STRIDE™ Test System for *On-Target White-box Testing*

STRIDE™ is a **test infrastructure** used by engineers to implement and execute tests for validating embedded software executing On-Target. STRIDE has been designed specifically for **On-Target White-box Testing**.

The STRIDE system consists of:

A software *Framework*

enabling the creation of fully automated ***Test Assets*** used to verify the internal design of your software

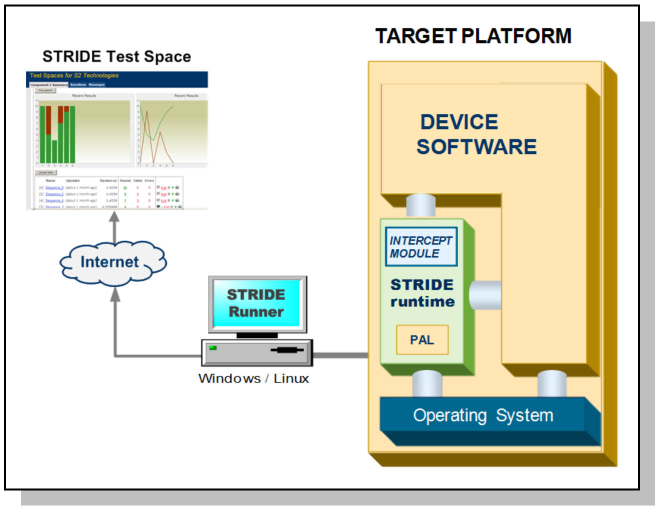
A hosted *Web Application* - *Test Space*

for storing and analyzing test results, thus providing on-demand availability of timely, accurate, and meaningful test results data

STRIDE Framework

Runtime

The STRIDE Framework includes a **Runtime** software source package that supports connectivity with the host system. It is integrated with embedded application software to enable *testability* to be compiled into the software with minimal impact on performance or the size of the application. The software is agnostic to the RTOS, CPU, and transport. It can be configured to support multi-processes, multi-threads, user/kernel spaces, or a simpler single application process. Typically the runtime is configured as a background thread and only executes when running tests. It provides **services for testing** and **source instrumentation**.

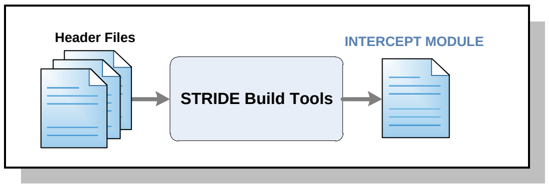


Runner

The ***Framework*** also contains a lightweight **Runner** application that is a host-based command-line utility for interactive and automated test execution. It also provides services for result publishing to our **hosted web application**.

Build Tools

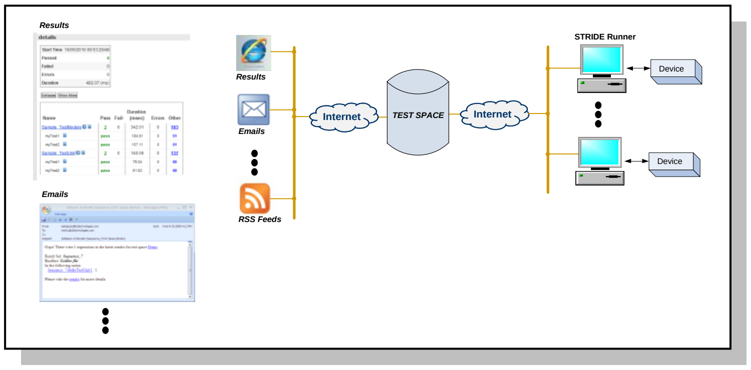
The **STRIDE Build Tools** are a set of command-line utilities that integrate with your target software build process. They preprocess standard C/C++ header files that contain STRIDE test declarations to auto-generate source code comprising a custom **Intercept Module**. The intercept module code is then built with the Runtime to provide *fixturing* and *harnessing* test logic as part of your build image.



STRIDE Test Space

Once tests have been implemented, they are executed using the host-based **Runner** application. Resulting **test reports** can be automatically uploaded to our **web application** - **STRIDE Test Space**. This application allows both local and distributed team members to track and collaborate on results.

Failure resolution is optimized by centralizing results and providing specific source information related to failures. *Test Space* also has built-in **collaboration features** in the form of email notification for new results and regression against a baseline as well as messaging.



