Astemes LUnit

Astemes - Anton Sundqvist

Anton Sundqvist

Table of contents

1. LUnit Basics

This document walks through the basic workflow using LUnit to test LabVIEW code. If you prefer to watch a video, there is an introduction available on this link.

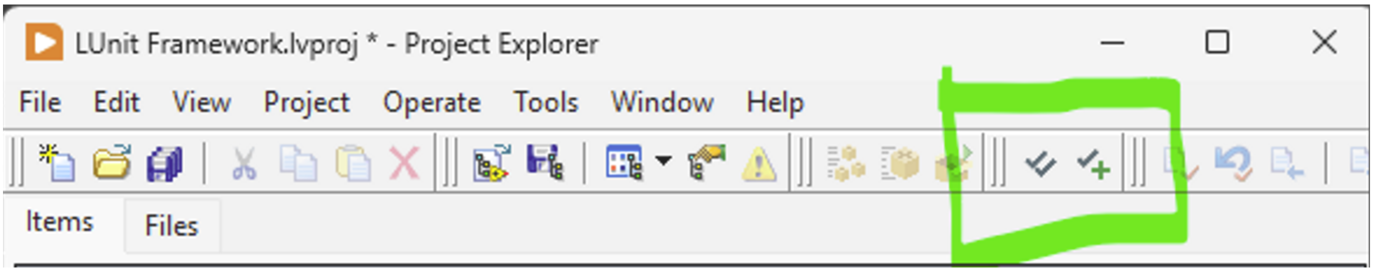
1.1 Prerequisites

To follow along with the instructions on this page you will need to have LabVIEW version 2020 or later installed as well as the LUnit unit testing framework.

1.2 Creating a Test Case

All tests you write will belong to a test case. The test case is implemented as a LabVIEW class, but in order to use it you will not need to know anything about object oriented programming.

To get started, create an empty project and add a test case to it by clicking the New Test Case button in the toolbar of the LabVIEW project.



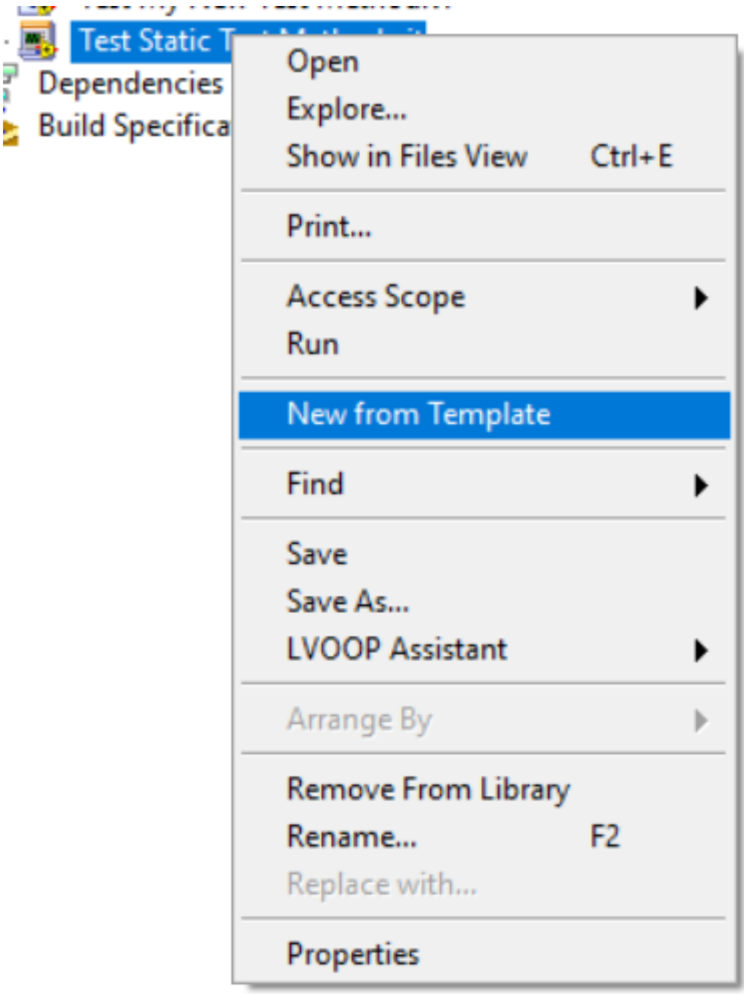
You can also do the same from the `Tools > LUnit > New Test Case...` menu option.

