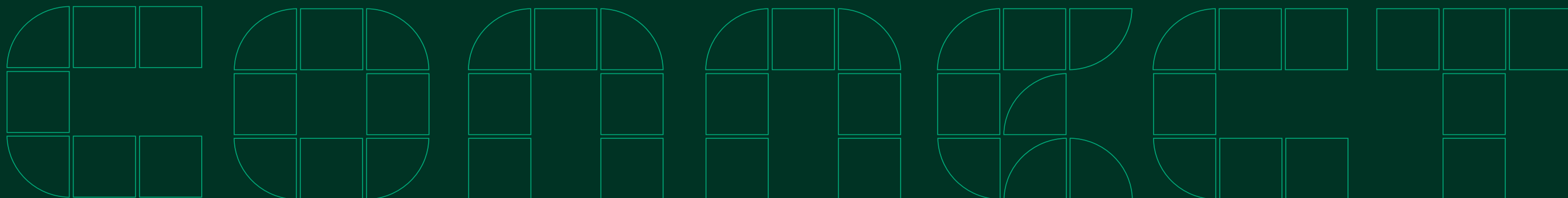


connect



What's new in TestStand?

Harsha Bhushan
Principal Product Manager

TestStand Product Management Team



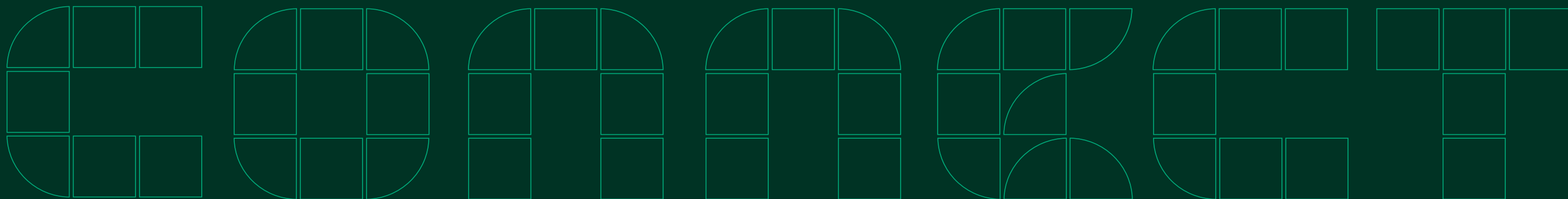
Harsha Bhushan
TestStand Product Manager



Shauna Rae
Electronics Test Product Manager

Agenda

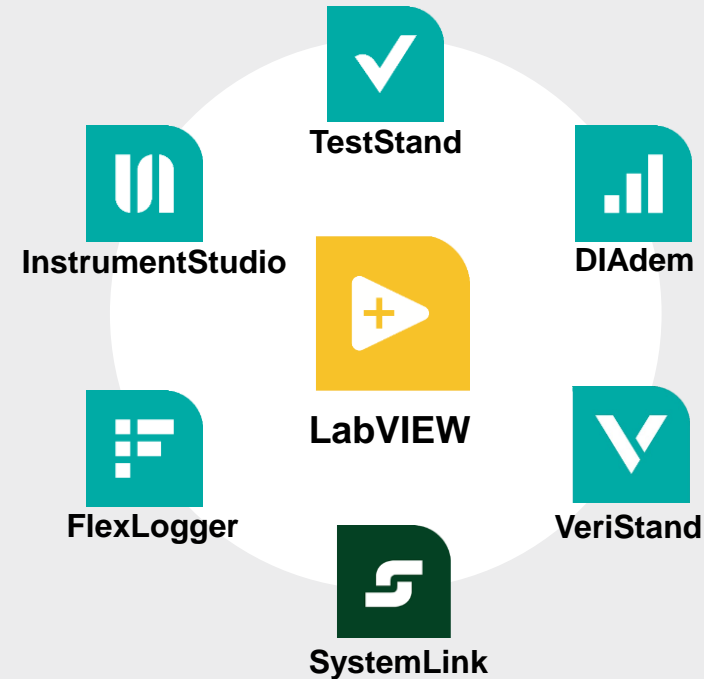
- NI software strategy
- What is TestStand?
- New features in TestStand
 - 2023 Q4
 - 2022 Q4
- TestStand Roadmap
 - 2024 Q4
 - Long term 2025+



NI software strategy

Evolving NI Test Software

Enable Automated Test & Measurement Professionals



1

Strengthen LabVIEW

Deliver new capabilities in **LabVIEW & NI Software** to meet the evolving requirements of applications and users

2

Connect LabVIEW+

Bridge seamlessly between tools, tasks and teams to accelerate the productivity of test professionals

3

Build Community

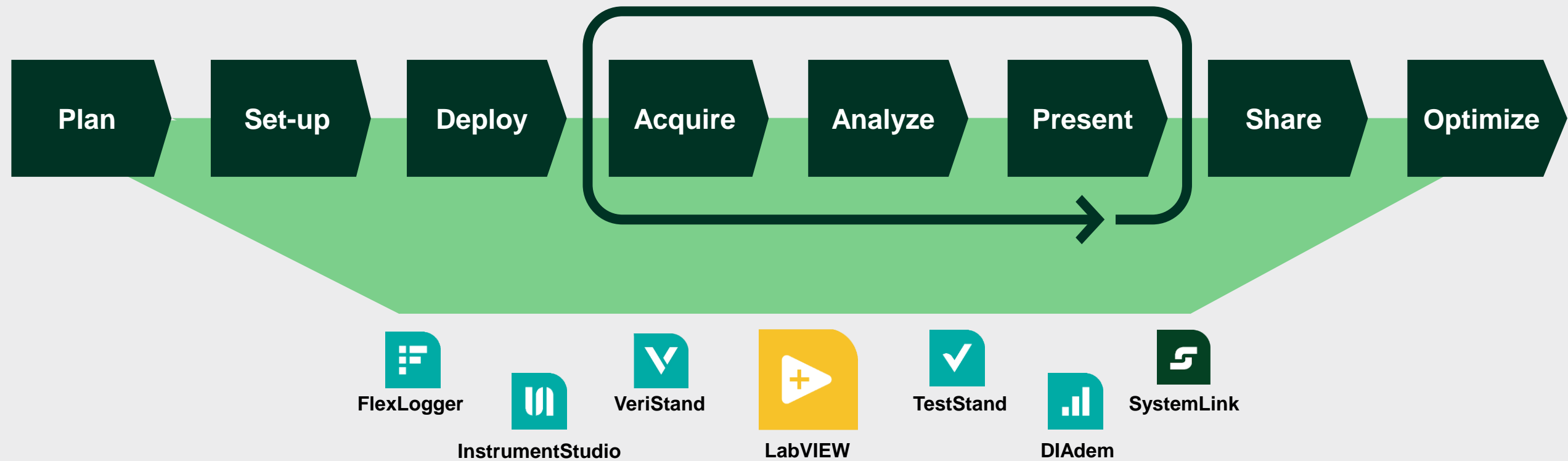
Engage and collaborate with the community to empower their continued success

Connect **LabVIEW+**

Bridge seamlessly between tools, tasks, and teams to accelerate the productivity of test professionals

LabVIEW is a graphical programming environment that accelerates test and measurement application development.

LabVIEW+ brings together a comprehensive & connected suite of software, including LabVIEW, and enables high-level development across test workflows.



Software for Professional Test Workflows

Electronics Validation Test

Characterizing electronic prototypes to ensure quality and performance

Set-up & Configure

Measure & Automate

Analyze & Share



Electronics Production Test

Functional test ensuring manufactured products meet specifications

Set-up & Configure

Measure & Automate

Deploy & Maintain



Electromechanical Validation Test

Characterizing physical prototypes to ensure quality and performance

Build & Customize

Configure

Analyze & Share



Embedded Software Test

Testing deployed software for defects across wide parameter variations

Configure & Map

Test & Bring Up

Automate & Execute





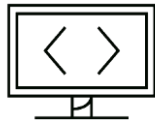
What is TestStand?

A test executive software that accelerates system development and deployment for engineers in validation and production.



