Controlling the User Interface

Use property nodes, invoke nodes, and control references to programmatically control front panel objects.

A. VI Server Architecture
B. Control References
C. Property Nodes
D. Invoke Nodes

A. VI Server Architecture

What is the purpose of the VI Server and the class hierarchy of properties and methods?

- VI Server Purpose and Use
- Properties and Methods
- VI Class Hierarchy

VI Server Purpose and Use

- Provides programmatic access to LabVIEW
- Use the VI Server to:
  - Programmatically control front panel objects and VIs
  - Dynamically load and call VIs
  - Run VIs on a computer or remotely across a network
  - Programmatically access the LabVIEW environment and editor (Scripting)

VI Server Hierarchy

Objects—Entities that exist within the current application instance.

Properties—Single-valued attributes of the object: read/write, read only, write only

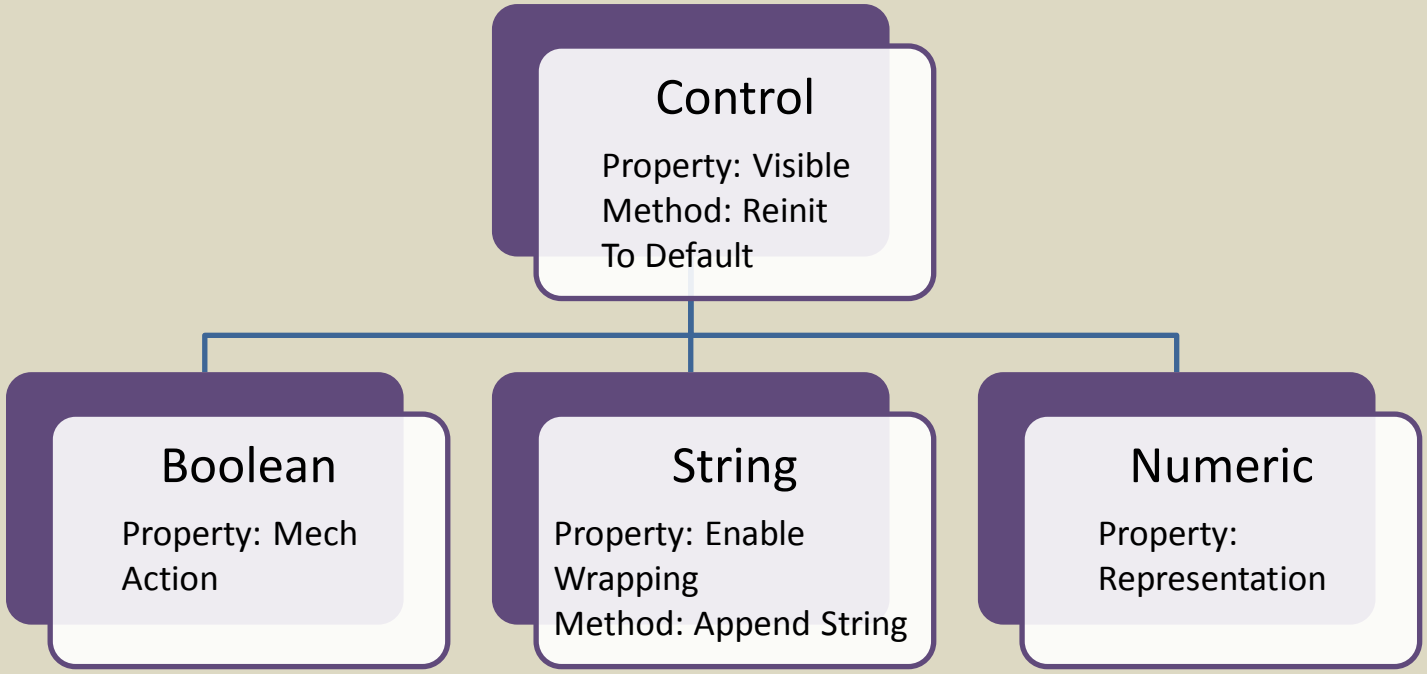
Methods—Functions that operate on the object

