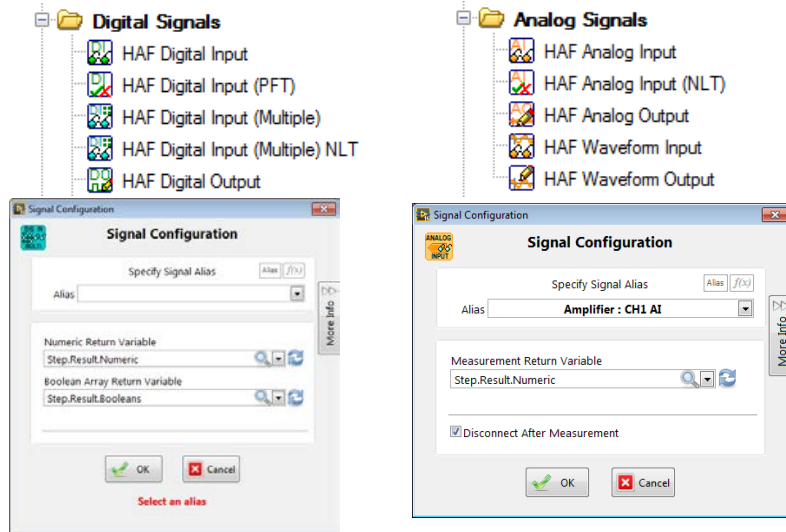
Action engine front-end implementation options

