OVERVIEW CARD

NI TestStand™ System and Architecture

NI TestStand is flexible test management software that offers the following major features:

- Out-of-the-box configuration and components provide a ready-to-run, full-featured test management environment.
- Numerous methods for modifying, configuring, and adding new components, which provide extensibility so you can create a test executive that meets specific requirements without altering the core TestStand Engine. You can upgrade to newer versions of TestStand without losing your customizations.
- Sophisticated sequencing, execution, and debugging capabilities, and a powerful sequence editor that is separate from the user interfaces.
- User interface controls for creating custom user interfaces and sequence editors.
- You can also create your own user interface in any programming language that can host ActiveX controls or control ActiveX automation servers.
- Example user interfaces with source code for National Instruments LabVIEW, National Instruments LabWindows™/CVI™, Microsoft Visual Basic .NET, C#, and C++ (MFC).
- An open language interface that provides support for many application development environments (ADEs). You can create code modules in a variety of ADEs and call pre-existing modules or executables.
- A comprehensive application programming interface for building multithreaded test systems and other sophisticated test applications.
- Integration with third-party source code control packages.
- Deployment tools to aid in transferring a test system from development to production.

373457B-01 Apr07

NATIONAL INSTRUMENTS

Architecture Overview

TestStand Sequence Editor
TestStand development environment for creating, modifying, executing, and debugging sequences.

Custom User Interfaces
Customizable applications that, depending on mode, edit, execute, and debug test sequences on a test station. User interfaces are available in several different programming languages and include full source code, which allows you to modify them to meet specific needs.

Process Models
Define the operations that occur for all test sequences, such as identifying the UUT, notifying the operator of pass/fail status, generating a test report, and logging results. TestStand includes three fully customizable process models: Sequential, Parallel, and Batch.

User Interface Controls
A powerful set of ActiveX controls and support APIs for creating custom user interfaces.

TestStand Engine
A set of DLLs that provides an extensive ActiveX Automation API for controlling and interacting with TestStand. The TestStand Sequence Editor, User Interface Controls, and user interfaces use this API.

Sequence File Executions
Created by the TestStand Engine when you execute a test sequence using the sequence editor or a user interface.

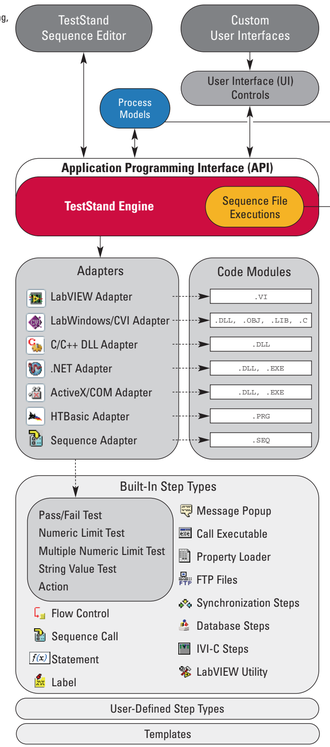
Adapters
Allow TestStand to call code modules in a variety of different formats and languages. Adapters also allow TestStand to integrate with various ADEs to streamline test code generation and debugging.

Code Modules
Program modules, such as LabVIEW VIs (.vi) or Windows Dynamic Link Libraries (.dll), that contain one or more functions that perform a specific test or action. TestStand adapters call code modules.

Built-In Step Types
Define the standard behaviors for common testing operations. Some step types use adapters to call code modules that return data to TestStand for further analysis. Other step types perform standard operations, such as calling an executable or displaying dialog boxes.

User-Defined Step Types
Define a set of custom step properties and default behaviors for each step of that custom type. You can also define data types.

Templates
Create custom sequences, steps, and variables to use as templates to build sequence files.



Sequence File Execution Flow

Sequence File Executions
You can execute a sequence directly, or you can execute a sequence file through a process model Execution entry point, such as Test UUTs and Single Pass.