Automate Your System

Create, execute, and debug test sequences using an interactive environment



Leverage Existing Code

Use code from LabVIEW, Python, C/C++, or .NET



Test More, Faster

Use native parallel execution to reduce test time and functionality for advanced tasks such as sweeping and looping



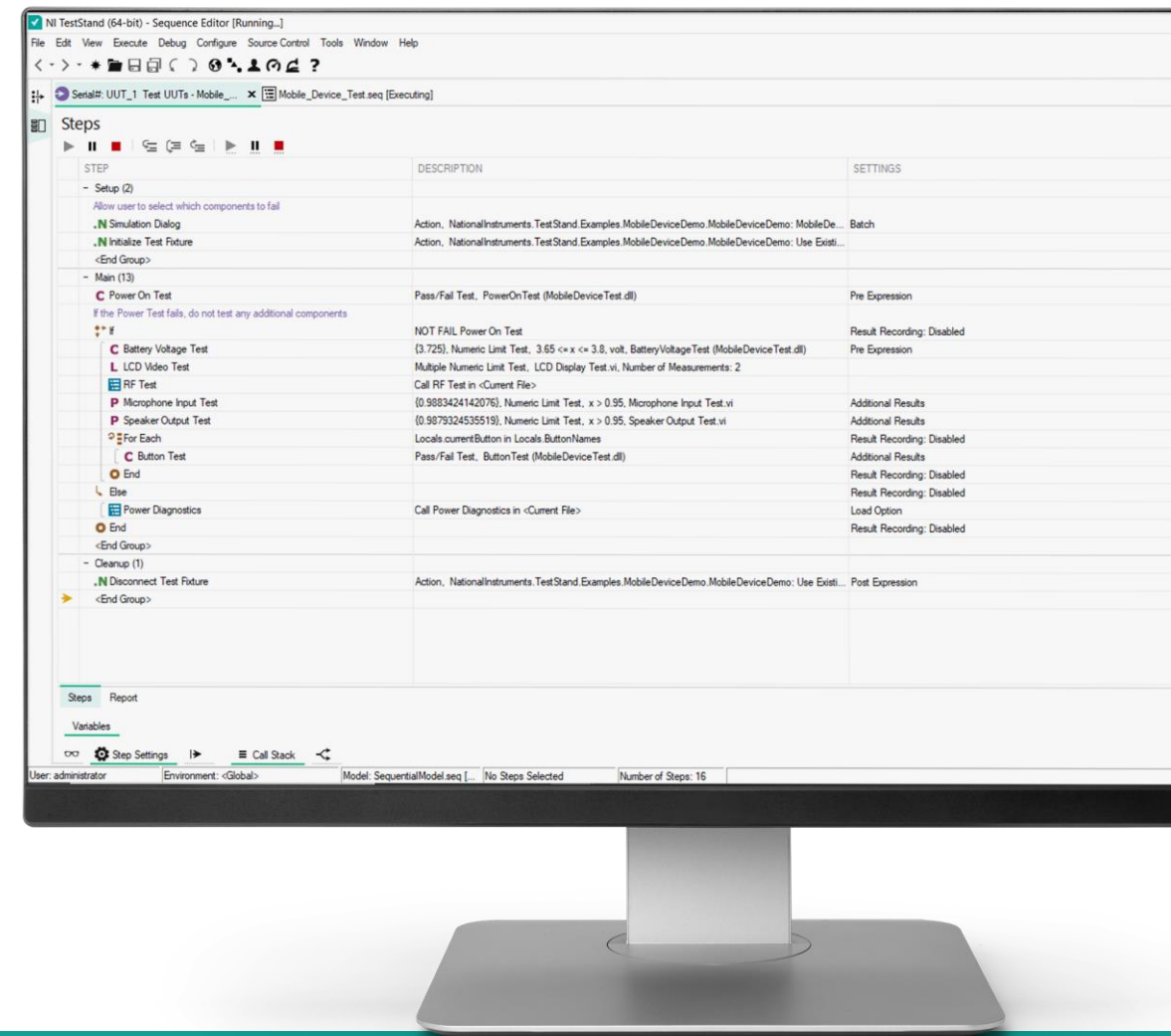
Keep Track of Results

Track units, automatically generate reports, and store results to local or network databases



Deploy To Your Testers

Scale your operations by deploying sequences to numerous test stations with custom or pre-built operator interfaces



Connect TestStand



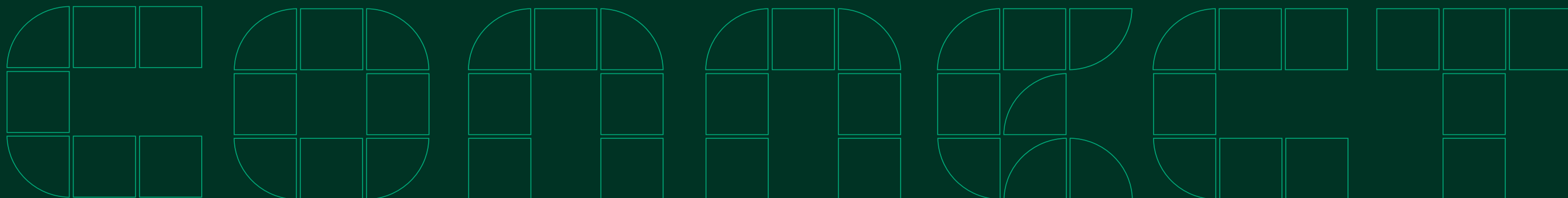
With **LabVIEW** to quickly develop complex reusable tests



With **Flexlogger** to efficiently acquire and log data from your DAQ hardware



With **InstrumentStudio** for interactive instrument control



New features in TestStand

2023 Q4, 2022 Q4

Specification Compliance Manager (SCM)

Specifications View

READ-ONLY

Spec Source: Excel

UPDATE SPECS

ATmega328P Characteristics

Parametric

Show/Hide Columns

| Spec Details | | | | Spec Conditions | | | Spec Limits | | | |
|-------------------|-------|-------------|---------------------|-----------------|------------|------------|-------------|---------|-----|------|
| Spec ID | Block | Spec Symbol | Spec Name | Vs (V) | Freq (MHz) | Temp (°C) | Min | Typical | Max | Unit |
| OperatingVolta... | - | Vcc | OperatingVoltage | - | - | [-40..125] | 2.7 | - | 5.5 | V |
| ADCInputVolta... | - | Avcc | ADCInputVoltage | [2.7..5.5] | - | [-40..125] | 2.4 | - | 5.8 | V |
| ActiveSupplyC... | - | Icc | ActiveSupplyCurrent | [5] | [8] | [-40..125] | - | 5.2 | 10 | mA |
| IdleSupplyCurr... | - | Icc | IdleSupplyCurrent | [5] | [16] | [-40..125] | - | 1.9 | 2.8 | mA |
| WatchdogFreq | - | Fwdt | WatchdogFreq | [2.7..5.5] | - | [-40..125] | 76 | 128 | 180 | KHz |
| AnalogCompOf... | - | Vacio | AnalogCompOffset... | [2.7..5] | - | [-40..125] | - | 10 | 40 | mV |

Import data from SCM to TestStand

NI TestStand (64-bit) - Sequence Editor [Edit]

File Edit View Execute Debug Configure Source Control Tools Window Help

< > * [Icons]

Sequence File 3

Steps: MainSequence

▶

▶

| STEP | DESCRIPTION | SETTINGS |
|---------------------|-------------|----------|
| + Setup (0) | | |
| - Main (0) | | |
| <Insert Steps Here> | | |
| + Cleanup (0) | | |

Variables Sequences

Variables

Filter by name

| NAME | VALUE | TYPE | COMMENT |
|---------------------------------|-------|---------------------------|---------|
| Locals ('MainSequence') | | | |
| ResultList | | Array of Result[0..empty] | |
| Parameters ('MainSequence') | | | |
| FileGlobals ('Sequence File 3') | | | |
| StationGlobals | | | |
| ThisContext | | Sequence Context | |
| RunState | | | |

Environment: <Global>

Model: SequentialModel.seq

No Steps Selected

Number of Steps: 0

Export results from TestStand to SCM

NI TestStand (64-bit) - Sequence Editor [Edit]

