LabVIEW Certification Overview

The National Instruments LabVIEW Certification Program consists of the following three certification levels:

- Certified LabVIEW Associate Developer (CLAD)
- Certified LabVIEW Developer (CLD)
- Certified LabVIEW Architect (CLA)

Each level is a prerequisite for the next level of certification.

A CLAD demonstrates a broad and complete understanding of the core features and functionality available in the LabVIEW Full Development System and possesses the ability to apply that knowledge to develop, debug, and maintain small LabVIEW modules. The typical experience level of a CLAD is approximately 6 to 9 months in the use of the LabVIEW Full Development System.

A CLD demonstrates experience in developing, debugging, deploying, and maintaining medium-to-large scale LabVIEW applications. A CLD is a professional with an approximate cumulative experience of 12 to 18 months developing medium-to-large applications in LabVIEW.

A CLA not only possesses the technical expertise and software development experience to break a project specification into manageable LabVIEW components but has the experience to see the project through by effectively utilizing project and configuration management tools. A CLA is a professional with an approximate cumulative experience of 24 months in developing medium-to-large applications in LabVIEW.

Note The CLAD certification is a prerequisite to taking the CLD exam. The CLD certification is a prerequisite to taking the CLA exam. There are no exceptions to this requirement for each exam.

CLA Exam Overview

A CLA demonstrates mastery in analyzing and interpreting customer requirements for the development of scalable application architectures in LabVIEW, organized in a modular project hierarchy with the intent to be fully developed later by a multi-developer team. The architecture meets the requirements using a software simulation with interfaces to abstracted hardware modules. To ensure successful integration, the Architect enables completion by a developer team with modules designed with consistent, well defined interfaces, data structures, module design patterns, messaging, and documented developer instructions with specific design requirements. The CLA is a professional with an approximate cumulative experience of 24 months in developing medium-to-large applications in LabVIEW.

Product: Your test computer will have the latest LabVIEW Full or Professional Development System installed for developing your application. Contact your proctor or testing center prior to the exam to get the details and familiarize yourself with the LabVIEW version that you will use to develop your application.
Refer to `ni.com/labview/how_to_buy.htm` for details on the features available in the LabVIEW Full / Professional Development System.

Please note that you will not receive extra exam time to compensate for non-familiarity with the LabVIEW environment. If you need time to customize the environment, please make arrangements with your proctor to hold off on giving you the exam packet until you are ready to start the exam.

The use of resources available in LabVIEW, such as the *LabVIEW Help*, examples, and templates are allowed during the exam. Externally developed VIs or resources are prohibited.

The CLA exam consists of a project that is very similar to the project that you worked on the CLD exam.

Your exam submissions should be transferred to a USB memory stick and turned in to your proctor.

Note **Do *not* detach the binding staple, copy, or reproduce / retain any section of the exam document or solution of the exam. Failure to comply will result in failure.**

CLA Exam Topics

1. Project Requirements
2. Project Organization and Hierarchy
3. Project Architecture and Design
4. Team-Based Design, Development, and Standardization Practices
5. Reusable Tools / Component Design

Note The CLA exam is cumulative and includes CLAD and CLD exam topics.

Topic	Subtopic
1. Project Requirements	a. Technical requirements b. Requirements tracking c. GUI Development d. Hardware Interface e. Error Handling
2. Project Organization and Hierarchy	a. LabVIEW project hierarchy b. Disk hierarchy c. LabVIEW paths d. Modular hierarchy
3. Project Architecture and Design	a. Main VI architecture b. Module / SubVI architecture c. Messaging Architecture d. Error Module e. File and Database I/O f. Simulation architecture g. User interface design h. Advanced design methods i. Documentation of requirements
4. Team-Based Design, Development, and Standardization Practices	a. LabVIEW development practices b. Modular functionality c. Documentation for Developer completion d. Clear Modular APIs
5. Reusable Tools / Component Design	a. LabVIEW technologies b. API design c. Design patterns

CLA Exam Topic Details

1. **Project Requirements**
   a. Technical requirements
      Determine and list the following requirements from the project specification:
      1. Application requirements—Goal and purpose of the application
      2. User interface requirements—Presentation and behavior of controls that interact with users
      3. Functional requirements—The functionality of the components and their interaction within the system
      4. Timing requirements—Hardware / software, event-based data overflow, daylight savings
      5. Error handling requirements—Warning, errors, critical errors, shutdown sequence
      6. Hardware or simulation requirements—Interface and operational requirements for switching to field devices
      7. Input/output requirements—Console, databases
      8. Initialization, shutdown requirements—User interface and program behavior during startup, error conditions, and shutdown
      9. Non-functional requirements—Accuracy, performance, modifiability
      10. Assumptions and constraints
          a) A functional assumption is an issue that is unclear in the specification
          b) A functional constraint is a design decision that is imposed by the specification

   b. Requirements tracking
      1. Identify and fulfill requirements
         a) Determine detail level of requirements
         b) Locate requirements tags in architecture only where requirements are fulfilled
      2. Methods or (utilization of) software tools to track requirements
         a) Use specified format for requirements tags for Requirements Gateway tracking
         b) Utilize provided tag file