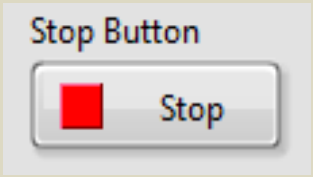
VI Server Hierarchy

Parent Class

Child Class

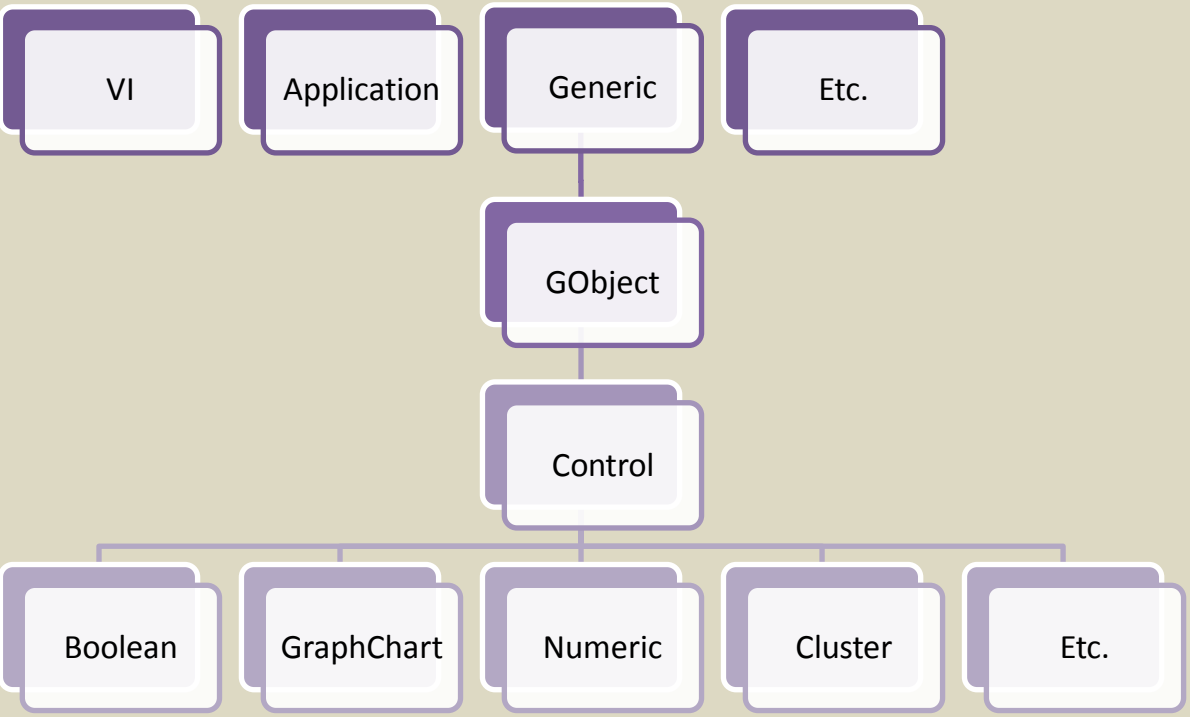
Object

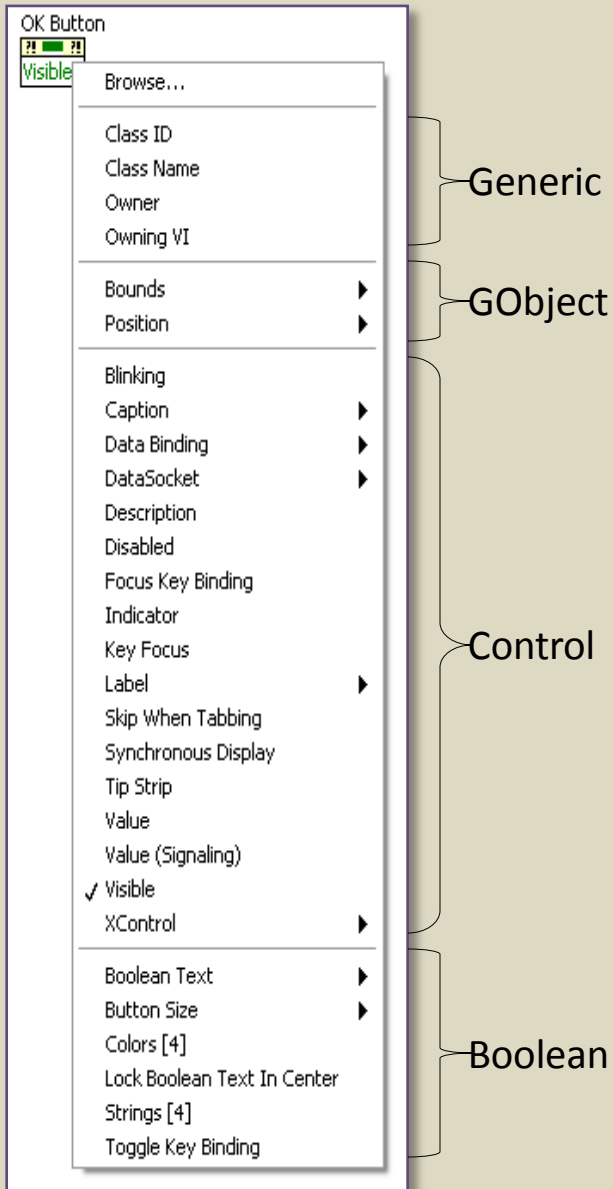




Property Values
Label Text: Stop Button
Visible: Yes
Boolean Text: Stop

VI Server Hierarchy





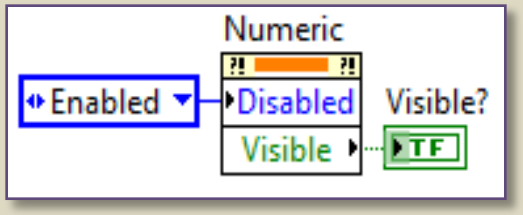
B. Property Nodes

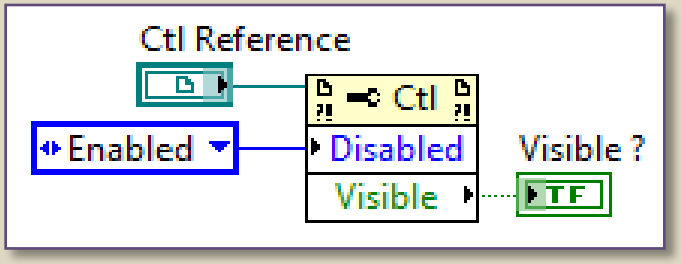
What are property nodes, how do we use them, and why?

- Definition
- Creating Property Nodes
- Execution Order
- Using Property Nodes

Definition