File Edit View Execute Debug Configure Source Control Tools Window Help

All Products > ATmega328p

Admin

← Hide Categories

✓ Compliance View

🕒 21 May 2024, 09:56 ↻

Search Categories 🔍

ATmega328P Character...
Parametric 100%

ATmega328P Characteristics ⓘ

Parametric

🔍 9 hidden columns

↻ Refresh

| | | | | Spec Details | | VA Validation | | | | | | | |
|--------------------------|--|--|--|-------------------|-------------------|---------------|-------------------|--------|---------|---------|---------|---------|--------------------|
| | | | | Spec ID | Spec Name | Health | Coverage | Cpk | Min | Max | Mean | Median | Standard Deviation |
| <input type="checkbox"/> | | | | OperatingVolta... | OperatingVolta... | ⚠ | <div>100.0%</div> | 2.161 | 2.9 | 6 | 4.115 | 4.1 | 0.306 |
| <input type="checkbox"/> | | | | ADCInputVolta... | ADCInputVolta... | ✓ | <div>100.0%</div> | 39.051 | 3.479 | 3.523 | 3.5 | 3.5 | 0.009 |
| <input type="checkbox"/> | | | | ActiveSupplyC... | ActiveSupplyC... | ⚠ | <div>100.0%</div> | 0.582 | 0.036 | 9.929 | 4.442 | 4.166 | 2.781 |
| <input type="checkbox"/> | | | | IdleSupplyCurr... | IdleSupplyCurr... | ⚠ | <div>100.0%</div> | 0.301 | 0.016 | 3.463 | 1.496 | 1.465 | 0.886 |
| <input type="checkbox"/> | | | | WatchdogFreq | WatchdogFreq | ✓ | <div>100.0%</div> | 13.233 | 114.612 | 119.317 | 117.021 | 117.057 | 1.019 |
| <input type="checkbox"/> | | | | AnalogCompO... | AnalogCompO... | ✓ | <div>100.0%</div> | 1.708 | -1.676 | 23.955 | 11.079 | 11.152 | 5.566 |
| <input type="checkbox"/> | | | | AnalogCompL... | AnalogCompL... | ✓ | <div>100.0%</div> | 1.937 | -17.025 | 17.249 | 2.194 | 3.018 | 8.128 |

⚙️ ESTIMATED TEST VECTORS: 48 TOTAL

| | 🔍 | PARAMETER NAME | TYPE | 🔗 | PARAMETER VALUE/SESSION | [COUNT], {RANGE} | PARAMETER: | Temp | VALUES: |
|---|---|----------------|-------|---|----------------------------|----------------------|----------------|-----------------------------|---------|
| ✓ | 0 | Temp | 123 ▼ | 🔗 | FileGlobals.SCM_Specifi... | ⚙️ ✓ [6] {-40 - 110} | Strategy | Start, Stop, Step | -40 |
| ✓ | 1 | Voltage | 123 ▼ | 🔗 | FileGlobals.SCM_Specifi... | ⚙️ ✓ [8] {2.7 - 5.5} | Mode | Dynamic | -10 |
| | | | | | | | Distribution | Linear | 20 |
| | | | | | | | Numeric Format | <Default> | 50 |
| | | | | | | | Start | FileGlobals.SCM_Specific... | 80 |

Environment: <Global> Model: SequentialModel.seq 1 Step Selected [0] Number of Steps: 3

Hot reloading

- Make changes to your source code while debugging
- Continue to execute your test sequence without rebuilding/restarting test execution
- C#, C++ hot reloading with TestStand & Visual Studio 2022
- Python hot reloading with Visual Studio Code

NI TestStand (64-bit) - Sequence Editor [Edit]

File Edit View Execute Debug Configure Source Control Tools Window Help

Computer Motherboard Test Sequence.... x

Steps: MainSequence

| STEP | DESCRIPTION | SETTINGS |
|-------------------------|---|--------------|
| .N ROM Test | Pass/Fail Test, NationalInstruments.TestStand.Ex... | |
| .N RAM Test | Pass/Fail Test, NationalInstruments.TestStand.Ex... | |
| .N Video Test | Numeric Limit Test, 0 < x < 10, hertz, NationalInstr... | |
| .N Keyboard Test | Numeric Limit Test, x > 5, NationalInstruments.Tes... | |
| If | ThisContext.SequenceFailed | |
| CPU Diagnostics | Call CPU Diagnostics in <Current File> | Precondition |
| .N ROM Diagnostics | Pass/Fail Test, NationalInstruments.TestStand.Ex... | Precondition |
| .N RAM Diagnostics | Pass/Fail Test, NationalInstruments.TestStand.Ex... | Precondition |
| .N Video Diagnostics | Numeric Limit Test, x < 0, NationalInstruments.Tes... | Precondition |
| .N Keyboard Diagnostics | Pass/Fail Test, NationalInstruments.TestStand.Ex... | Precondition |
| End | | |
| Else | | |
| .N Powerup Diagnostics | Numeric Limit Test, No Comparison, NationalInstru... | |
| End | | |

Step Settings

There are no steps selected.

Environment: <Global> Model: SequentialModel.seq No Steps Selected Number of Steps: 20

Filtering variables

- Filter variables and their properties by name

| Variables Sequences | | | |
|-------------------------------------|------------------------------|---------------------------|---------|
| Variables | | | |
| 🔍 | | | |
| NAME | VALUE | TYPE | COMMENT |
| Locals ('MainSequence') | | | |
| ResultList | | Array of Result[0..empty] | |
| SeqFileDirectory | "" | String | |
| CSVOutputStream | Nothing | Object Reference | |
| 123 Vin | 0 | Number | |
| 123 Vout | 0 | Number | |
| 123 lin | 0 | Number | |
| 123 lout | 0 | Number | |
| 123 Efficiency | 0 | Number | |
| dev2 | | NI_IOSession (Container) | |
| <Right click to insert Local> | | | |
| Parameters ('MainSequence') | | | |
| <Right click to insert Parameter> | | | |
| FileGlobals ('Efficiency.seq') | | | |
| <Right click to insert File Global> | | | |
| StationGlobals | | | |
| ThisContext | | Sequence Context | |
| RunState | | | |
| Engine | IEngine | Object Reference | |
| Root | Nothing, SequenceContext,... | Object Reference | |
| Main | Nothing, SequenceContext,... | Object Reference | |

Expression editor improvements

- Delimiter highlighting (Parenthesis matching)
- Fixed width font Lucida sans support for better readability