Process Model Sequence File Execution
When you start an execution through a process model Execution entry point, the process model defines how to test the UUTs. The Sequential model tests one UUT at a time. The Parallel model tests multiple independent test sockets at the same time. The Batch model tests a batch of UUTs using dependent test sockets.

Process Model Result Processing
The TestStand Engine collects the results of each step that executes into a result list. Process models use the result list to generate reports and log data to databases.

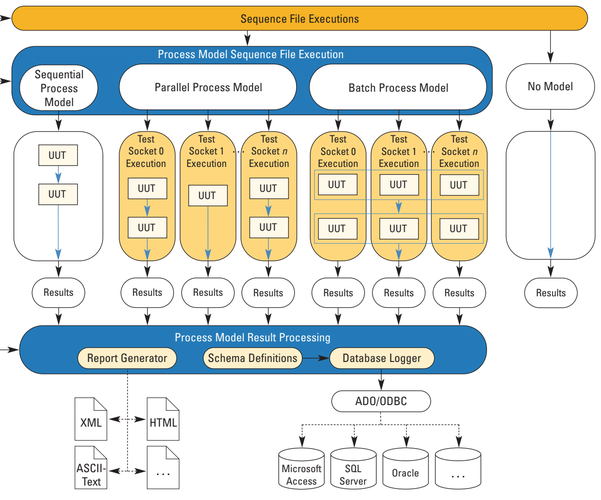
Unit Under Test (UUT)
Device or component that you are testing.

Test Socket Execution
For each test socket, or fixture, in the system, the Parallel and Batch models launch a separate test socket execution that controls the testing of UUTs in that test socket.

Report Generator
The report generator traverses test results to create reports in XML, HTML, and ASCII-text formats. You can fully customize the reports.

Schema Definitions
Schema definitions define SQL statements, table definitions, and TestStand expressions that define how to log results to a database. You can fully customize the schemas.

Database Logger
The database logger traverses test results and exports data into database tables using schema definitions.



Execution Object Structure

Execution Object
Contains information TestStand needs to run a sequence, its steps, and any subsequences it calls. You can suspend, interactively debug, resume, terminate, or abort executions.

Thread Object
Represents an independent path of control flow.

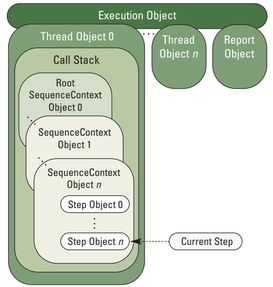
Report Object
Contains the report text. The process model updates the Report object, and the sequence editor or user interface displays it.

Call Stack
Lists the chain of active sequences waiting for nested subsequences to complete. The first item in the call stack is the most-nested sequence invocation.

Root SequenceContext Object
Represents the execution of the least-nested sequence invocation that contains a list of steps and calls to other sequences.

SequenceContext Object
Represents the execution of a sequence that another sequence called.

Current Step
Represents the executing step of the currently executing sequence in the call stack.



Sequence File Structure

Sequence File
Contains any number of sequences, a set of data types and step types the sequence file uses, and any global variables that sequences in the sequence file share.

Sequences
Contain groups of steps, local variables, and parameters used for passing data between steps and subsequences.

Types
Sequence files contain definitions of all data types and step types that its sequences use. Variables and properties in a sequence are instances of data types. Steps in a sequence are instances of step types.

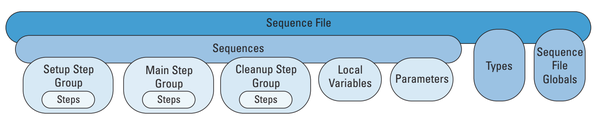
Sequence File Globals
Store data you want to access from any sequence or step within the sequence file in which you define the sequence file global variable.

Setup, Main, Cleanup Step Groups
TestStand executes the steps in the Setup step group first, the Main step group next, and the Cleanup step group last. By default, a sequence moves to the Cleanup step group when a step generates an error in the Setup or Main step group.