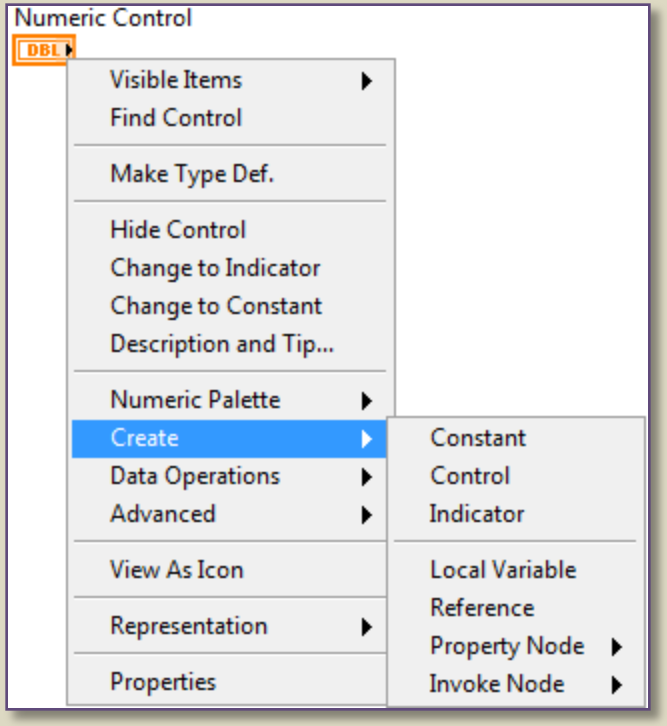
- Block diagram nodes that allow the program to dynamically access and modify properties of objects
- You also can use the Property Node to access the private data of a LabVIEW class
- Implicitly or Explicitly linked to objects by references





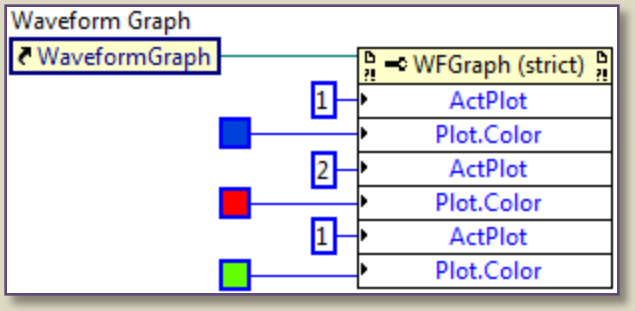
Creating Property Nodes

- Functions > Programming > Application Control > Property Node
- Right click any control or indicator
  - Create Reference to explicitly wire to existing property node
  - Create Property Node for implicit link