One step for all actions



Best when result data is the same for all actions

One step for each action

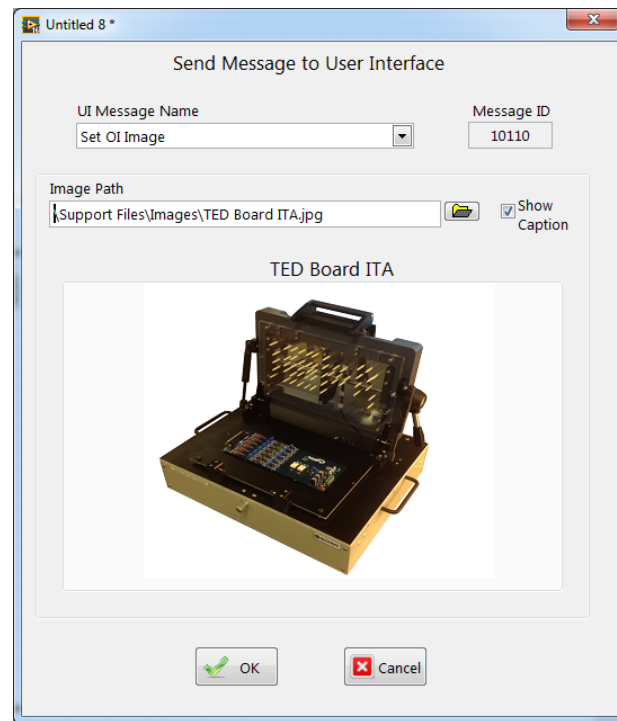


Best when result data is different from action-to-action

Common applications of Custom Step Types

Advanced TestStand Interactions

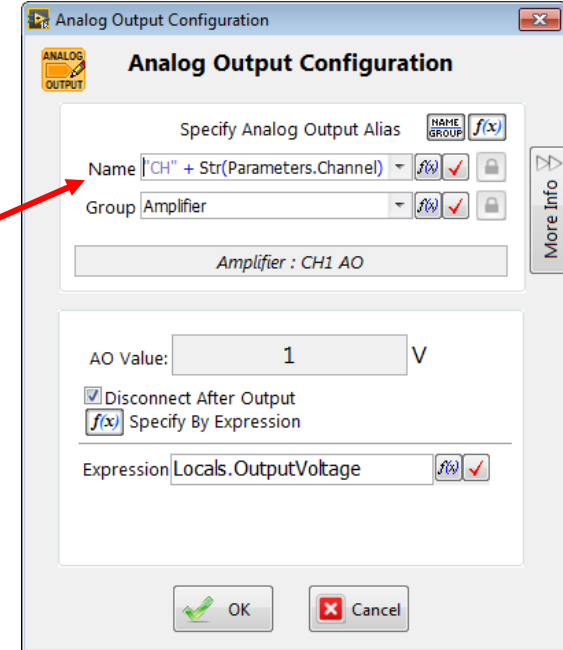
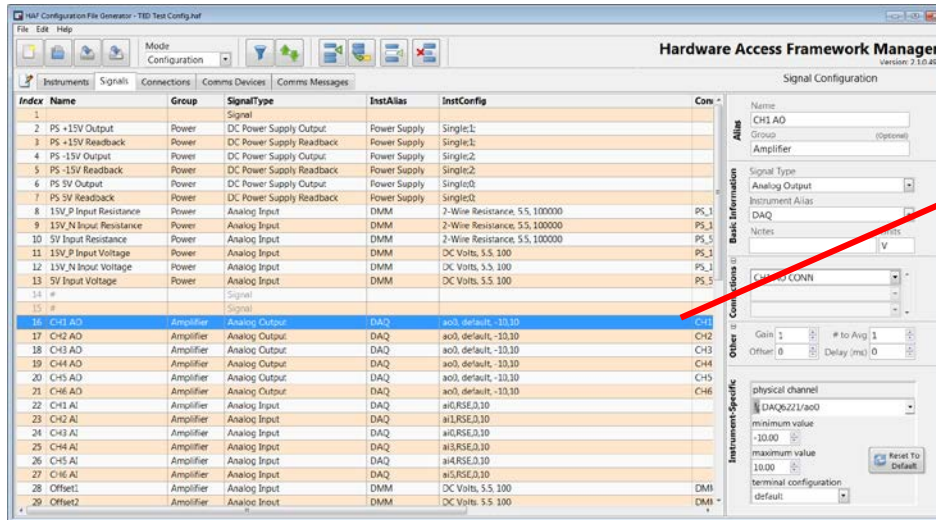
- Custom UI Messages
- Retrieving execution data
 - Serial & part numbers
 - Socket info
- Special process model interfaces



Common applications of Custom Step Types

High Level Architecture Interface

- Multi-IO interactions (Files + Hardware + TestStand API)
 - Hardware Abstraction Frameworks
 - Specialty Tests (eg. multi-limit waveform analysis)



Custom Step Type Guidelines

- Practical
- Intuitive
- Accessible
- Deployable
- Maintainable



Custom Step Type Guidelines

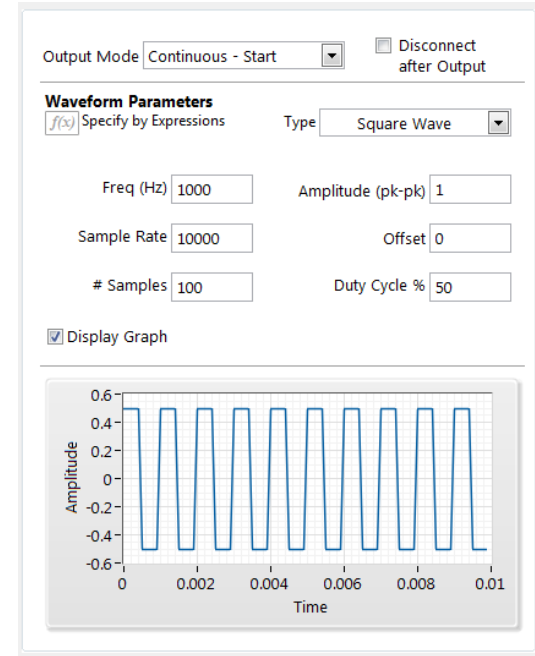
▪ **Practical**

- Intuitive
 - Time saved > time spent
 - Return on investment
- Accessible
 - Will it make developers lives easier?
- Deployable
 - Not every code module should be a CST
- Maintainable