Local Variables
Store data relevant to the execution of the sequence. You can access local variables from within steps and code modules defined in a sequence.

Parameters
Use parameters to exchange data between calling and called sequences.

Steps
Perform built-in operations or call code modules. A step is an instance of a step type, which defines a set of step properties and default behaviors for each step.

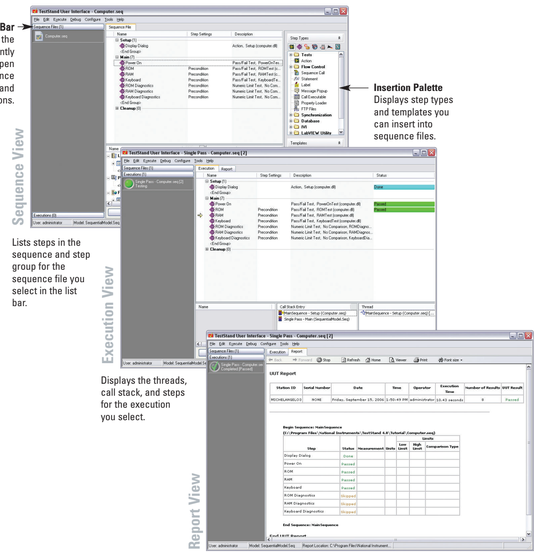


OVERVIEW CARD

NI TestStand™ System and Architecture

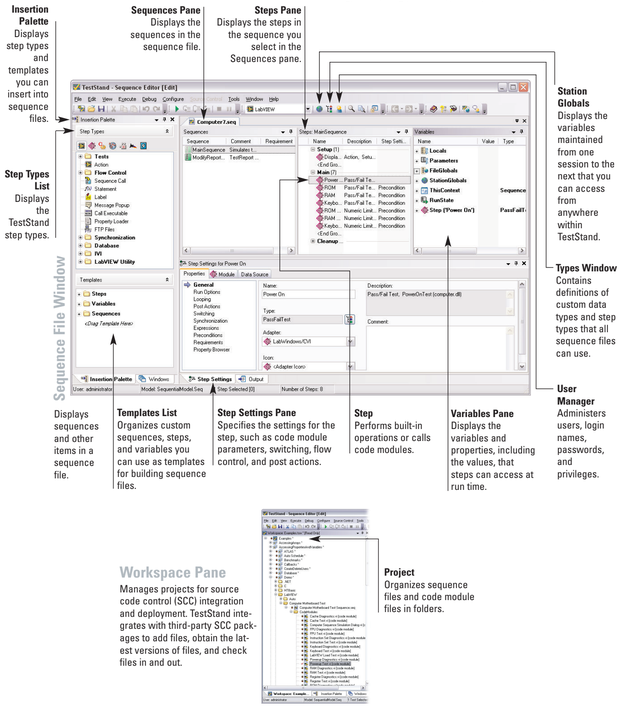
User Interface Overview

TestStand includes separate user interface applications developed in LabVIEW, LabWindows/CVI, Microsoft Visual Basic .NET, C#, and C++ (MFC). Because TestStand includes the source code for each user interface, you can fully customize the user interfaces. You can also create your own user interface using any programming language that can host ActiveX controls or control ActiveX automation servers. With the user interfaces in operator mode, you can start multiple concurrent executions, set breakpoints, and single-step. In editor mode, you can modify sequences, display sequence variables, sequence parameters, step properties, and so on.