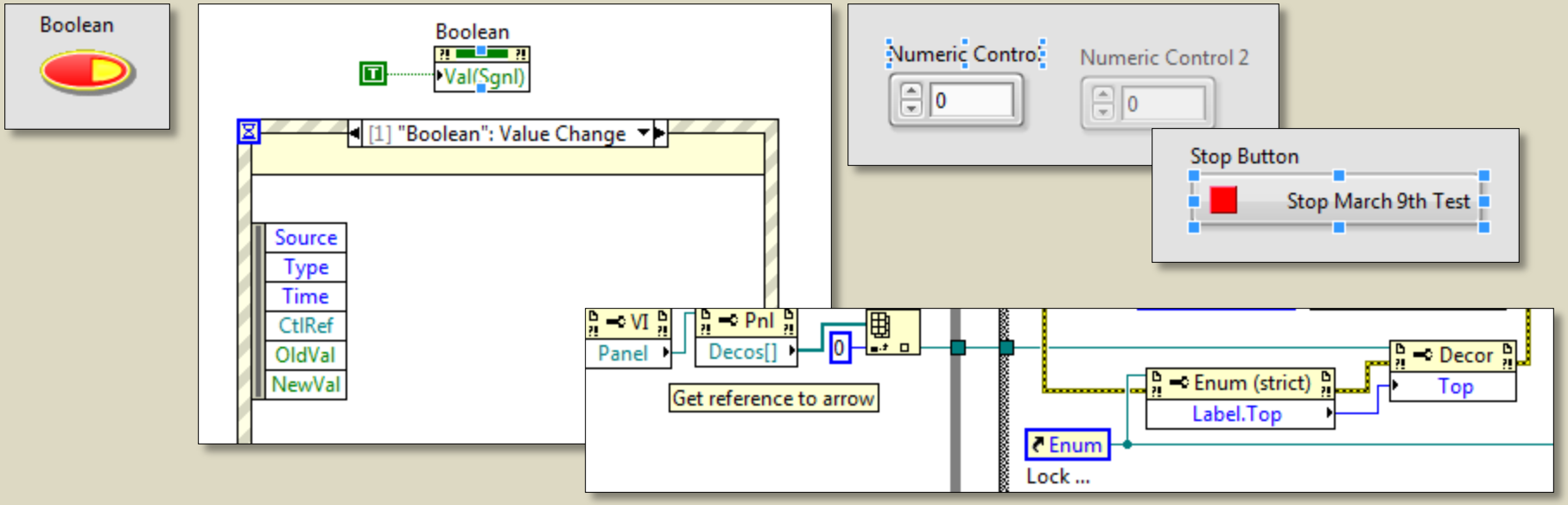
Execution Note

The node executes from top to bottom. The Property Node does not execute if an error occurs before it executes, so always check for the possibility of errors. If an error occurs in a property, LabVIEW ignores the remaining properties and returns an error.



Property Node Uses

- Value Change [Value, Value (Signaling)]
- UI Usability [Text, Disabled, Visible]
- Visual Effects [Bounds, Blinking]



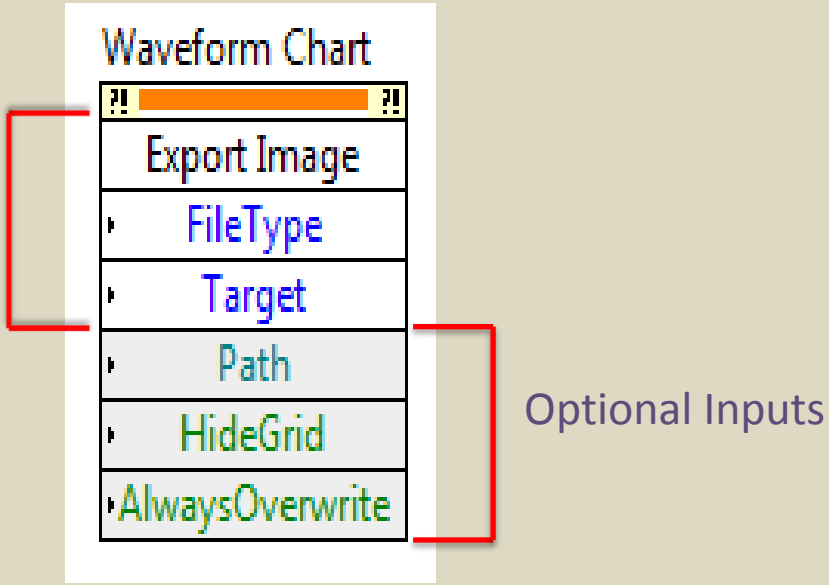
C. Invoke Nodes

What are invoke nodes, how do we use them, and why?