Save the test case in a convenient location. Some like to keep the tests next to the code they are testing, and other keep them in a separate folder called `Tests` or similar. I personally find the later option with a separate top level directory the most convenient.

1.3 Adding a Test Method

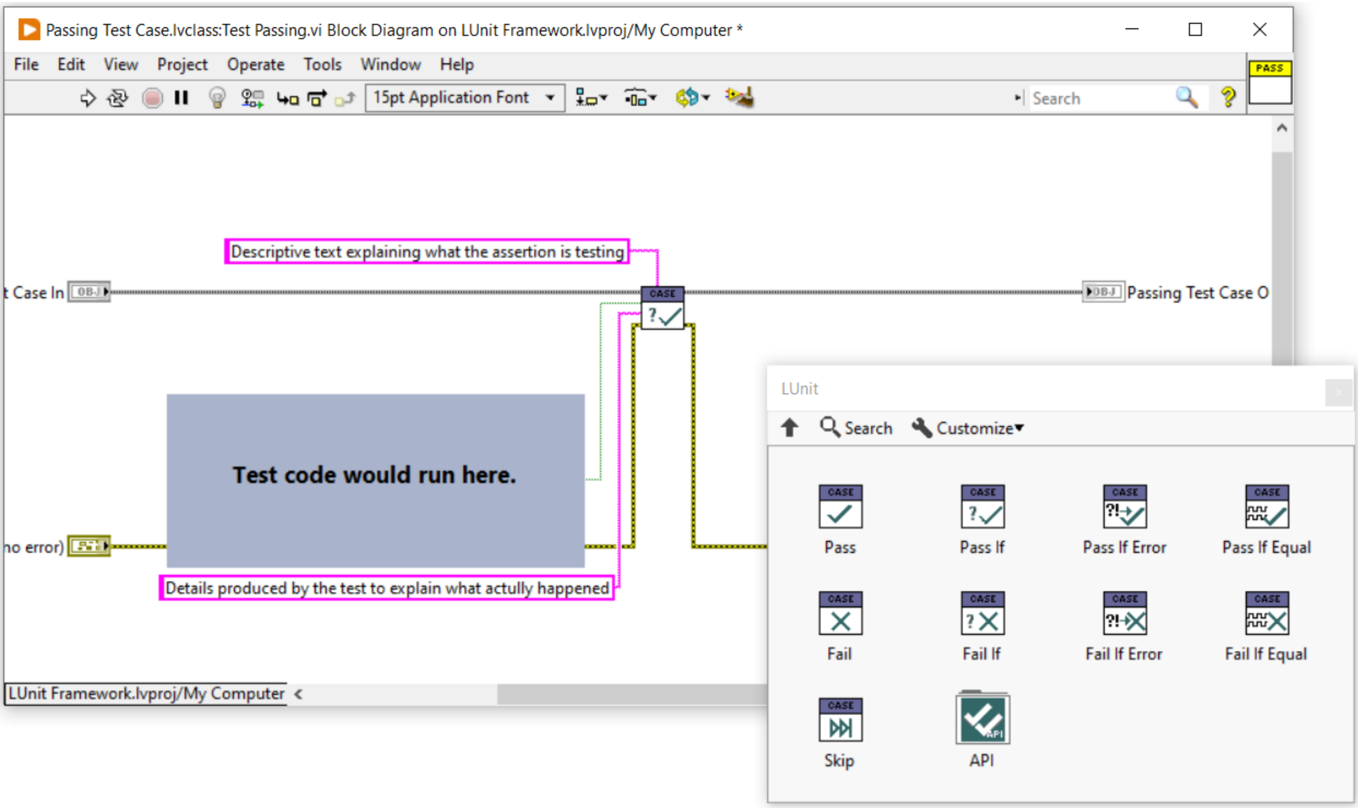
Now you have a test case and may add some test methods to the test case. A test method is a vi belonging to the test case class and will get executed by the framework.

To create a new test method, right-click on the Test Static Test Method.vit and select `New from Template`. **Important:** You need to save the test method using a name starting with the upper or lower case letters `test`.



You can create test methods any way you like and you are free to delete the template methods if you choose to. It is important however that the connector pane uses the same pattern of terminals as the template and that the name of the vi starts with the word `test`.