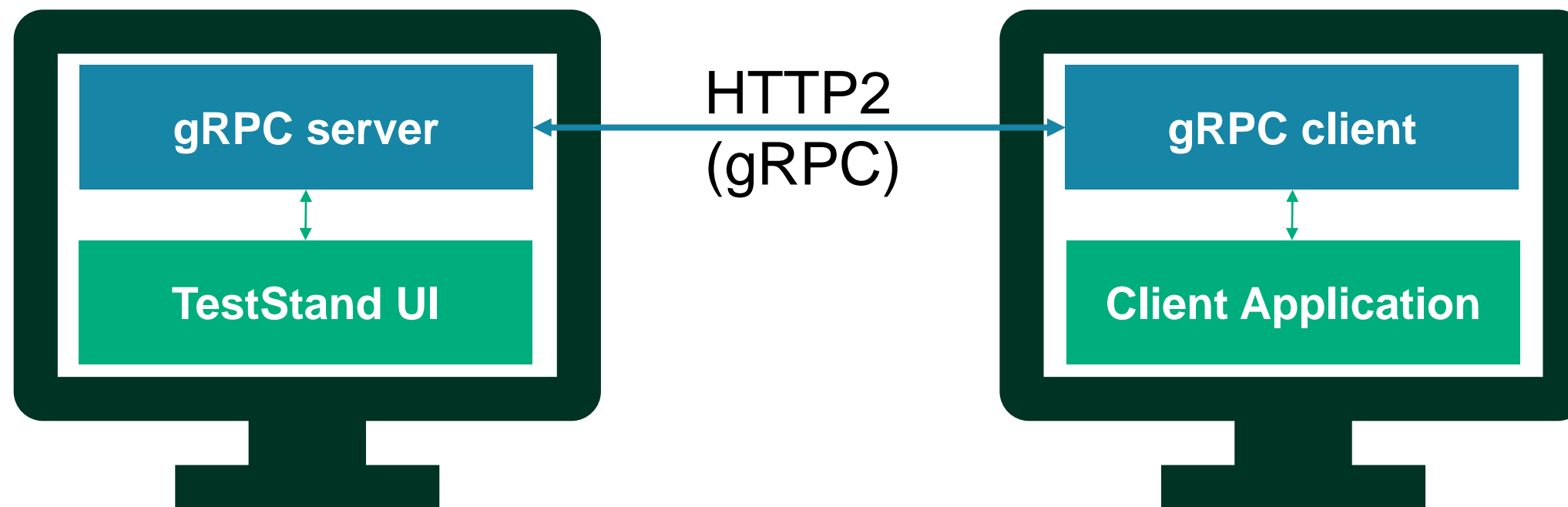


gRPC API for TestStand

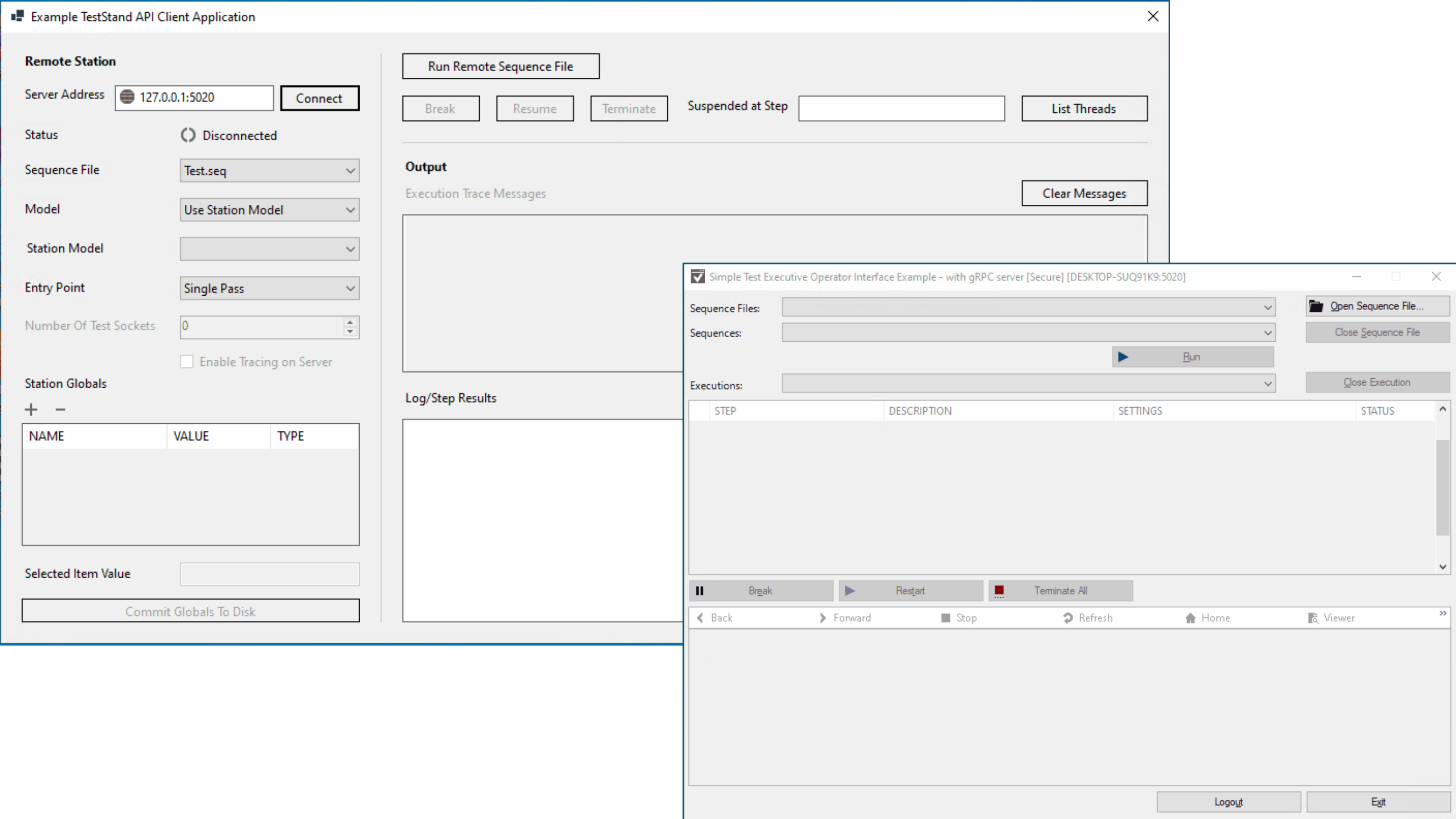
Early access on github



<https://github.com/ni/grpc-teststand-api>



Remotely control & monitor TestStand UI
Any Operating System
Any programming language



Remote Station

Server Address

Status ⏸ Disconnected

Sequence File

Model

Station Model

Entry Point

Number Of Test Sockets

☐ Enable Tracing on Server

Station Globals

+

-

| NAME | VALUE | TYPE |
|------|-------|------|
|------|-------|------|

Selected Item Value

Suspended at Step

Output

Execution Trace Messages

Log/Step Results

Simple Test Executive Operator Interface Example - with gRPC server [Secure] [DESKTOP-SUQ91K9:5020]

Sequence Files:

Sequences:

| STEP | DESCRIPTION | SETTINGS | STATUS |
|------|-------------|----------|--------|
|------|-------------|----------|--------|

Back

Forward

Stop

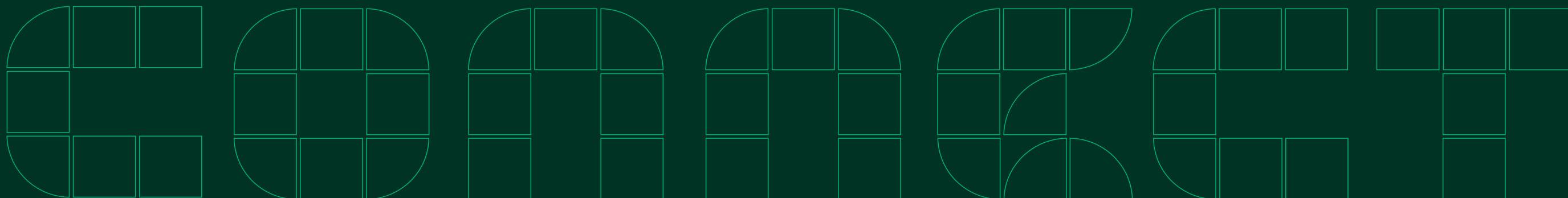
Refresh

Home

Viewer

Logout

Exit



TestStand Roadmap

2024 & beyond



Short-term product focus

Provide better interoperability with modern and performant programming languages and frameworks

Improve usability and efficiency for engineers to quickly develop scalable and maintainable test systems

Long-term product focus

Enable test deployment and development on modern, secure and cross-platform environment