- Definition
- Creating Invoke Nodes
- Using Methods

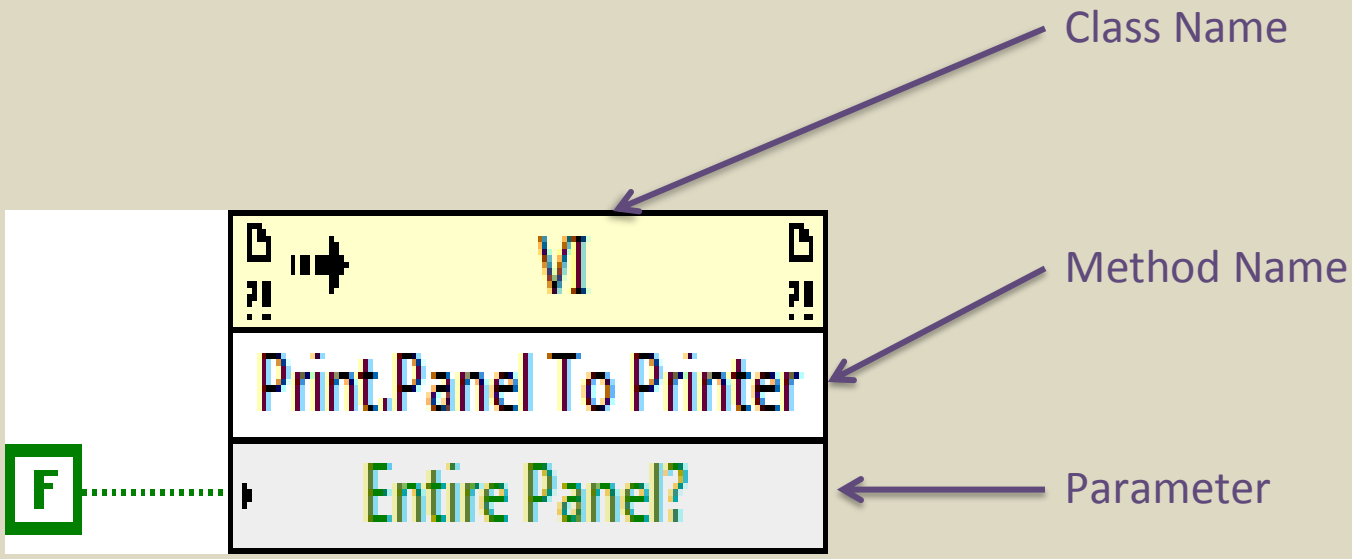
Definition

Invoke Nodes call methods or actions on objects.



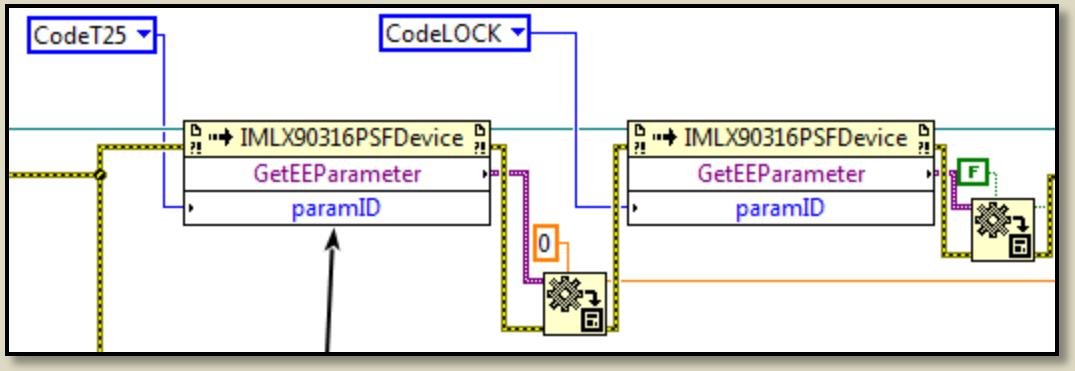
VI Methods

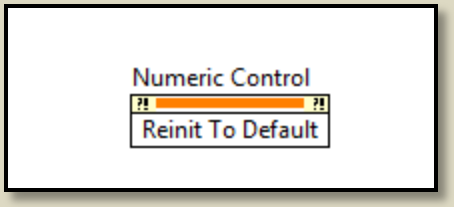
Invoke Nodes call methods or actions on objects.