You should now make your test method test something useful by implementing the block diagram of the vi. To perform tests you will use the assertions available in the provided palette, or using quick drop.



1.4 Using Assertions

The result of each test is determined using assertions. There is a set of assertions to choose from, as shown in the figure above, and the names should be self explanatory. One test method may contain multiple assertions and the result from each assertion will show up in the result view.

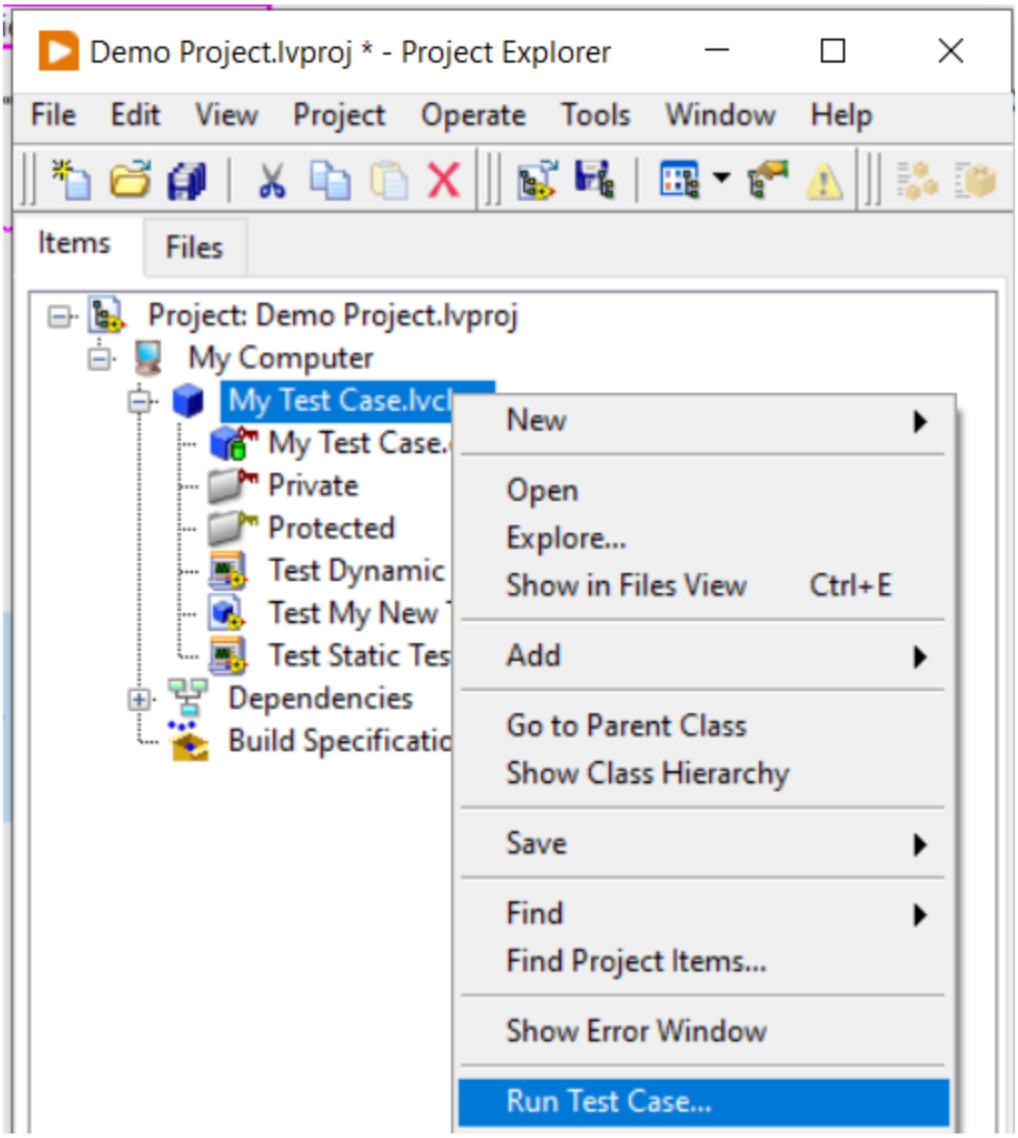
One pro-tip is that the `Pass if Equal.vi` assertion also works well for array data types. The result of comparing arrays will show up in the result view as shown below.

