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, and 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 demonstrates mastery in architecting LabVIEW applications for a multi-developer environment. 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.

Exam Overview

Product: Your test computer will have the LabVIEW Full Development System version 8.6.1 or LabVIEW 2009 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. Please note that you will not receive extra time before or after the exam to compensate for non-familiarity with the LabVIEW environment. Refer to `ni.com/labview/how_to_buy.htm` for details on the features available in the LabVIEW Full Development System.
Exam Duration: 4 hours
Style of exam: Short answer, VI, and application architecture development
Passing grade: 70%

The CLA exam validates LabVIEW software architecting skills and experience in a team-based environment. The exam involves project planning, software architecture, and VI development. The exam does not involve any hardware.

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 two sections that are based on a project that is very similar to the project that you worked on the CLD exam.

- Project planning document—40 points
- Application architecture development—60 points

A project planning document and detailed application specification will be provided. The specifications consist of general and technical requirements for the application.

The project planning document contains questions that will help you in the development of the architecture of the application. The short answers for the project planning document should be typed in a WordPad document.

The application architecture should contain all the major modules organized in a LabVIEW project. The main VI and the modules should contain the governing architecture, data structures, documentation, and specifications so that a team member can develop the block diagram.

Your exam submissions should be transferred to a disk and turned in to your proctor.

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

Exam Logistics

1. Complete the Agreement to the Terms and Conditions.
2. U.S. registrants: Fax the agreement and register for the exam by calling National Instruments at (888) 484-4436 or register online at ni.com/training.
3. International registrants: Contact your local office to resister and schedule the exam.
4. Pay the certification exam fee or local currency equivalent.

For general questions or comments, email: certification@ni.com.

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
6. Test Plan Design
7. Deployment

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

The exam topics encompass the knowledge and skills demonstrable of a Certified LabVIEW Architect and not just what will be on the exam, Thus, because the purpose of the exam is to qualify Certified LabVIEW Architects, any of the topics in this guide has the potential to be included on the CLA certification exam.

Exam Topics (Overview)

Topic	Subtopic
1. Project Requirements	a. Technical requirements b. Project test requirements c. Requirements tracking
2. Project Organization and Hierarchy	a. LabVIEW project hierarchy b. Disk hierarchy c. LabVIEW paths
3. Project Architecture and Design	a. Main VI architecture b. Module / subVI architecture c. Simulation architecture d. User interface design e. Advanced design methods f. Documentation
4. Team-Based Design, Development, and Standardization Practices	a. LabVIEW development practices b. Configuration management c. Project management
5. Reusable Tools / Component Design	a. LabVIEW technologies b. API design
6. Test Plan Design	a. Code review b. System tests
7. Deployment	a. Deployment planning b. Build specifications

CLA Topics 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, resolution, jitter, data overflow, daylight savings
5. Error handling requirements—Warning, errors, critical errors, shutdown sequence
6. Hardware or simulation requirements—Interface and operational requirements for field devices
7. Input / Output requirements—Console, file, printer, and network devices
8. Initialization, shutdown requirements—User interface and program behavior during startup, error conditions, and shutdown
9. Non-functional requirements—Accuracy, performance, security, and 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. Project test requirements:

(refer to section 6, *Test Plan Design* for more information)

1. Define code review items
2. Define system tests
3. Specify test methods—Manual / automated
4. Specify software tools

c. Requirements tracking:

1. Classify requirements by the following criteria:
    a) Customer specification
    b) Software design
    c) Risks involved
    d) Verifiable by tests
2. Methods or (utilization of) software tools to track requirements
    a) LabVIEW based tools—NI Requirements Gateway
    b) Third party tools—DOORS, Rational tools, and so on
    c) Proprietary tools—spreadsheets, databases, and so on

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 subVI / custom controls
         c) Plug-in VIs
         d) LabVIEW Project libraries
         e) Support files (documentation and so on)
      2. Specify a naming convention
   b. Disk hierarchy
      1. Mimic project hierarchy on disk
      2. Project hierarchy on network sever (source control)
   c. Paths
      1. VI search path configuration
      2. Use (relative) symbolic paths
      3. Utilization of relative and absolute path in code
3. **Project Architecture and Design**
   a. Main VI architecture
      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
      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. Module / subVI architecture
      1. Select a cohesive architecture and design for modules and subVIs
      2. Define and develop a clear API
      3. Define a consistent connector pane and icon
      4. Define error handling and ensure critical errors are handled appropriately
   c. Simulation module architecture
      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
   d. User interface design
      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