Invoke Node Uses

- Reinit to Default
- Object-oriented 3rd party drivers
- Print image to file



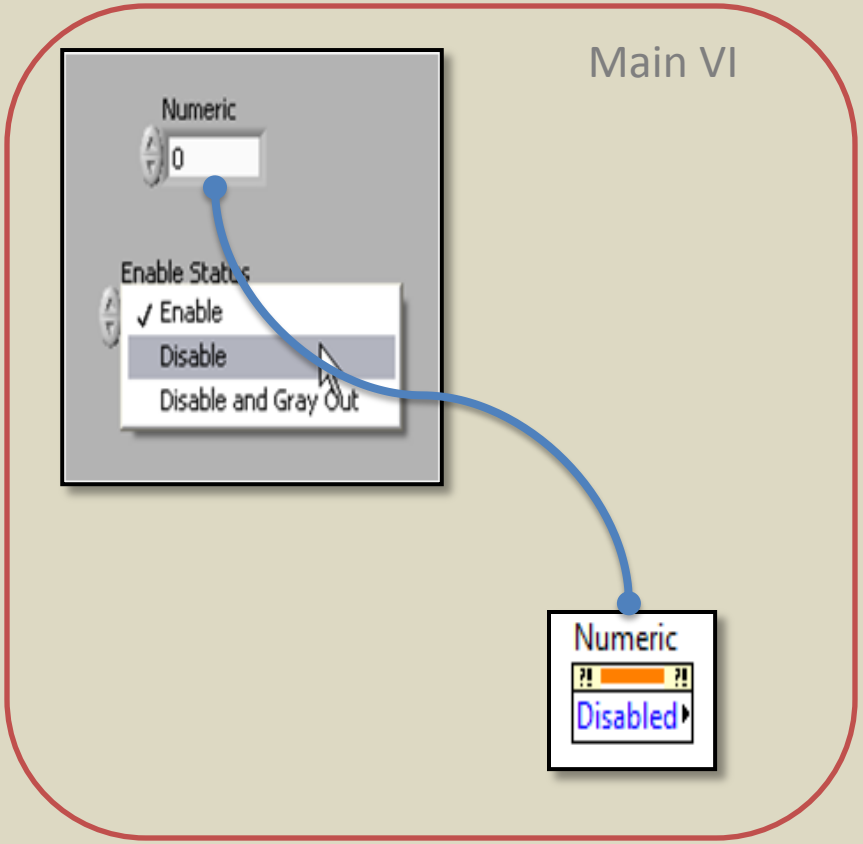


D. Control References

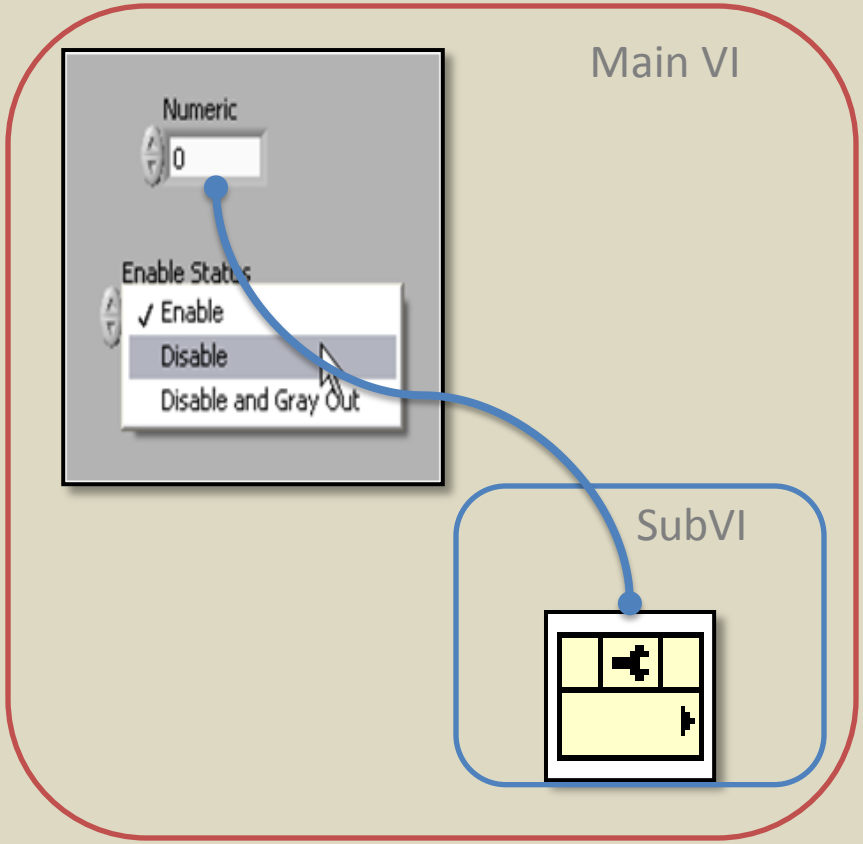
How can we interact with the VI Server?