Please note that the `Pass if Equal.vi` assertion will fail if either the type or the value does not match.

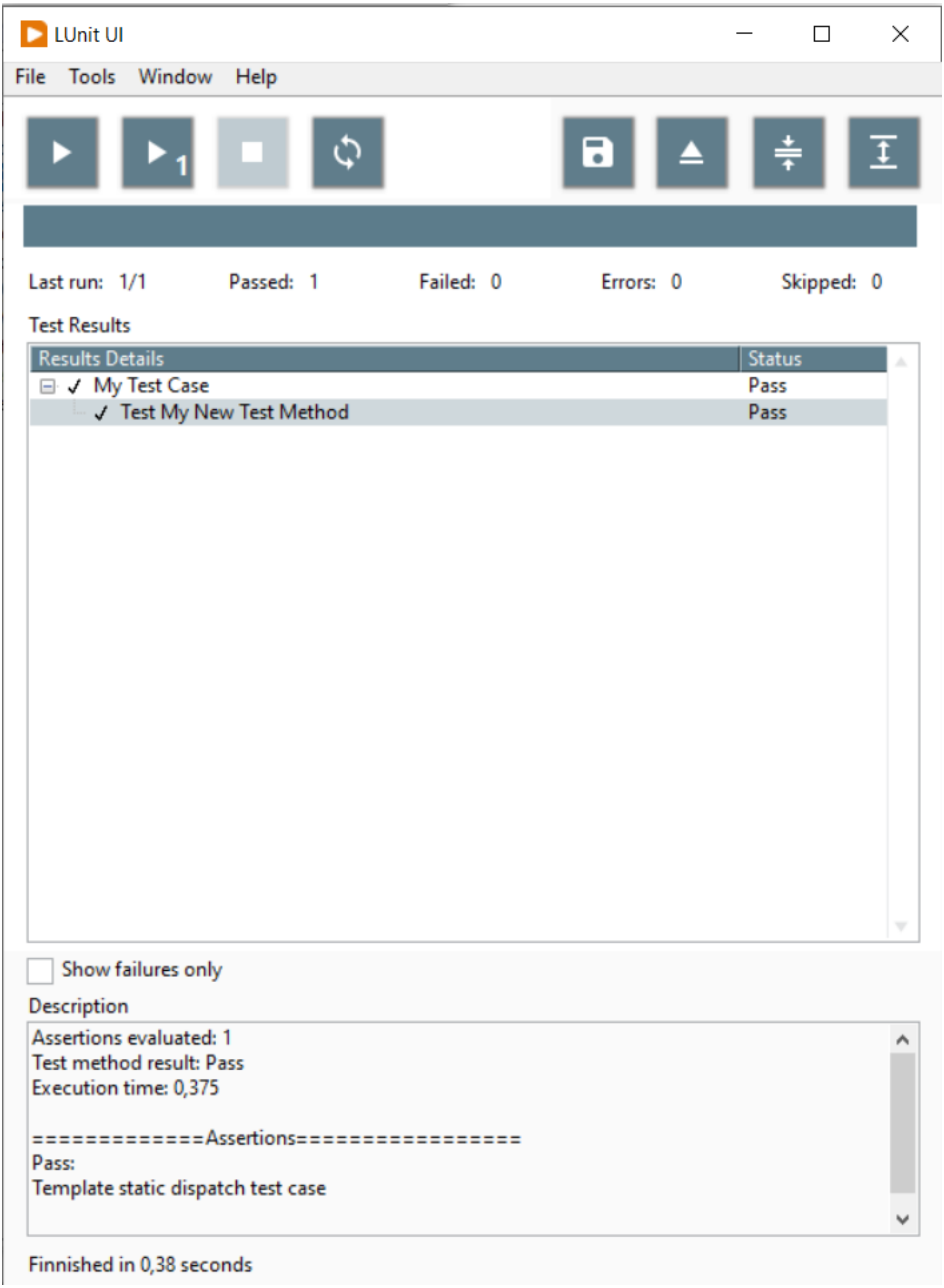
1.5 Running the Test Case

You can run a single test vi (using the Run Arrow) and it will run and show the user interface with the results of the test.

To run all tests contained in a test case, you can right click it in the project window and select the `Run Test Case...` menu option.