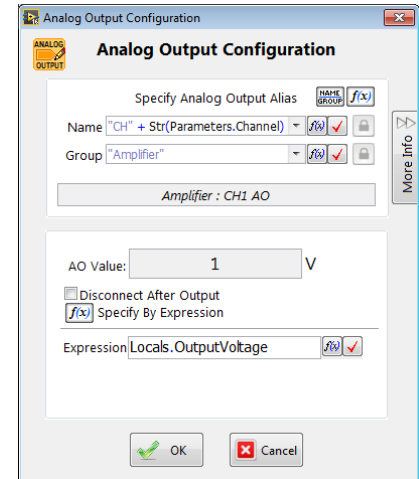
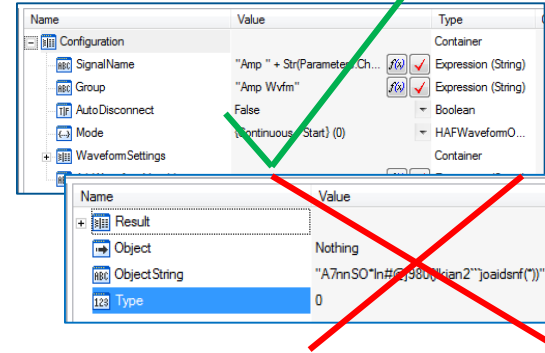
Custom Step Type Guidelines

- Practical
- **Intuitive**
- Accessible
- Deployable
- Maintainable
- Reduce ambiguity
 - Only show pertinent information
 - Include live feedback
- Tip strips and in-dialog help
- Use pre-existing conventions
 - Standard Variables
 - Eg. Step.Result.Numeric
 - Drop-down boxes

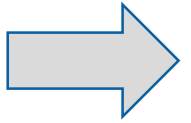


Custom Step Type Guidelines

- Practical
 - Intuitive
 - **Accessible**
 - Deployable
 - Maintainable
- Allow programmatic access
 - Store individual variables in properties
 - Avoid using flattened string data
 - Step.Configuration for Setup
 - Step.Result for return data
 - Interface with TestStand API
 - SequenceContext
 - ExpressionEdit ActiveX object
 - Access variables (eg. locals, parameters)



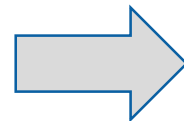
Custom Step Type Guidelines



- Practical
- Intuitive
 - Use Packed Project Libraries for source code
 - Use TestStand Public directories
 - Relative paths, no changes to search directories
- Accessible
- **Deployable**
 - Create installer using TestStand Deployment Utility
- Maintainable



Custom Step Type Guidelines



- Practical

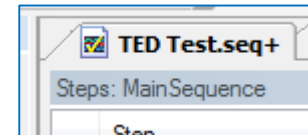
- Intuitive

- Accessible

- Deployable

- **Maintainable**

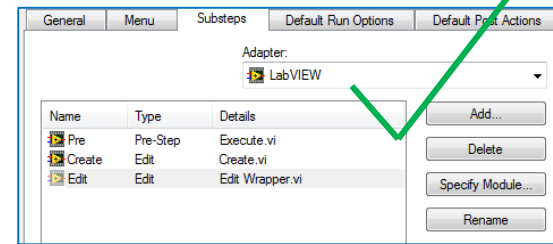
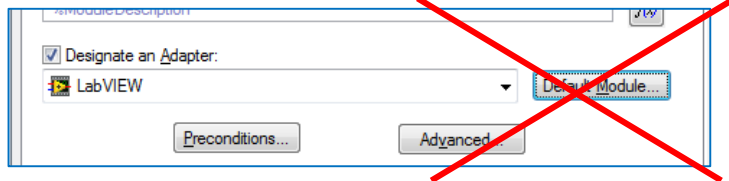
- Use version numbers to update old sequences



- Keep backwards compatibility, or change StepType Name
 - Don't change property names