TestStand Sequence Editor Overview

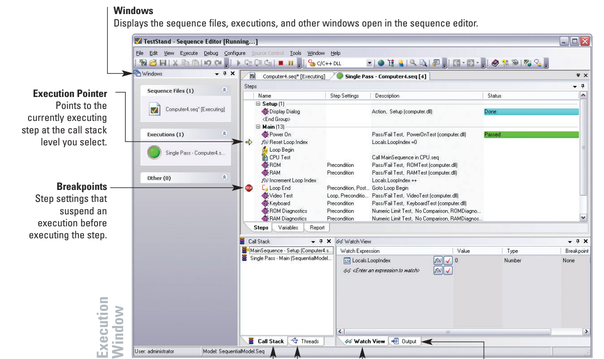
You can use the fully customizable TestStand development environment to create, modify, execute, and debug sequences. You can also use the sequence editor to modify step types and process models. You can customize the environment by docking, auto-hiding, and floating panes to optimize your development tasks. The sequence editor provides familiar LabVIEW, LabWindows/CVI, and Microsoft Visual Studio .NET debugging tools, including breakpoints, single-stepping, stepping into or over function calls, tracing, a Variables pane, and a Watch View pane. In the TestStand Sequence Editor, you can start multiple concurrent executions, execute multiple instances of the same sequence, and execute different sequences at the same time. Separate Execution windows display each execution. In trace mode, the Execution window displays the steps in the currently executing sequence. When you suspend an execution, the Execution window displays the next step to execute and provides single-stepping options.



Workspace Pane

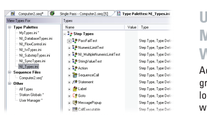
Manages projects for source code control (SCC) integration and deployment. TestStand integrates with third-party SCC packages to add files, obtain the latest versions of files, and check files in and out.

Project
Organizes sequence files and code module files in folders.



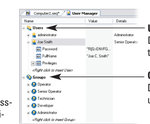
Types Window

Contains definitions of custom data types and step types that all sequence files can use.



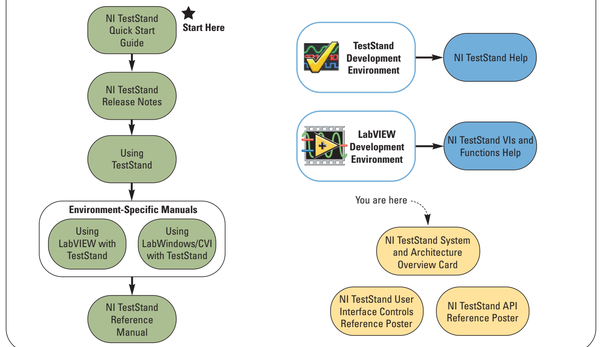
User Manager Window

Administers groups, users, login names, passwords, and privileges.



Documentation Map & Overviews

The *Guide to TestStand Documentation* topic in the *NI TestStand Help* contains links to all the TestStand documentation in electronic format. Access the *NI TestStand Help* by selecting **Start»All Programs»National Instruments»TestStand 4.0»Online Help»NI TestStand Help** or by selecting **Help»NI TestStand Help** in the TestStand Sequence Editor.



Printed Documentation

NI TestStand Quick Start Guide
Use this document for system requirements and installation instructions. This document also contains information about the different TestStand licensing options.

NI TestStand Release Notes
Use this document to learn about new features and upgrade information.

Using TestStand
Use this manual to familiarize yourself with the TestStand environment and the basic features you use to build and run test sequences.

Using LabVIEW with TestStand
Use this manual in conjunction with the *Using TestStand* manual to learn how to use LabVIEW with TestStand.

Using LabWindows/CVI with TestStand
Use this manual in conjunction with the *Using TestStand* manual to learn how to use LabWindows/CVI with TestStand.

NI TestStand Reference Manual
Use this manual to learn about TestStand concepts, architecture, and features.

Online Help

NI TestStand Help
Use this help file to learn more about the TestStand environment and the TestStand User Interface Controls and Engine APIs. The *NI TestStand Help* also includes basic information about using an ActiveX automation server.

NI TestStand VIs and Functions Help
Use this help file to learn more about TestStand-specific VIs and functions. This help file is accessible only from LabVIEW.

Cards and Posters

NI TestStand User Interface Controls Reference Poster
Use this poster to learn about the controls available for writing custom user interfaces for TestStand.

NI TestStand API Reference Poster
Use this poster as an overview of the TestStand API. This poster lists the properties, objects, methods, and API inheritance of the TestStand API.