Improve TestStand on-boarding experience

| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

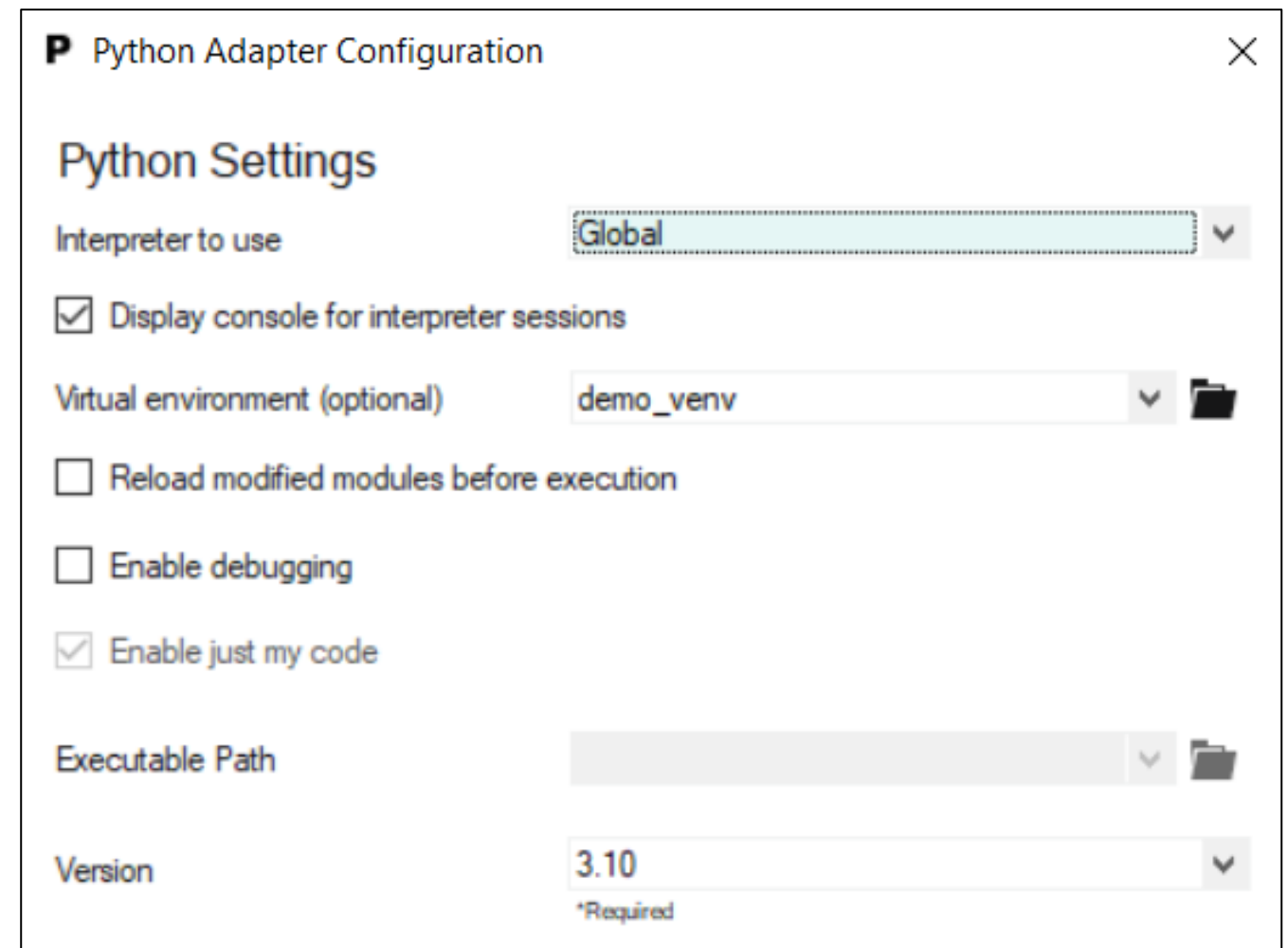
| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

Python virtual environment using venv



Virtual environments help isolate packages & dependencies of python projects

- Venv is a built-in python package to create Virtual environment
- Call a python code in a virtual environment from TestStand

A screenshot of the 'Python Adapter Configuration' dialog box. The title bar says 'P Python Adapter Configuration'. The main section is 'Python Settings'. It contains several settings: 'Interpreter to use' is a dropdown menu with 'Global' selected; 'Display console for interpreter sessions' is a checked checkbox; 'Virtual environment (optional)' is a dropdown menu with 'demo_venv' selected, accompanied by a folder icon; 'Reload modified modules before execution' is an unchecked checkbox; 'Enable debugging' is an unchecked checkbox; 'Enable just my code' is a checked checkbox; 'Executable Path' is a text field with a folder icon; and 'Version' is a dropdown menu with '3.10' selected, with a '*Required' label below it. A close button (X) is in the top right corner.

P Python Adapter Configuration

Python Settings

Interpreter to use: Global

☒ Display console for interpreter sessions

Virtual environment (optional): demo_venv

☐ Reload modified modules before execution

☐ Enable debugging

☒ Enable just my code

Executable Path:

Version: 3.10
*Required

| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

.NET core code module support

- We acknowledge the need for a text-based language more performant than Python
- .NET core:
 - Performant & cross-platform
 - Supports newer versions of C#
 - Has hot-reloading capabilities while debugging in Visual Studio
 - Availability of many 3rd party IP to leverage
- .NET core support in TestStand
 - .NET adapter in TestStand would now support .NET 8
 - Call and debug .NET 8 assemblies from teststand
 - Many .NET framework assemblies can be called too*
 - A step towards Linux support



| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

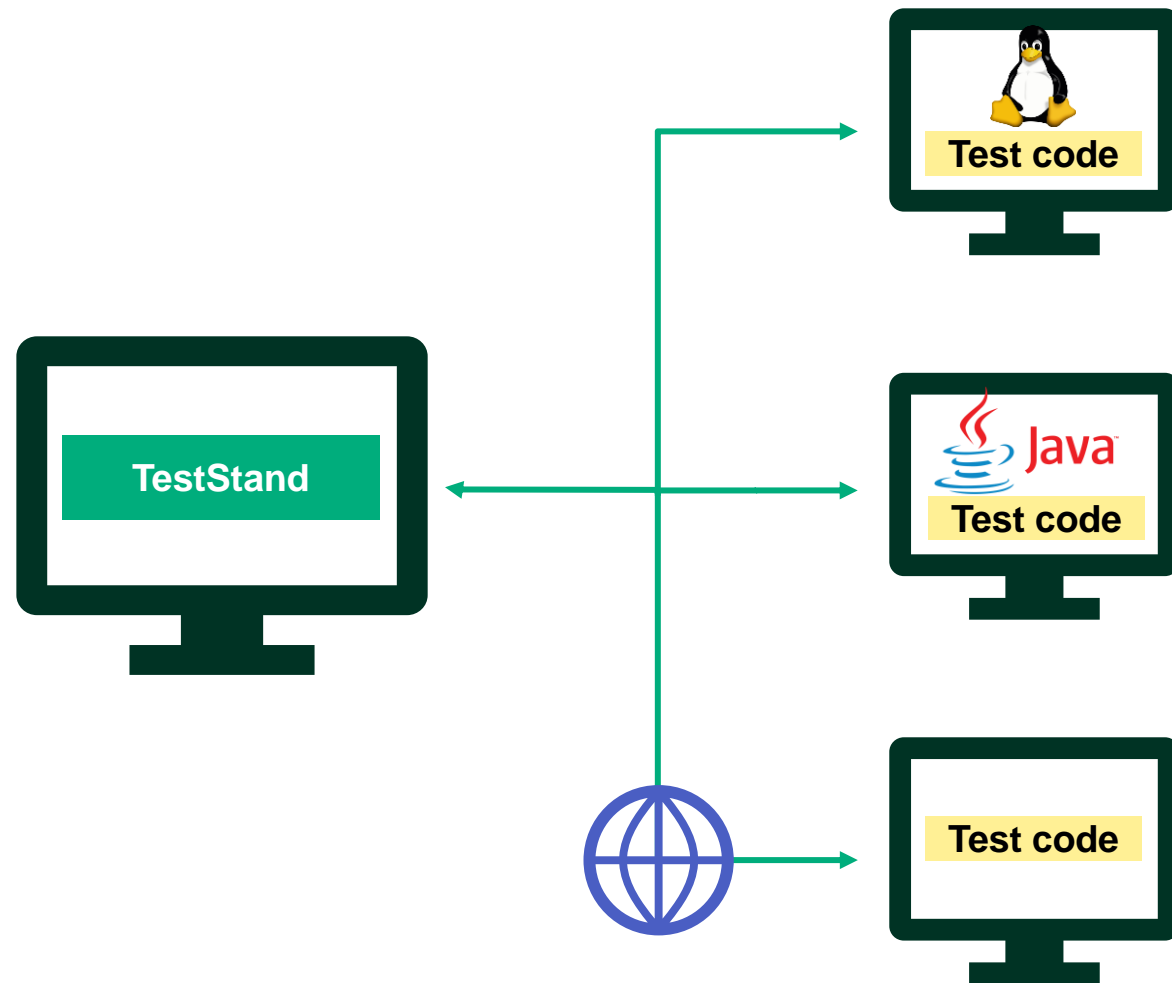
Python Anaconda distribution

- Has built-in data science & ML packages
- Execute TestStand Python step with Anaconda distribution
- Switch between CPython & Anaconda distributions in TestStand during execution



| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

Remote Procedural Call steps



- Call test code from any OS
- Call test code written in any programming language*
- Call test code present on a machine anywhere on earth
- No/low code
- Powered by gRPC

| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

Native Python API for TestStand

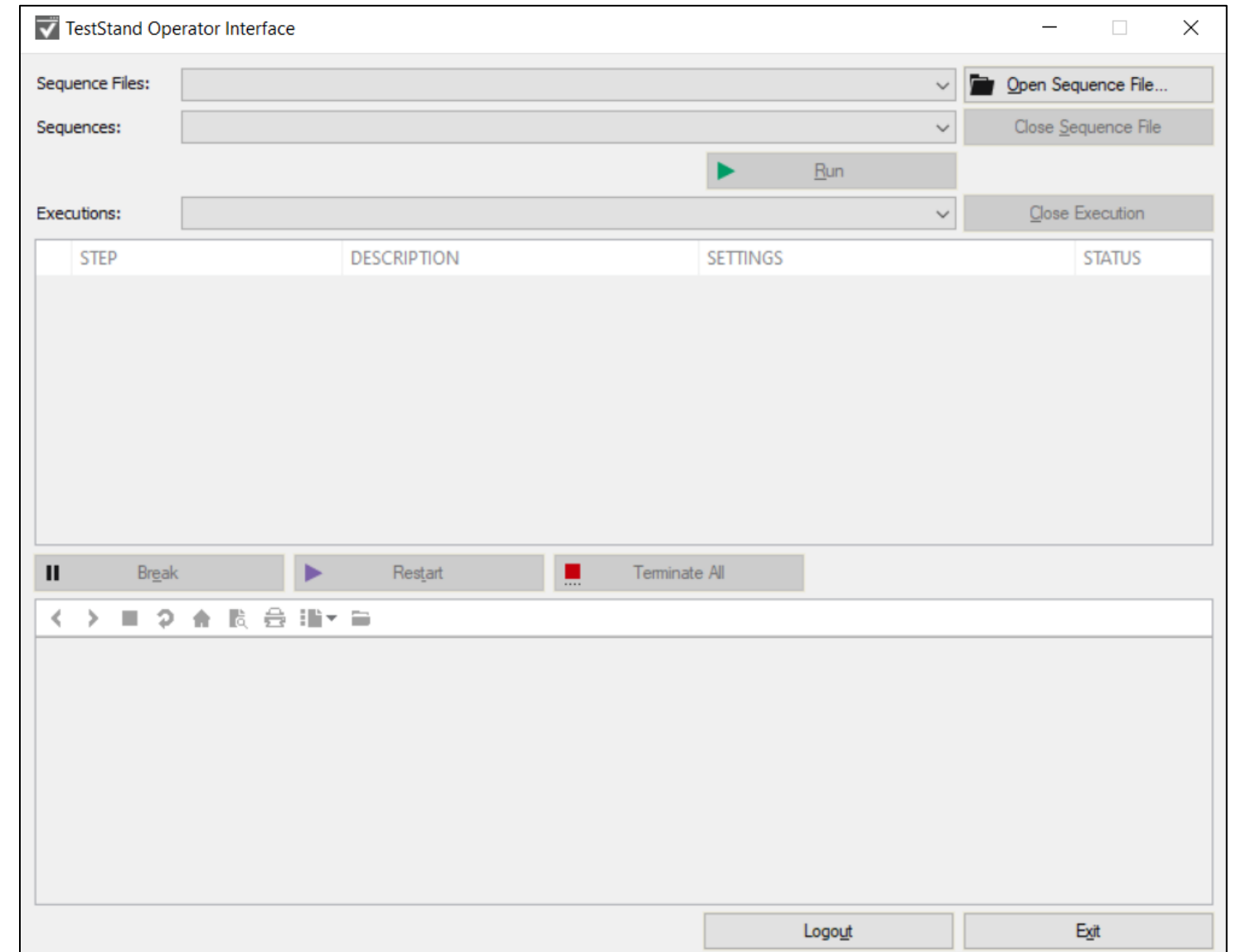
- Python has 28% of market share
- Easy to program
- Want to control TestStand through native Python API?



| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

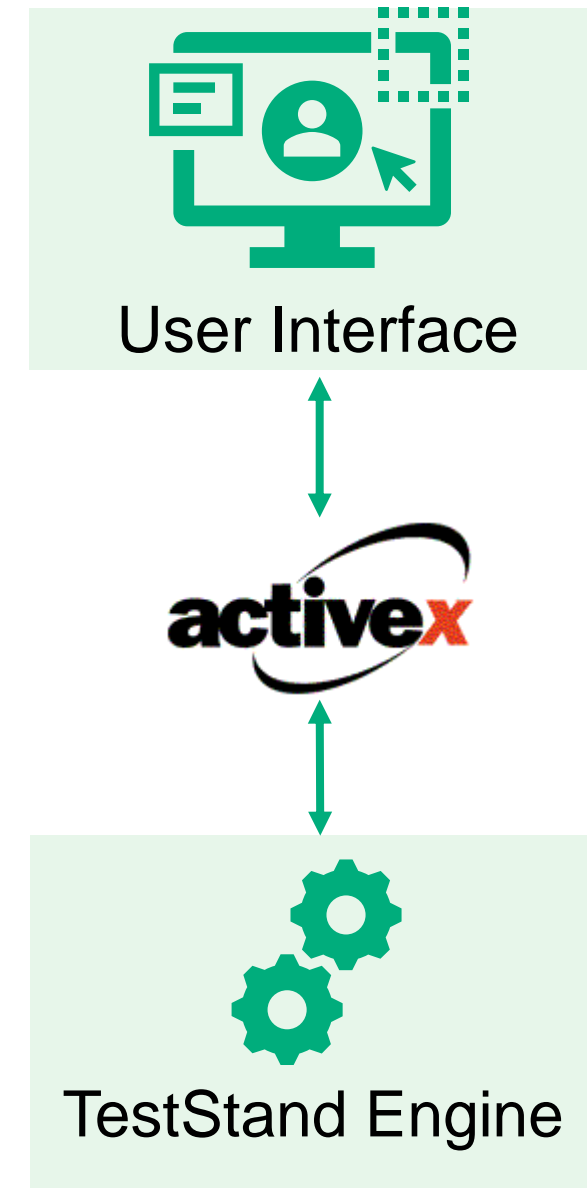
Operator Interface/ User Interface

- Simple user interfaces for operators on the production floor
- Can be customized according to needs



Operator Interface/ User Interface

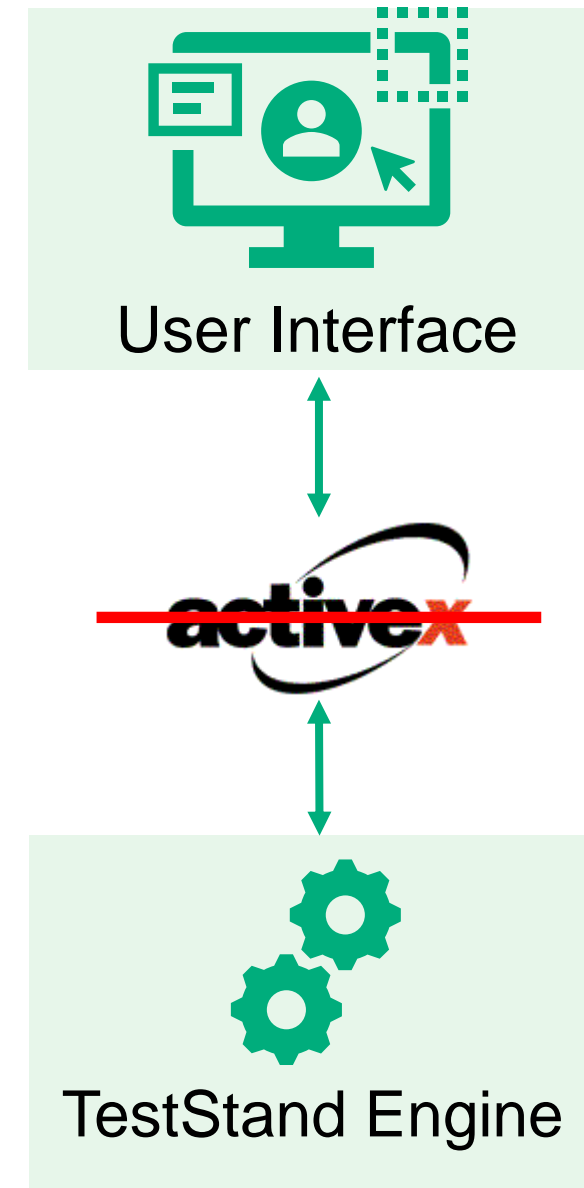
- Simple user interfaces for operators on the production floor
- Can be customized according to needs



