Worldwide Technical Support and Product Information
ni.com

Worldwide Offices
Visit ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

National Instruments Corporate Headquarters
11500 North Mopac Expressway  Austin, Texas 78759-3504  USA  Tel: 512 683 0100

For further support information, refer to the Additional Information and Resources appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the Info Code feedback.
Contents

Student Guide
A. NI Certification .................................................................v
B. Course Description ............................................................vi
C. What You Need to Get Started ...........................................vi
D. Installing the Course Software .........................................vii
E. Course Goals ......................................................................vii
F. Course Conventions ..........................................................viii

Lesson 1
Software Architecture – Introduction
A. Software Architecture – Quality ......................................1-2
B. Documenting a Software Architecture ..............................1-23
C. Architecture Quality .......................................................1-43

Lesson 2
Designing an API
A. APIs .................................................................................2-2
B. API Design Techniques ....................................................2-6
C. Polymorphic VIs for an API ..............................................2-26
D. Project Libraries .............................................................2-35
E. Passing Data in an API ......................................................2-40

Lesson 3
Multiple Processes and Inter-Process Communication
A. Introduction – Processes ..................................................3-2
B. Foundational APIs ..........................................................3-18
C. Foundational Design Patterns .........................................3-47
D. Scaling Foundational Design Patterns ..............................3-79
E. What is an Asynchronous Dynamic Process? ....................3-97
F. By Reference Inter-Process Data Storage .........................3-135
G. Inter-Target Communication ..........................................3-158
H. Analysis of LabVIEW Communication APIs ....................3-165
I. Communication Reference Architectures .........................3-171

Lesson 4
Advanced User Interface Techniques
A. Subpanels – Examples .....................................................4-4
B. XControls Overview – Review of Controls .......................4-17
C. XControl Abilities ..........................................................4-30
D. XControl Properties and Methods ....................................4-52
E. XControl Development – Editing XControls in Use ............4-60
Lesson 5
Introduction to Object-Oriented Programming in LabVIEW
A. What is Object Oriented Programming? ............................................................... 5-2
B. Recommended Resources ..................................................................................... 5-26
C. Class Diagrams and Design (Optional Section) ................................................... 5-30

Lesson 6
Plug-In Architecture
A. Using Plug-In Architecture with VI Server ........................................................ 6-4
B. Using Plug-In Architecture with LVOOP ............................................................ 6-11

Lesson 7
Tips, Tricks, & Other Techniques
A. Variant Attributes ................................................................................................. 7-2
B. Callback VIs ......................................................................................................... 7-4
C. VI Scripting ........................................................................................................... 7-7
D. What is a Drop In VI? ........................................................................................... 7-10
E. Other ..................................................................................................................... 7-25

Lesson 8
Error Handling
A. Basic Error Handling Strategies ........................................................................... 8-2
B. Comprehensive Error Handling Strategies ........................................................... 8-5
C. LabVIEW Implementation Examples ................................................................... 8-21

Lesson 9
Additional Resources
A. Now What? ........................................................................................................... 9-2

Appendix A
Additional XControl Programming Techniques
A. Refresh Section ...................................................................................................... A-2
B. Shortcut Menus ..................................................................................................... A-4
C. Resizing the XControl ......................................................................................... A-7
D. Defining Custom Events ....................................................................................... A-8

Appendix B
Additional Information and Resources
This lesson explores creating a scalable, flexible, and modular user interfaces with subpanels and creating a reusable and distributable user interface component that encapsulates user interface code.

A valuable resource for additional user interface information, including front panel design and downloadable controls, is the UI Interest Group on NI Community at https://decibel.ni.com/content/groups/ui.
B. XControls Overview – Review of Controls

There are several types of controls available in LabVIEW. Choosing the right type of control can be as important as choosing the data type for a control. Except for XControls, the following control types are described in detail in the LabVIEW Help and other LabVIEW courses.

- **Custom Control**—Allows multiple developers to start with a consistent look and data type for a control. However, a developer can modify a custom control after adding it to a front panel window, and there is no linking or updating between a control instance and the control file. Use these controls as templates to give developers a starting point. Custom controls, type definitions, and strict type definitions give you more control over the appearance of a control than the standard control properties.