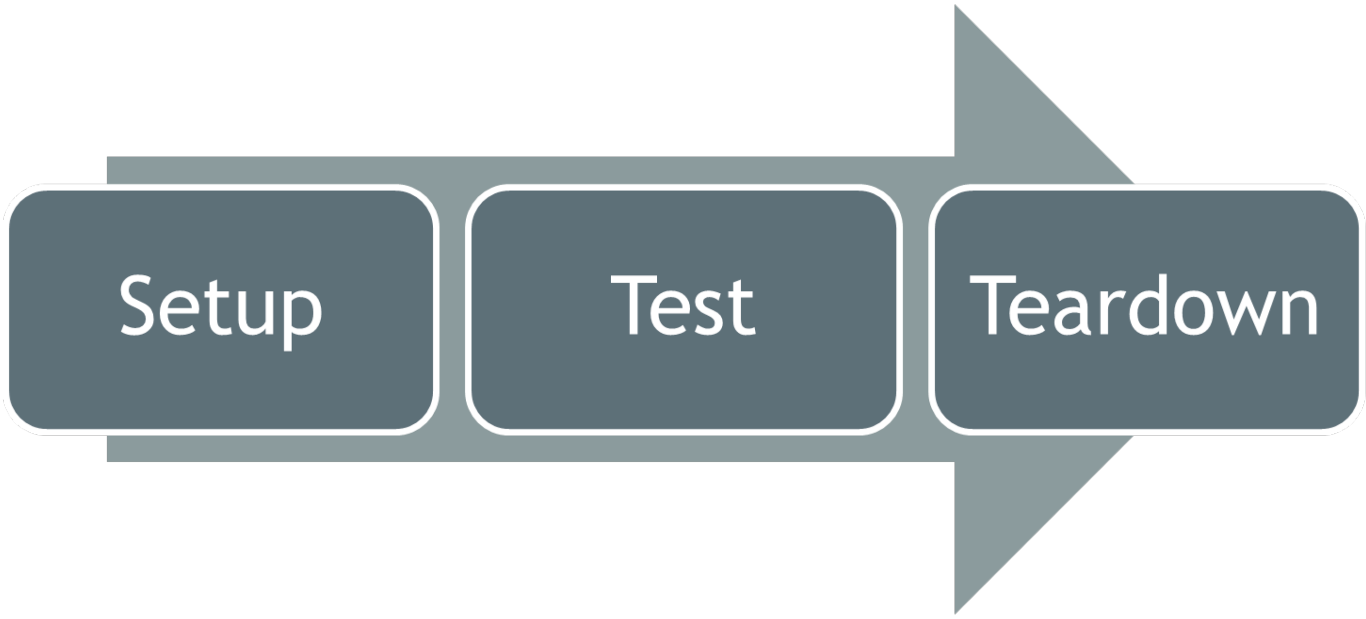


This will open the test execution user interface and run the test case. Alternatively you can also launch the user interface from the tools menu through the `Tools > LUnit > LUnit UI...` menu option. This will open the user interface and show all tests in the current project. As the test is run, the results are also shown as visual icons overlays in the project explorer.


LUnit UI
File
Tools
Window
Help
Last run: 1/1
Passed: 1
Failed: 0
Errors: 0
Skipped: 0
Test Results
Results Details
Status
My Test Case
Pass
Test My New Test Method
Pass
Show failures only
Description
Assertions evaluated: 1
Test method result: Pass
Execution time: 0,375
=============Assertions=================
Pass:
Template static dispatch test case
Finnished in 0,38 seconds

1.6 Using the Setup and Teardown methods

You can add a Setup and a Teardown method to the test case by overriding the corresponding dynamic dispatch vi:s. The Setup vi will run once before each test method in the test case and the Teardown will run once after the test method is completed. This is useful in some cases, but should not be overused as it makes the test methods less verbose. If you need to pass data from the Setup vi to the test method vi or Teardown vi, you can bundle the data into the test case class wire.