Modern Operator Interface

Graphical interfaces to execute tests at a production site

- Secure UI controls
- Cross platform
- Create interfaces in C#, LabVIEW, web
- Better user management




Web based
controls

| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

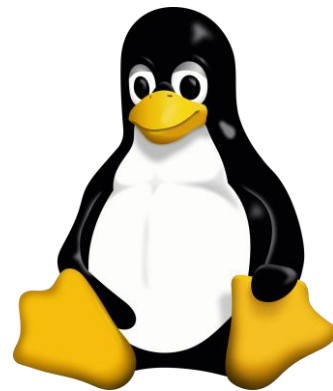
Deploying test sequence on Linux Desktop



No frequent
updates



Secure



Linux Desktop



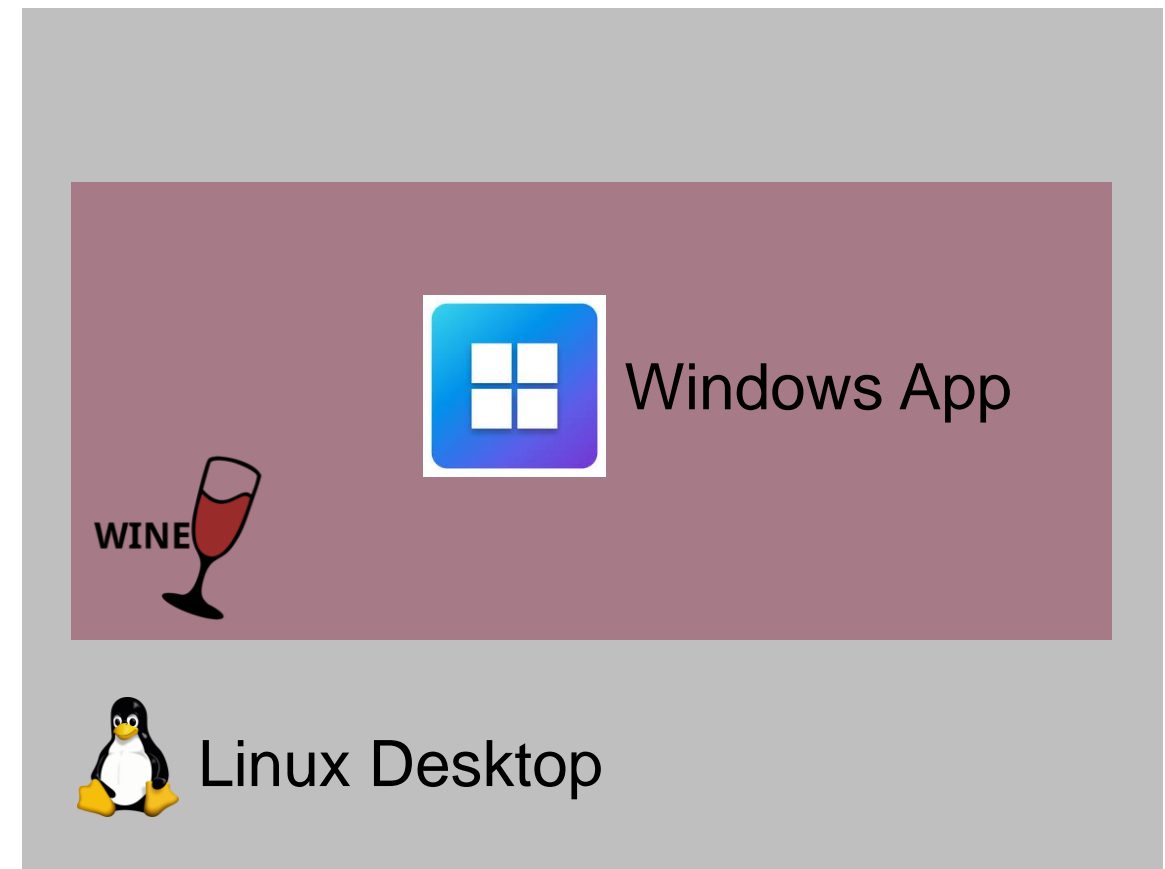
Performant



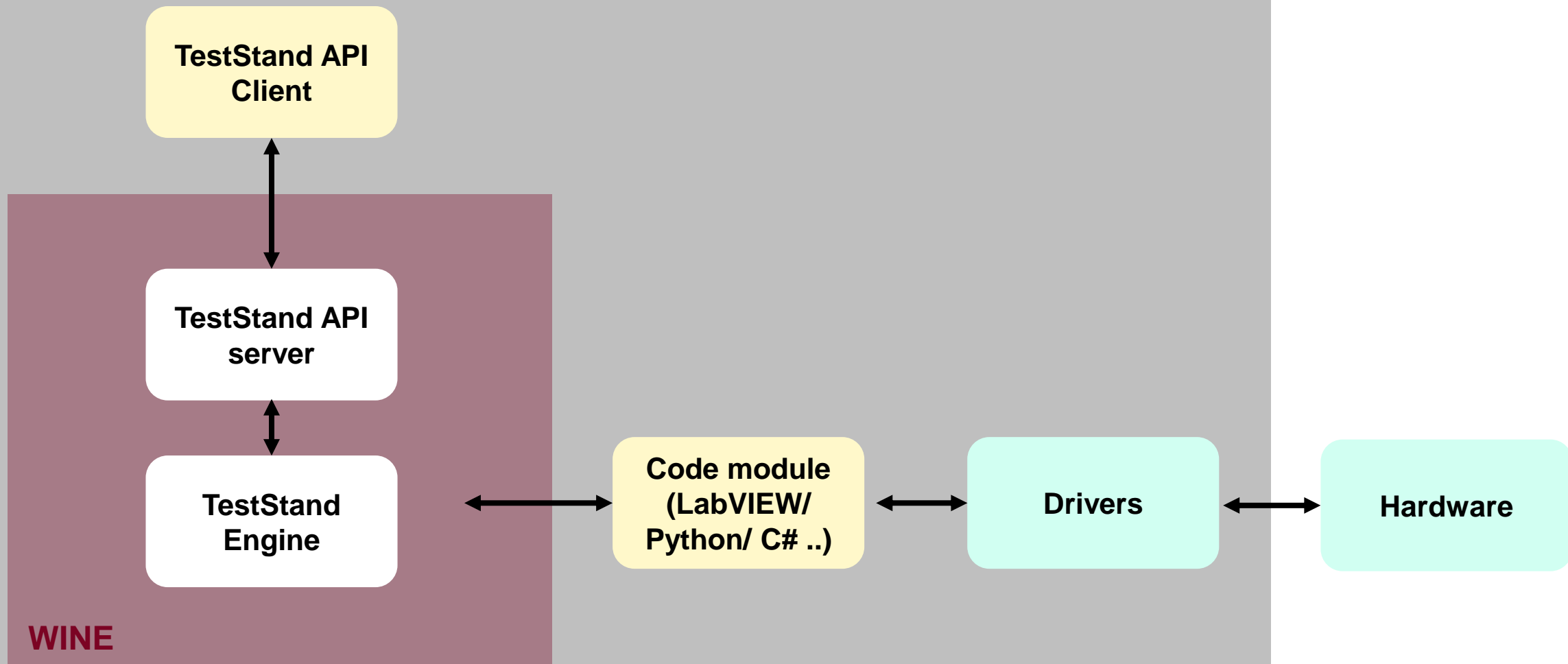
Open source

Deploying test sequence on Linux Desktop

- WINE (Wine Is Not an Emulator)
- Run Windows application on Linux
- Free & open source
- Performant
- Light weight



Deploying test sequence on Linux Desktop



Linux Desktop

| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Easily create test sequence variants for Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

Source control using Git in TestStand

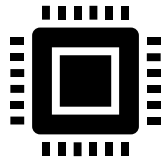
Git is a popular source control system.