- **Type Definition**—Enforces the data type, but not the appearance of the control. Any changes to the control file update all instances of the control. Use type definitions for non-visible controls, or for constants on the block diagram, such as clusters and enumerated values. Type definitions give developers freedom in the appearance of visible controls while still maintaining a consistent data type.

- **Strict Type Definition**—Enforces the data type and the appearance of a control. The data type and appearance of every control instance are linked to the control file and receive any updates made to the file. Use strict type definitions when you want a control to maintain a consistent appearance throughout an application.

- **XControl**—Allows you to include code with a control.

- **Other controls**—You can also develop or use ActiveX or .NET controls for your project. Functionally, these controls are similar to XControls.
What is an XControl?

- XControls allow the architect to create controls with custom appearance and edit-time and run-time functionality.
- For the user, the XControl has the same functionality as other controls.

What is an XControl?

- XControls allow the architect to create controls with custom appearance and edit-time and run-time functionality. The architect can distribute XControls to their developers to use.
- For the user, the XControl has the same functionality as other controls. Developers can place XControls in VIs like other controls and create property nodes for them.
XControl

Simple Xgraph

<Exercises>\...\Demonstrations\XControl\Simple XGraph\Test X Graph Simple XControl.vi

XControl

Explore the XControl in the <Exercises>\...\Demonstrations\XControl\Simple XGraph directory.

Xgraph simple.xctl is the Xcontrol.

Run Test X Graph Simple XControl.vi to see the XControl used in a VI.
Designing an XControl

What is an XControl?
A control with custom behavior in addition to custom appearance.

What aspects of an XControl need to be defined?
If you were to design an XControl, what aspects of the control would you have to define? You may not yet understand the mechanics of building an XControl, but you can still gather the basic requirements and then explore how those requirements map to the mechanics.

The goal is to look at XControls from a high level. What are all of the aspects that you would need to identify? Then, map each aspect to one of the four required XControl abilities. For example, you must define the data type of the XControl, which corresponds to the Data ability covered in later slide.
Advantages of XControls

XControls have the following advantages over standard controls and design patterns:

- **Encapsulate UI code**—You can use XControls to abstract code specific to the user interface from the main architecture of a project. UI code includes tasks such as setting display properties, formatting data for display, and responding to events. Because UI code often exists in the top-level VI, UI code in LabVIEW can increase the complexity of the main VI.

  Because LabVIEW limits dynamic changes to the user interface, UI code often contains tricks, such as moving or hiding controls, which may distract a developer or tester from the code that governs the functionality of the program. UI code can also impose requirements on the design pattern used. For example, if the program must respond to events, even for a single control, the program must use an event-based architecture. Placing the UI code in an XControl frees the main application from the complexity and constraints of UI code.

- **Create reusable and distributable UI components**—An architect can create a library of XControls and distribute them to multiple developers for use throughout the project. XControls make this process as robust as possible because an XControl consists of a single library that contains all code related to the control. XControls also allow continuous improvement. You can update an XControl after distribution, and the developer can define the behavior of existing control instances to ensure that the integrity of the control remains intact and settings are not lost.

- **Can have custom dynamic run-time and edit-time behavior**—Unlike custom controls, which you use to create custom user interface components that vary cosmetically from built-in LabVIEW controls and indicators, XControls have dynamic run-time and edit-time behavior that is defined by VIs that run in the background.
Typical Uses of XControls

You can use XControls to accomplish the following tasks:

- **Add functionality to existing controls**—You can use an XControl to expand the functionality of existing controls. Encapsulating a control in an XControl allows you to add properties and methods. You can also add shortcut menu options and/or configuration dialog boxes for the control without adding code to the client application. XControls also allow you to alter the data type of the control and include inherent data analysis or processing in the control. When you encase an existing control in an XControl, you must replicate any edit-time or VI Server capabilities of the original control that you intend to use, such as configuration dialog boxes, properties, and methods.

  Refer to [ni.com/info](http://ni.com/info) and enter the info code rdsxca to view the *String XControl with Autocomplete Functionality* example on the NI Developer Zone for more information about using XControls to add functionality to existing controls.

- **Combine Existing Controls**—You can combine the functionality of two or more existing controls to simplify the application using the controls. If your user interface contains controls which affect only the display or analysis of data for another control, you can combine these controls into an XControl and assign a single data type to both controls. This prevents you from having to interact with the configuration controls in your client application.