- Implicitly and Explicitly Linked Property Nodes
- Creating Control References
- Selecting the VI Server Class

Implicit and Explicit Nodes



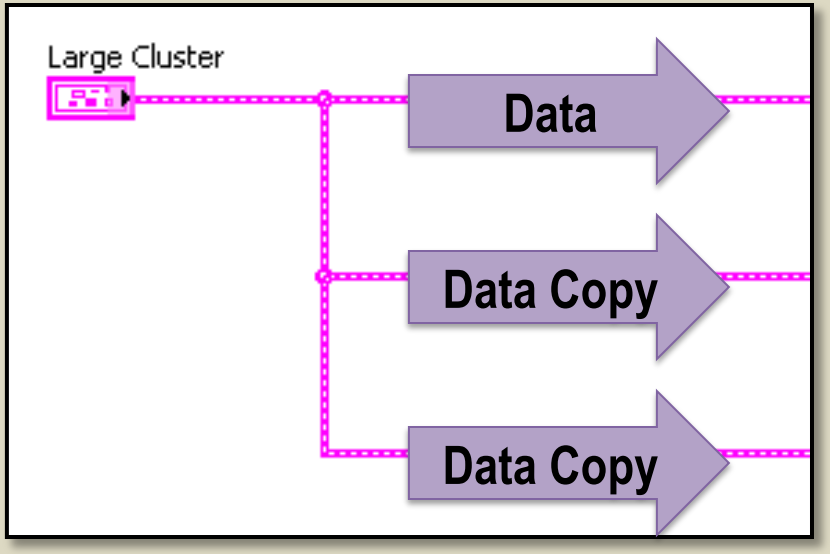
Implicitly Linked Property Node



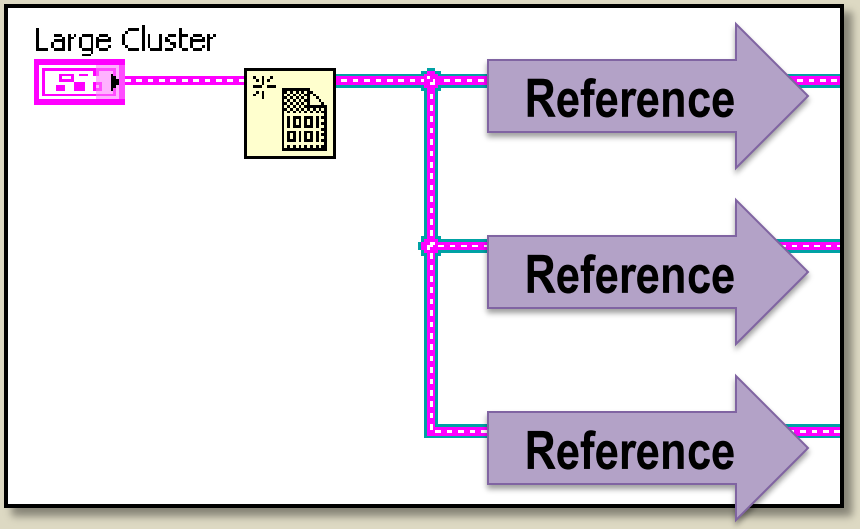
Explicitly Linked Property Node

Data By Reference

Manipulate references to the data instead of the data itself.



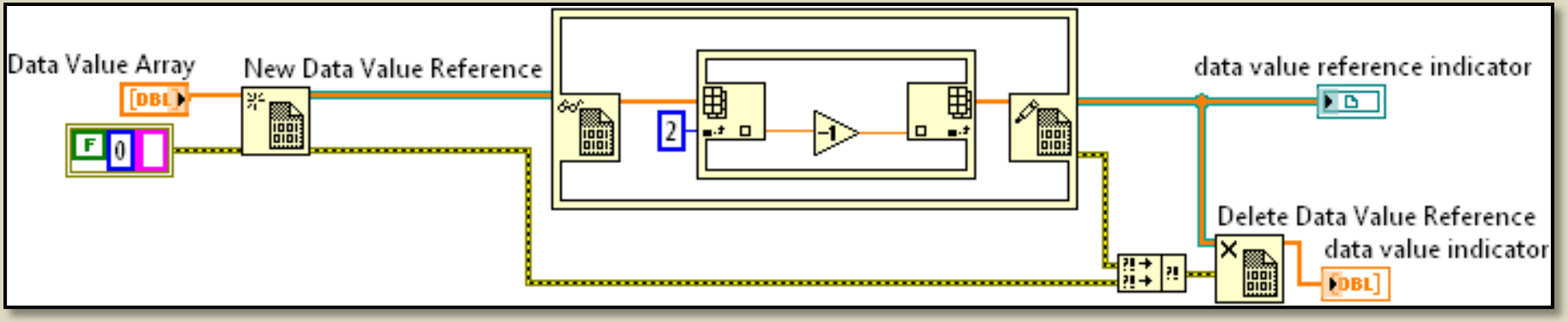
Traditional dataflow: branches may create copies