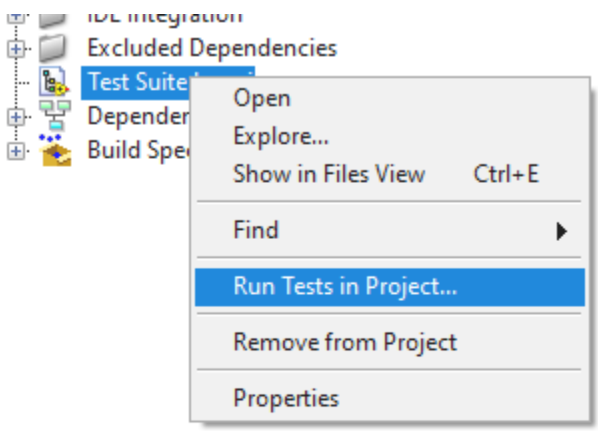


1.7 Organizing Tests

It does make sense to organize tests in some manner, especially as the number of tests increase. There are many common practices around where to keep tests and how to organize the folder structure on disk. In general it is useful to keep tests separated from the source, as there should be no dependency from the source on the test code.

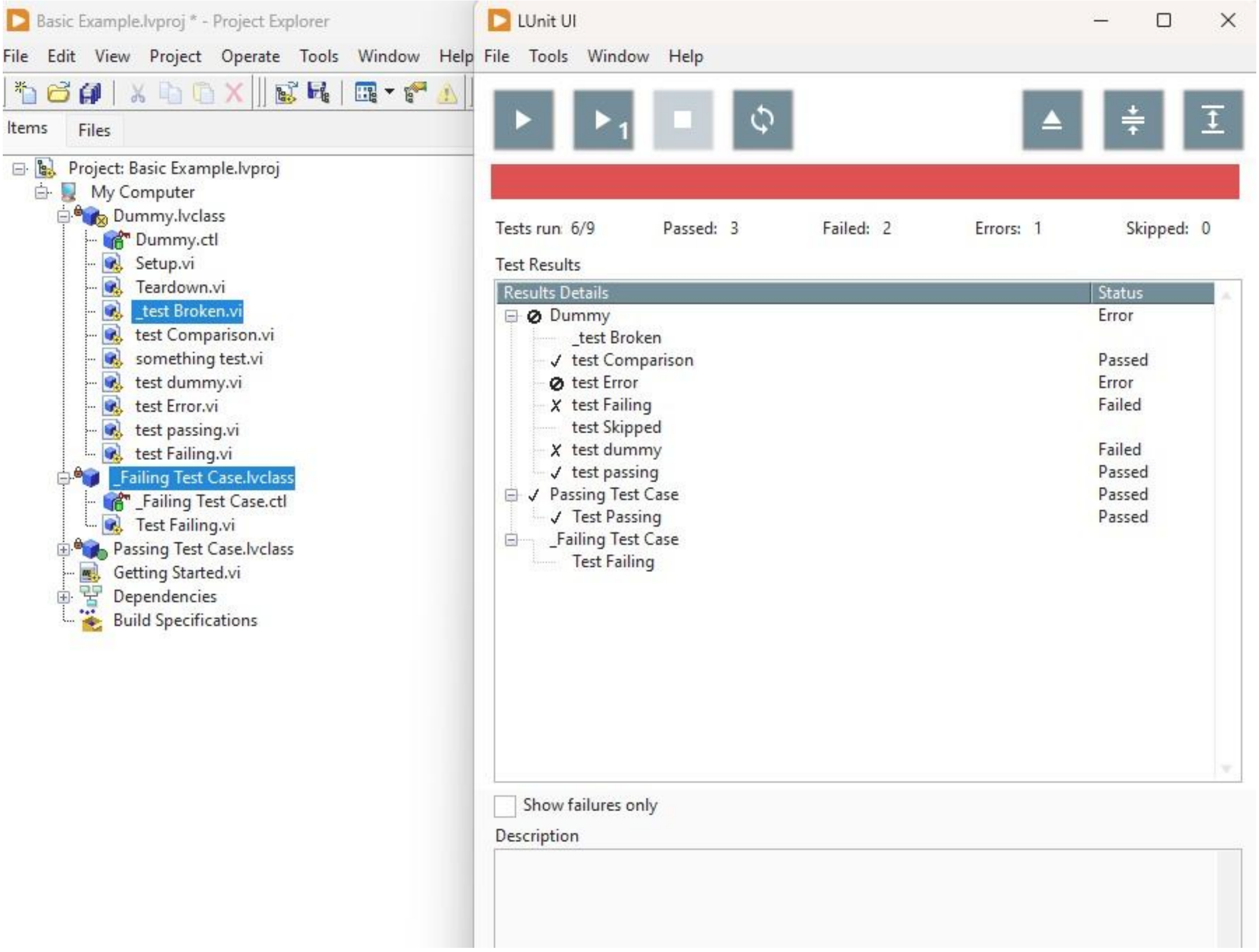
While developing, it is convenient to keep the tests within the active LabVIEW project, as they may then be run quickly through the project integration and you can run all the tests in the project through the tools menu option. As test suites grow, it becomes less convenient as the test time accumulates with larger test suites and some of the tests might not be relevant to the feature under development. While it is important to run the whole suite to catch regression issues, this does not have to be done as frequently as tests covering the new feature.