  Refer to [ni.com/info](http://ni.com/info) and enter the info code rdcnfl to view the *Creating New Front Panel Objects with LabVIEW XControls* tutorial on the NI Developer Zone for more information about using XControls to combine existing controls.
Typical Uses of XControls (continued)

• Abstract UI Components
  – Use event-based programming without using an event-based design pattern
  – Separate an entire UI from the rest of the implementation

• Create New Controls
  – Use the picture control
  – Custom graphs are a common example

Refer to the <Exercises>\Advanced Architectures in LabVIEW\Demonstrations\XControl directory for an example of using XControls to create custom graph controls.
When to Use XControls

When you should use XControls:
• To create reusable components with dynamic behavior
• To encapsulate extended functionality of a control
When you should not use XControls:
• To accomplish purely cosmetic changes
• When working on a single-shot application that will not use the XControl in several places
Use Strict TypeDefs instead when:
• You do not need dynamic run-time and edit-time behavior
Another XControl Example…

One of the LabVIEW examples includes an XControl called the Dual Mode Thermometer control. This simple XControl example is a good example you can use to learn the basic components of an XControl, which can often be confusing to developers new to XControls.

The Dual Mode Thermometer XControl is one single control with the ability to keep track of whether or not it is displaying temperature in Celsius or Fahrenheit. Furthermore, it also has the ability to automatically convert temperature from one to the other. This is all possible using a series of regular controls and logic built into a standard VI, but the Dual Mode Thermometer Control bundles all of this functionality into one reusable control package.

The following requirements make the Dual Mode Thermometer Control a good use case for developing an XControl:

1. Thermometer must keep track of which temperature unit it represents
2. Thermometer must convert temperature between Fahrenheit and Celsius
3. Thermometer must represent a single numeric value and look (visually) like a thermometer
Dual Mode Thermometer XControl

Demonstrate the Dual Mode Thermometer XControl example in the NI Example Finder.

**Dual Mode Thermometer XControl**

1. Open the NI Example Finder.
2. On the Search tab, search for the keyword `xcontrols`.
3. Double-click `Simple Dual Mode Thermometer XControl.lvproj` to open the project.
4. From the Project Explorer, open `Run Dual Mode Thermometer XControl.vi`.
5. Run the VI.
6. Demonstrate the functionality of the Dual Mode Thermometer XControl.

Many of the block diagrams shown in the XControl section are from the Dual Mode Thermometer XControl example.
Using XControls

- Manage XControls via the Project Explorer
- XControls appear as regular terminals
- You start with shell code

Using XControls

You create XControls in the Project Explorer. In the Project Explorer window, right-click My Computer and select New»XControl from the shortcut menu. LabVIEW creates a new XControl library (.xctl) in the project. Within the XControl library, LabVIEW automatically creates required components. You just need to open and edit each component to add functionality.

The XControl library file is an XML file that includes the names and locations of each component.
XControl Library File

The XControl library file (.xctl) is an XML file that includes the names and locations of each component.
Structure of an XControl

The basic structure of an XControl includes:

- Properties that define the characteristics or descriptive information of the XControl
- Methods that define what actions the XControl can take
- Abilities that define the data, appearance, and initialization of the XControl
C. XControl Abilities

XControl abilities are fundamental parts of an XControl that you implement to define the behavior of the control, such as data type, appearance, and initialization behavior. LabVIEW automatically calls or uses XControl abilities at appropriate times. The following required abilities must be present in any XControl:

- **Data**—Type definition VI that defines the data type of the XControl.
- **State**—Type definition VI that defines additional persistent data for the XControl.
- **Init**—VI that initializes the control when an instance is dropped or loaded.
- **Facade**—VI that defines the appearance and main behavior of the XControl. An XControl can also have optional abilities. Right-click the XControl and select **New Ability** to add optional abilities.

XControls have the following optional abilities:

- **Uninit**—Cleans up resources created inside the XControl.
- **Convert State for Save**—Alters the state of the XControl as an instance of it is saved.
Exercise 4-1: Creating an XControl (Optional)

To create an XControl that functions as a scalable radio control button.