   c. GUI Development
      1. Build GUI based on specification
         a) Determine the appropriate control type based on functional specifications
         b) Use Type Definitions
      2. Design interface that meets functional requirements
         a) Utilize the LabVIEW Development Guidelines
         b) Organize, modularize, or group user interface components to follow a process, or logical sequence
         c) Utilize advanced LabVIEW development techniques

d. Hardware Interface
   1. Use abstraction to separate simulation and hardware modules
      a) Develop API to interface with the controller module
      b) Design a scalable interface that enables transition from simulation to hardware
      c) Develop method to select hardware or simulation modules
   2. Develop simulation architecture based on hardware
      a) Select a modular architecture that simulates external hardware
      b) Select user interface components that closely mimic the function of the hardware

e. Error Handling
   1. Develop centralized error handling
      a) Demonstrate methods to handle warning, critical errors, and shutdown error conditions as defined in the specification
      b) Develop architecture that integrates the error module in the main VI and within other modules
   2. Design a shut down method that is responsive to the error module and is abstracted from the functional modules

2. Project Organization and Hierarchy

a. LabVIEW Project hierarchy
   1. Develop a LabVIEW Project hierarchy for team-based development
      a) Modules and their hierarchy
      b) Shared subVIs, custom controls
      c) Plug-in VIs
      d) LabVIEW Project libraries
      e) Support files (documentation, configuration, and log files)
   2. Utilize a naming convention

b. Disk hierarchy
   1. Mimic project hierarchy on disk
   2. Use auto-populating folders
   3. Organize project and disk hierarchy by module or other functional based scheme

c. Paths
   1. Utilize and require developer to use relative paths

d. Modular Hierarchy
   1. Organize by module or other functional based scheme
   2. Sub folders based on code artifacts such as controls or module subVIs

3. **<u>Project Architecture and Design</u>**
   a. <u>Main VI architecture</u>
      1. Select an advanced, scalable, and modular architecture that enables the following:
         a) Handling of user interface events and user events
         b) Asynchronous and parallel processing of events
         c) Initialization, shutdown, state persistence, and restoration
         d) Effective error (logic and run-time) handling
         e) Timing (event or poll-based)
         f) Team-based development of functional modules
      2. Develop data and event messaging structures
      3. Develop architecture to handle configuration data
      4. Develop interfaces for simulation and other modules
      5. Utilize the LabVIEW Development Guidelines for memory optimization
   b. <u>Module / subVI architecture</u>
      1. Select a cohesive architecture and design pattern for modules and subVIs
      2. Define and develop a clear API
      3. Define a consistent connector pane and icon
   c. <u>Messaging Architecture</u>
      1. Modularize messaging scheme for initialization, use, and shutdown
      2. Demonstrate method for messaging for developers to complete
      3. Demonstrate loose coupling of messaging module
   d. <u>Error Module</u>
      1. Modularize centralized error handling for clear initialization, use, and shutdown
      2. Demonstrate error handling integration with functional modules
      3. Integrate for shutdown as specified
      4. Demonstrate file logging
      5. Discriminate actions for different types of error severity
   e. <u>File and Database I/O</u>
      1. Modularize I/O for clear initialization, use, and shutdown
      2. Communicate access methods for developers to complete
      3. Specify data formats and conversion to application data structures
      4. Integrate for Configuration data and Error logging
   f. <u>Simulation module architecture</u>
      1. Select a modular architecture that simulates external hardware
      2. Design a scalable interface that can ease transition from simulation to hardware
      3. Select user interface components that closely mimic the function of the hardware

- g. <u>User interface design</u>
  1. Utilize the *LabVIEW Development Guidelines*
  2. Organize, modularize, or group user interface components to follow a process, or logical sequence
  3. Utilize advanced LabVIEW development techniques
- h. <u>Advanced design methods</u>
  1. Develop an architecture for a modular, scalable, and maintainable application
  2. Implement, develop, and enhance standard design patterns to suit project requirements
  3. Utilize an event-based design for user interface events and define user generated events for timing, error, signaling, and so on
  4. Abstract functionality and develop a clear and consistent interface API for modules and subVIs
  5. Utilize and standardize scalable data types and data structures
  6. Utilize object oriented design, recursion, VI Server, and advanced file I/O techniques
- i. <u>Documentation of Requirements</u>
  1. Utilize the *LabVIEW Development Guidelines*
  2. Document the following:
     - a) Main architecture for module integration
     - b) Data structures and data and message communication mechanism
     - c) Modules, subVIs, and interfaces (API)
     - d) Simulation module, interfaces, and requirements for transitioning from simulation to hardware module

4. Team-Based Design, Development, and Standardization Practices

- a. <u>LabVIEW development practices</u>
  1. Establish and use consistent development style—Utilize the *LabVIEW Development Guidelines* as well as company developed standards
  2. Use templates as a starting point for development
  3. Document VI Properties, the block diagram, and the user interface (tip strips, and so on)
  4. Develop reusable modules and tools to standardize development
- b. <u>Architecture for modular development</u>
  1. Select a cohesive architecture and design for modules and subVIs
  2. Define a consistent connector pane and icon
  3. Define error handling and ensure critical errors are handled appropriately
  4. Develop major structures and messaging
  5. Develop sufficient detail for the developer to implement the specific requirements.
- c. <u>Documentation instructions for the developer to complete the application</u>
  1. Document coding completion of algorithms, transactions, and logic for a developer team to complete functionality