e. Advanced design methods
   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 IO techniques

f. Documentation
   1. Utilize the *LabVIEW Development Guidelines*
   2. Document the following architectures using the functional specification
      a) Main architecture for module integration
      b) Data structures and message passing 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. LabVIEW development practices
   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. Configuration management
   1. Utilize a source control system to manage and track project changes
   2. Develop team policies and processes for using the source control system
   3. Develop a project hierarchy and VI naming scheme to avoid cross-linking
   4. Utilize built-in LabVIEW tools for configuration management

c. Project management
   1. Define clear project goals and requirements
   2. Plan the project before diving into development
   3. Estimate resources and schedules using proven and standard methods
   4. Define tasks and timelines for team and individual developers
   5. Communicate and update project progress to the development team, stakeholders, and the customer

5. **Reusable Tools / Component Design**
   a. LabVIEW technologies
      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
      2. Develop tools to automate development, testing and project related tasks
   b. API design
      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
6. **Test Plan Design**
   a. Code review
      1. Define code review guidelines and strategy
      2. Define the items that are necessary for code review for each of the following:
         a) User interface style
         b) Block diagram development practice and code optimization
         c) Memory and resource optimization
         d) Error handling
         e) Timing mechanism
         f) Documentation
   b. System tests
      1. Define system test guidelines and strategy
      2. Define the following categories and tests that should be performed under these categories
         a) Performance test
         b) Regression test
         c) Usability test
         d) Configuration test
         e) Functional test
         f) Stress test
         g) Reliability and deployment test
      3. Determine the LabVIEW, third-party, or custom software that you could utilize to perform the test
      4. Develop an architecture and guidelines for development of customized test(s)

7. **Deployment**
   a. Deployment plan
      1. Identify the components required to be included in or with the deployed executable
      2. Identify and resolve issues that pertain to deploying the project as an executable
      3. Identify deployment target(s) and set the requirements for code and build specifications
   b. Build specifications
      1. Identify and set requirements for project build specifications
      2. Utilize the project API and source control API to enhance and automate builds

CLA Exam Format

The CLA exam has two sections that are based on a project similar to the project that you worked on for the CLD exam.

- Project Planning Document—40 points
- Application Architecture Development—60 points

In the CLA exam, you will be given a sketch of the user interface and the functional specifications. You must complete the project planning document and develop an architecture for the project, according to the following details.

Project Planning Document:
In this section of the exam, you must provide short answers to approximately ten questions that will help you to plan the architecture, development, and deployment of your project.

The responses to the project planning document should be short, concise, and typed in a WordPad document. The document should be saved to a disk along with the application architecture VIs and given to the exam proctor.

Note The project planning document includes the following main topics:

- Project requirements, assumptions, and constraints
- Project (files) organization considerations
- Architecture for the following:
  - Main VI, modules, and subVIs
  - Timing, error handling and shutdown, initialization, and file I/O
- Team-oriented project development
  - Configuration management issues and resolutions
  - Development (technical) challenges and resolutions
  - Utilization of LabVIEW features
- Tools and reusable components
  - Select and apply technologies and approaches to situations
  - Identify pros / cons of one method over others
- API design
- Software test plan development
  - Code review
    - Test coverage
    - Utilization of software tools
    - Possible issues and resolutions
  - System tests
    - Test coverage
    - Utilization of software tools
    - Possible issues and resolutions
    - Deployment planning
  - Required components
  - Technical issues and resolutions

Application Architecture Development:
For this section, 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 are for documenting the functional requirements 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 main (controller) VI. The main VI should *not* include any functional logic for the application, but should include the following:
   i. User interface
   ii. Driving architecture
   iii. Major data structures
   iv. Event, data, timing, and error communication method(s)
   v. Error handling
   vi. Documentation listing the functional details, which are sufficient for a developer to complete the functionality of the VI
   vii. Fully connected modules and /or subVIs

b. Develop shell (stub) modules and subVIs, which should *not* include any 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. Documentation listing the functional details, which are sufficient for a developer to complete the functionality of the VI

c. 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. Your design should *not* include any functional logic, but should include the following:
   i. Appropriate front panel objects
   ii. Architecture and API
   iii. Inputs, outputs, icon, and connector pane, depending on your design
   iv. Major data structures
   v. Error handling and error communication
   vi. Documentation listing the functional details, which should be sufficient for a developer to complete the functionality of the VI