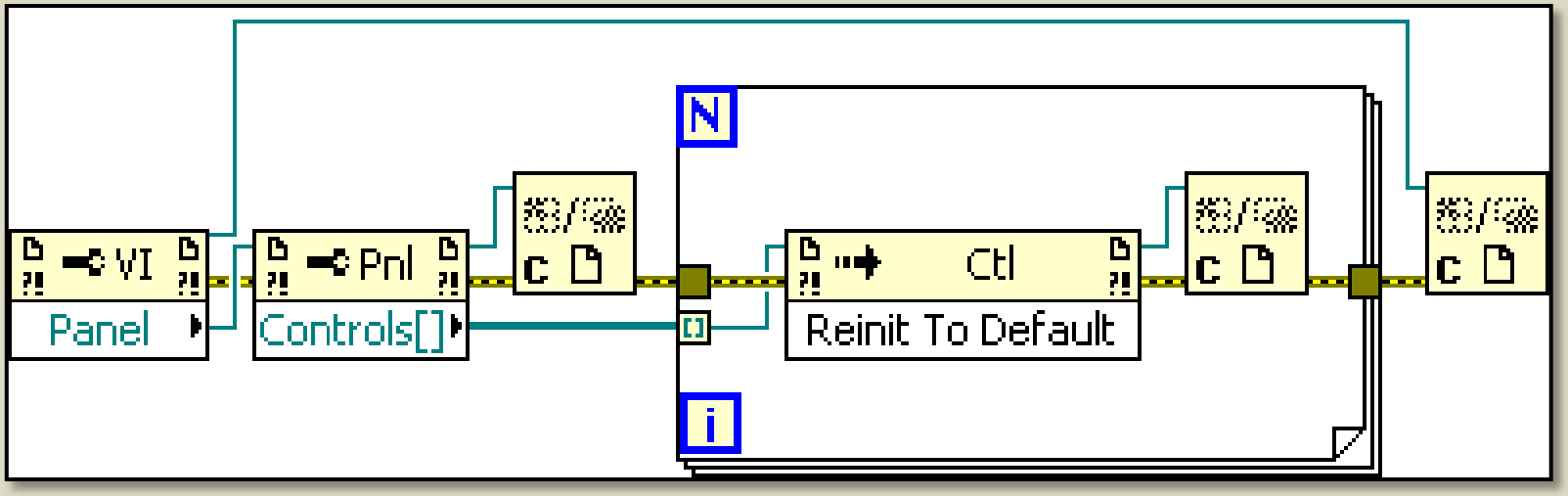
By reference: points to memory location

Operating on Data by Reference

Data Value References must be used in conjunction with the In Place Element structure

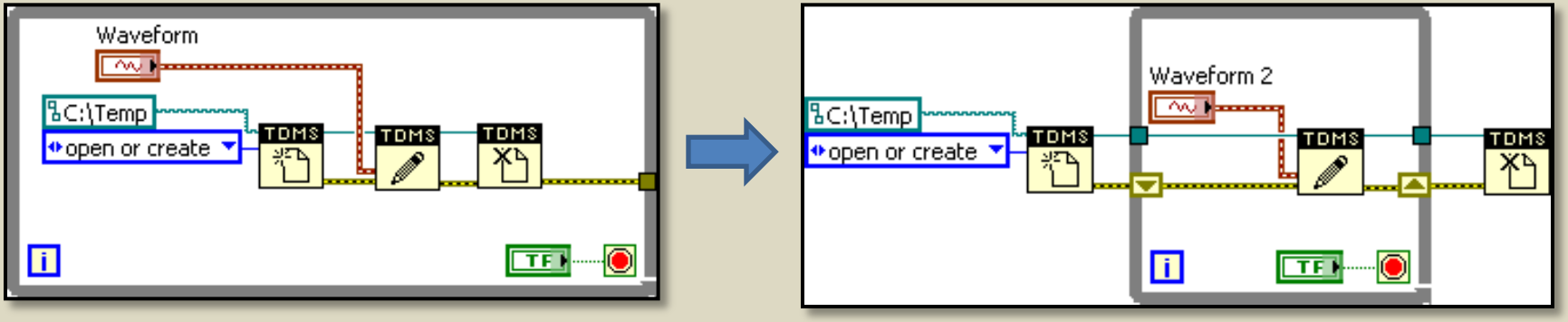


Closing References



References in Loops

Only place code that needs to be repeated inside the loop. Opening and closing references should be outside of the loop.



LabVIEW Cleanup

- LabVIEW cleans up many references when the owning VI goes idle and others when the process closes.
- Manually close references to avoid undesirable memory growth, particularly for long-running applications.

User Interface Property Nodes

- May load the front panel into memory.
- Require an extra buffer allocation for data values sent to or read from the Property Node, in comparison to reading and writing directly to a terminal.

Alexandra's Example