- Source control of TestStand sequence files
- Diffing sequence files
- Merge changes to sequence files

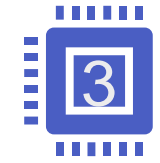
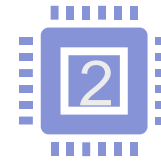
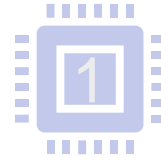


| Capability | Shipped | 2024 | 2025+ |
|--|------------------------|------|-------|
| Interoperability | | | |
| Support Python virtual environments | | ✓ | |
| Support for calling .NET Core assemblies (.NET 8) | | ✓ | ✓ |
| Support Python Anaconda distribution | | | ✓ |
| Remote Procedure Call steps | | | ✓ |
| Native Python API for TestStand | | | ✓ |
| Modern, secure & cross-platform environment | | | |
| gRPC API for remote control & execution of test sequence | Early Access on GitHub | | |
| Modern Operator Interface | | ✓ | ✓ |
| Deploying Test sequences on Linux Desktop | | | ✓ |
| Development of Test sequences on Linux Desktop | | | ✓ |
| Improve test development efficiency | | | |
| Hot reloading of C# & C++ modules in Visual Studio 2022 | 2023 | | |
| Integration with SystemLink Specification Compliance Manager | 2023 | | |
| Filter variables & their properties | 2023 | | |
| Integration with Git source code management | | | ✓ |
| Develop scalable tests for variants of Device Under Test (DUT) | | | ✓ |
| Performance | | | |
| Remove dependency on LabVIEW ADE version for source VIs and support better build times for steps using source-only VIs | 2023 | | |
| Improved performance of Python enumerations | 2023 | | |

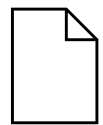
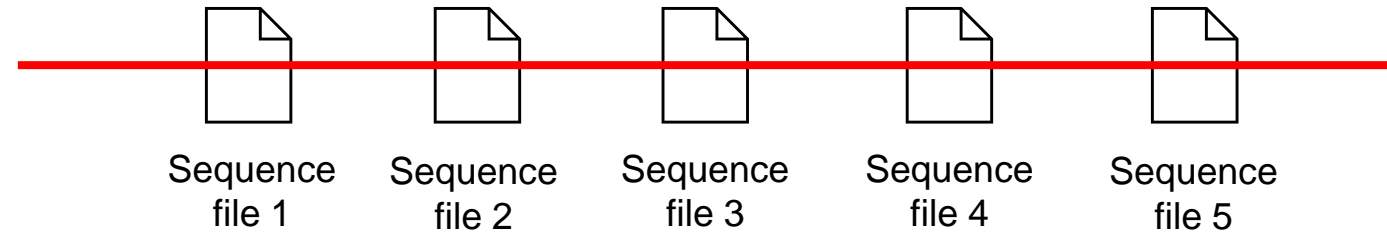
Develop scalable tests for variants of devices



Device Under Test (DUT)



Variants of DUTs



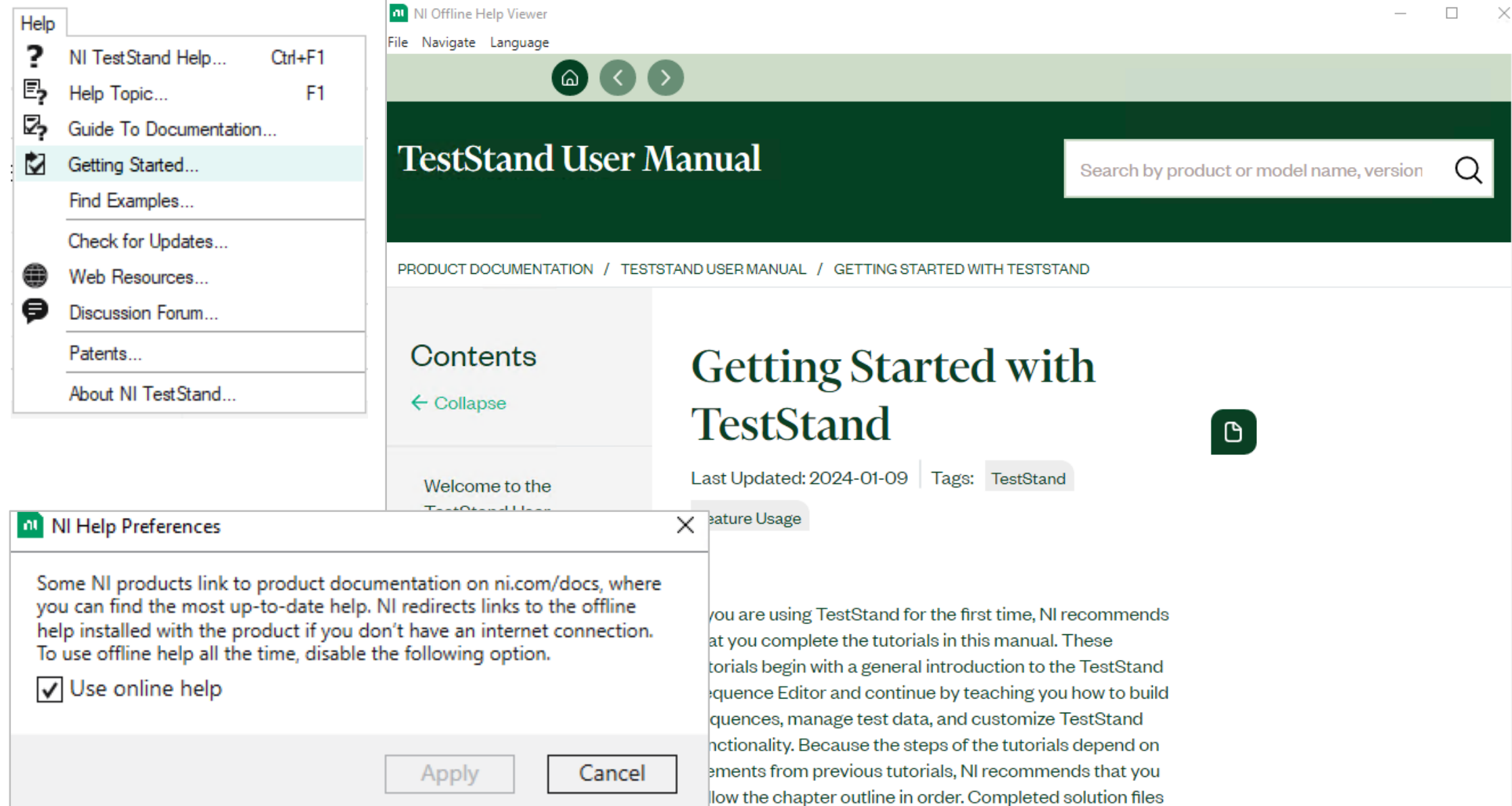
Master
Sequence
file

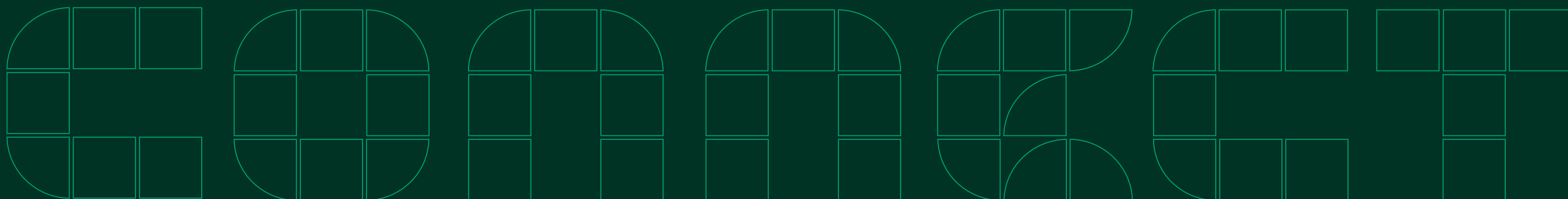


- Single sequence file
- Run only variant specific test cases
- Use variant specific configuration (limits, parameters etc)

Modernization of TestStand Help

- View TestStand Help
 - online when the system is connected to the internet
 - Offline, when the system is not connected to the internet
- Option to always view help offline





Q & A