STRIDE Test Features

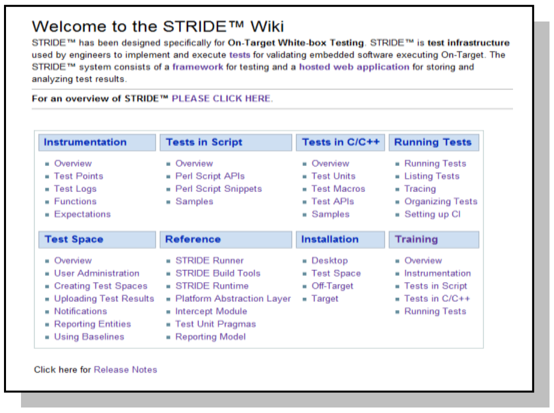
STRIDE supports three general **types of testing**:

- Unit Testing,
- API Testing, and
- Integration Testing.

STRIDE has been designed to deliver a broad spectrum of **testing capabilities** that enable teams to ***test earlier and more effectively***. Unique features such as **assertion macros**, **fixturing**, and **function doubling** facilitate deeper test coverage of your software components. STRIDE also provides **Expectations** as an added validation technique. Expectation testing is unique in that it *does not* focus on calling functions and validating return values. The technique rather validates the expected sequencing of the software, along with state data, executing under normal operating conditions.

Tests can be implemented in both **native code** on the target and **script** on the host.

For more detailed information please check-out the stride wiki @ www.stridewiki.com.



Welcome to the STRIDE™ Wiki

STRIDE™ has been designed specifically for **On-Target White-box Testing**. STRIDE™ is **test infrastructure** used by engineers to implement and execute tests for validating embedded software executing On-Target. The STRIDE™ system consists of a **framework** for testing and a **hosted web application** for storing and analyzing test results.

For an overview of STRIDE™ PLEASE CLICK HERE.

Instrumentation	Tests in Script	Tests in C/C++	Running Tests
Overview	Overview	Overview	Running Tests
Test Points	Perl Script APIs	Test Units	Listing Tests
Test Logs	Perl Script Snippets	Test Macros	Tracing
Functions	Samples	Test APIs	Organizing Tests
Expectations		Samples	Setting up CI
Test Space	**Reference**	**Installation**	**Training**
Overview	STRIDE Runner	Desktop	Overview
User Administration	STRIDE Build Tools	Test Space	Instrumentation
Creating Test Spaces	STRIDE Runtime	Off-Target	Tests in Script
Uploading Test Results	Platform Abstraction Layer	Target	Tests in C/C++
Notifications	Intercept Module		Running Tests
Reporting Entities	Test Unit Pragmas		
Using Baselines	Reporting Model		

Click here for Release Notes

UNIQUE FEATURES OF STRIDE	
Works on virtually any target platform	
Runtime written in standard C	• Delivered as source code to be included in build system • Uses a platform abstraction layer (PAL) for portability • Supports multi-threading and multi-process environment
Integrates with your existing Build System	• Harnessing code generated for tests written in C/C++ • Remoting code generated for tests written in script
Supports Off-Target Testing	• Standard SDKs for Windows and Linux hosts • Used for self-training / sample execution • Isolated Unit testing
Provides built-in automation and reporting	
Testable Builds	• Leverages existing software build process • Built-in test automation controllable from host • Timing analysis on test execution • Automatic test report generation
Functional Builds	• Software works exactly the same as before • Test logic is separated and not executed unless invoked via the **runner** • Nominal impact on performance • Single *STRIDE_ENABLED* compiler feature flag
Offers testing techniques for deeper coverage	
Test Macros	• Shortcuts for testing assertions & automatic report annotation on failures
Fixturing	• Test code executing on Target can perform file operations on the host • Remote files services include opening, reading, writing, etc
Expectations	• Validation technique using source instrumentation - **Test Point** • Test Points span threads, processes, and CPU boundaries • Simple table definitions used to define expected Test Points • Optional hooks *(routines)* can be setup for custom data validation
Doubling	• Intercept C/C++ global functions on target to stub, fake, or mock

Supports test implementation in C/C++ and Script

Tests in C/C++	• Any combination of C/C++ • Harnessing auto-generated • Focus on calling APIs, validating C++ classes, isolating modules, critical processing, etc.
Tests in Script	• Perl scripting on host supported • Minimal software build dependencies • Callable target based functions used for setup • Focus on code execution sequencing along with state data (data flow, state machines, ..)

Includes Web-based test results management

Centralized Hosting	• Test results automatically aggregated and published • Results organized into projects and spaces to enable tracking • Detailed reporting, trend charting, etc
Team Collaboration	• Automatic email notification on regression with file name & line number • Commenting & messaging support

For more detailed information please check-out the stride wiki @ www.stridewiki.com. We suggest you review the What is Unique About STRIDE article.