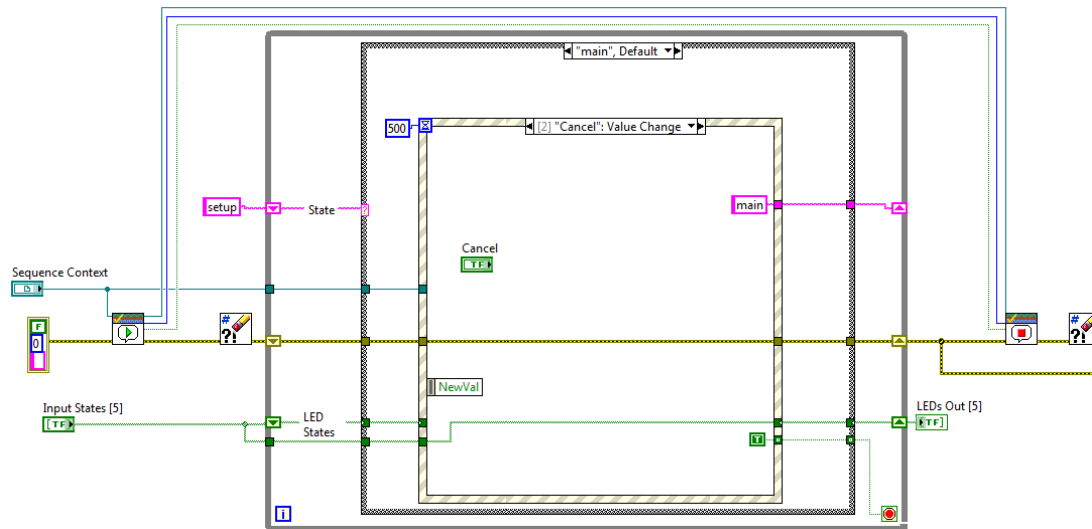
- DO NOT use the default module for runtime code

- Use pre- or post- substeps



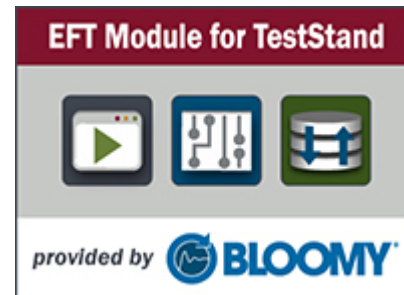
Additional Tips

- Debugging
 - Can't Step Into Pre/Post Steps
 - Open VI first, set breakpoint
- Close References
 - Avoid TestStand object leaks
 - Avoid using API if unsure
- Use TestStand modal dialog and termination monitor functions
- Allow user to "Cancel"
 - Revert config to original



References

- Creating a Waveform Custom StepType and adding information to the ASCII report
 - <http://www.ni.com/tutorial/3184/en/>
- Best Practices for Custom Step Type Development
 - <http://www.ni.com/product-documentation/8300/en/>
- Bloomy EFT Module for TestStand
 - www.bloomy.com/eft-module-teststand
 - <http://sine.ni.com/nips/cds/view/p/lang/en/nid/215009>



Questions?

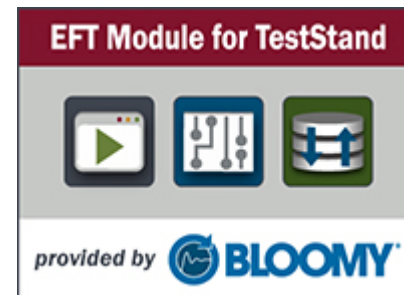
Grant Gothing

grant.gothing@bloomy.com

508 271-7340

Stop by the Bloomy pavilion at **Booth #205**

Follow our blog at www.bloomy.com/support/blog



Stay Connected During and After NIWeek



ni.com/niweekcommunity



facebook.com/NationalInstruments



twitter.com/niglobal



youtube.com/nationalinstruments

Please provide feedback on this session via the NIWeek Mobile App