2. Document multiple similar steps by completing a first step with detailed requirements and subsequent steps referring to the technique of the first step
3. Use LabVIEW code on the block diagram to demonstrate technique and compliment this with developer instruction documentation

d. <u>Clear modular APIs</u>
1. Define the APIs for modules and subVIs
2. Develop APIs for functional modules to enable modularity and abstraction
3. Develop Architecture with APIs for error handling, initialization, and shutdown

5. <u>Reusable Tools / Component Design</u>

a. <u>LabVIEW technologies</u>
1. Determine the optimal method for developing a reusable component or a productivity enhancement tool from the following technologies:
    a) Custom controls
    b) Merge VI
    c) SubVI
    d) XControls
    e) VI template

b. <u>API design</u>
1. Develop a simplified API to wrap advanced LabVIEW functions
2. Develop manager VIs to handle common tasks, such as reference management of queues, user events, and so on
3. Utilize project access options to restrict or allow access to components of libraries

c. <u>Design Patterns</u>
1. Select appropriate design patters for modules and subVIs based on functional requirements
2. Use documentation to describe the completion of routine design pattern elements

CLA Exam

In the CLA exam you will be required to design an architecture that covers the requirements given in a project specification.

Exam Duration: 4 hours
Style of exam: Application architecture development
Passing grade: 70%

Application Architecture Development:
You must develop an application framework consisting of a main VI, modules, supporting subVIs, and custom controls (type definitions). A module is a subVI or group of subVIs that performs a set of functions. A module may have it own hierarchy of subVIs.

Note You are *not* required to submit a functional application. The functional details of the requirements should be documented in the main VI, modules, and subVIs. You must provide this documentation in the architecture to enable developers on your team to develop the functionality.

The architecture has the following minimum requirements:

a. Develop a project hierarchy
b. Develop a main VI. The main VI should include the following:
   i. Modular User interface
   ii. Driving architecture
   iii. Major data structures
   iv. Event, data, timing, and error communication method(s)
   v. Error handling
   vi. Fully connected modules and /or subVIs
c. Develop shell (stub) modules and subVIs, which should *not* include detailed functional logic, but should include the following:
   i. Inputs, outputs, icon, and connector pane
   ii. Architecture and API
   iii. Major internal data structures
   iv. Error handling and error communication
   v. Instructions or comments listing the functional details, which are sufficient for a developer to complete the functionality of the VI
d. Develop an interface for hardware simulation as a separate module or as part of the main VI or any other module, depending on your design.
e. Develop inter-process communication mechanism
f. Develop an error handling and shutdown strategy

Requirements Tracking
The project specification will detail requirements identified by a unique identifier. In order to demonstrate coverage of a requirement, you must include the ID of the requirement in the documentation of your architecture. Requirements can be covered in any part of the architecture's documentation, including:

- VI Documentation Property
- Control Documentation Property
- Project or Library Documentation Property
- Comments on the front panel or block diagram

A single requirement may be covered by multiple sections of code if all of those sections are necessary to fulfill the requirement.
To cover a requirement, the following text should be in the in the documentation of the code: `[Covers: ID]`   Example: `[Covers: CD1]`

The provided USB memory stick contains a text file that has all of the Tags. This file is provided as a convenience for use in placing the tags in the application code.

Note A requirements tracking tool (Requirements Gateway) will be used to verify the requirements coverage, hence adherence to the above syntax is crucial.

Please refer to the CLA sample exams to see the how the coverage is documented in the VI and the instruction / comments that need to be included in the VI for a developer to complete the implementation.

Grading:
The point allocation for the CLA exam consists is as follows: (Total: 100 points)

• User interface and block diagram style	: 10 points
• Documentation	: 20 points
• Requirements coverage	: 30 points
• Architecture development	: 40 points

CLA Exam Preparation Resources

Use the following resources for additional exam preparation:

- Managing Software Engineering in LabVIEW
- Advanced Architectures for LabVIEW
    - Instructor-led training
    - Self-paced training using the course manuals
- CLA Sample Exams:
    - www.ni.com/claprep

CLA Exam Scenarios

The following table lists possible exam scenarios that you may receive to develop a solution for your CLA exam. This list is intended to give a general idea of what exams will be administered, and there may be variations within each exam.

Exam Scenario	**Description**
Coffee Machine	The coffee machine simulates ingredient storage, and performs grinding, brewing and dispensing operations to prepare hot water, coffee and latte.
Pizza Machine	The Pizza Machine simulates creating a customized pizza recipe and then making, baking and cutting the pizza.
Security System	The multi-zoned security system simulates the arming, disarming, tamper, bypass and alarm functions.
Thermostat	The thermostat simulates scheduled programmatic heating and cooling control for heating, ventilation and air-conditioning (HVAC) system.