A good workflow is to keep a separate LabVIEW project file as a test suite where all tests are collected. Tests may be moved from the active project into this test suite and the test suite may then be added to the active LabVIEW project as a project item. All tests within the test suite project may then be executed using the right clikc menu option, but will not be executed when running all tests in current project from the tools menu option and will not need to be loaded with the project.



When executing tests in a Continuous Integration environment, this test suite LabVIEW project file is a good entry point for running tests.

In LUnit version 1.3.1, an additional feature was introduced to help speeding up test execution by skipping slow or unrelated tests. A test method (`vi`) or entire test case class (`lvclass`) may be marked with an underscore `_` character as the first character, making it into a `dashed` test class or test case. `Dashed` tests may be skipped when running through the user interface by using the menu option `Ignore Dashed Tests` as shown below.



Notice that the test "_test Broken.vi" and the entire "_test Failing Test Case.lvclass" were not executed during the test run in the previous image. By toggling the `Ignore Dashed Tests` menu option, it is straight forward to optimize test execution for either execution speed or test coverage. In a typical test driven development workflow, test execution speed must be very fast as the tests are executed repeatedly on time scale of minutes. It is however important to catch regressions by regularly running the full test suite, including the dashed tests. When running LUnit using the API, as would be the case on a Continuous Integration server, dashed tests are never skipped.

2. Framework Architecture

The LUnit unit testing framework is derived from the xUnit architecture. This page briefly introduces the main concepts. The core was rewritten as of version 1.2 and the discussion below is accurate only for versions more recent than 1.2.

2.1 General Architecture

The framework defines a Runnable interface, which, as the name, implies defines a class as runnable. The base Test Case class implements the runnable interface and an xecution of a test is done by instantiating an object of a Test Case class and running it. To run more than one test, multiple instances of the Test Case class are instantiated.

Multiple tests may be aggregated into a Test Suite object, which also implements the Runnable interface, and may be executed by calling the Run VI on the Test Suite. While a Test Case will always be done after one call to the Run VI, a Test Suite may require multiple calls before beeing done. Test Suites may form composites (suites of suites), but this is not visible through the Runnable interface.

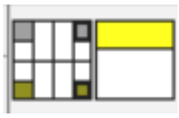
Test Suites are typically generated through test discovery using factory VIs located in the Test Suite class. These VI:s can generate test suites containing all tests in a specific class, library or project.

2.2 Test Case

A Test Case is the base class containing all test methods which are included in the test case. The Test Case class defines two dynamic dispatch test methods called Setup.vi and Teardown.vi. These methods are executed before and after each test method in the test case.

2.3 Test Methods

A test method is a VI belonging to a class inheriting from the Test Case class. The test method name must start with the letters `test` in either upper or lower case. The conector pane of the test case must use the 4-2-2-4 pattern and have the standard connectors for static or dynamic dispatch methods.



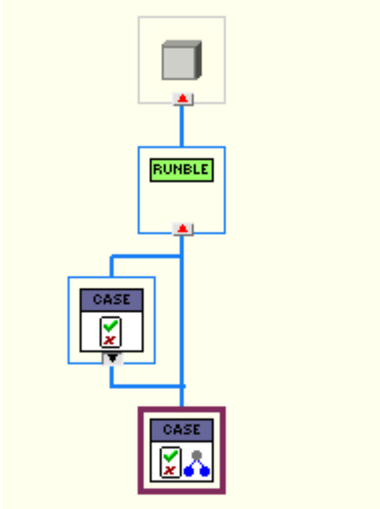
2.4 Static and Dynamic Test Methods

A Test Method may be implemented as either a dynamic or static dispatch vi, just like for any LabVIEW class. The concept is slightly overloaded by the LUnit framework to allow for minimal code duplication when testing class hierarchies.

A static test method will only be executed for the test case in which it has been defined.

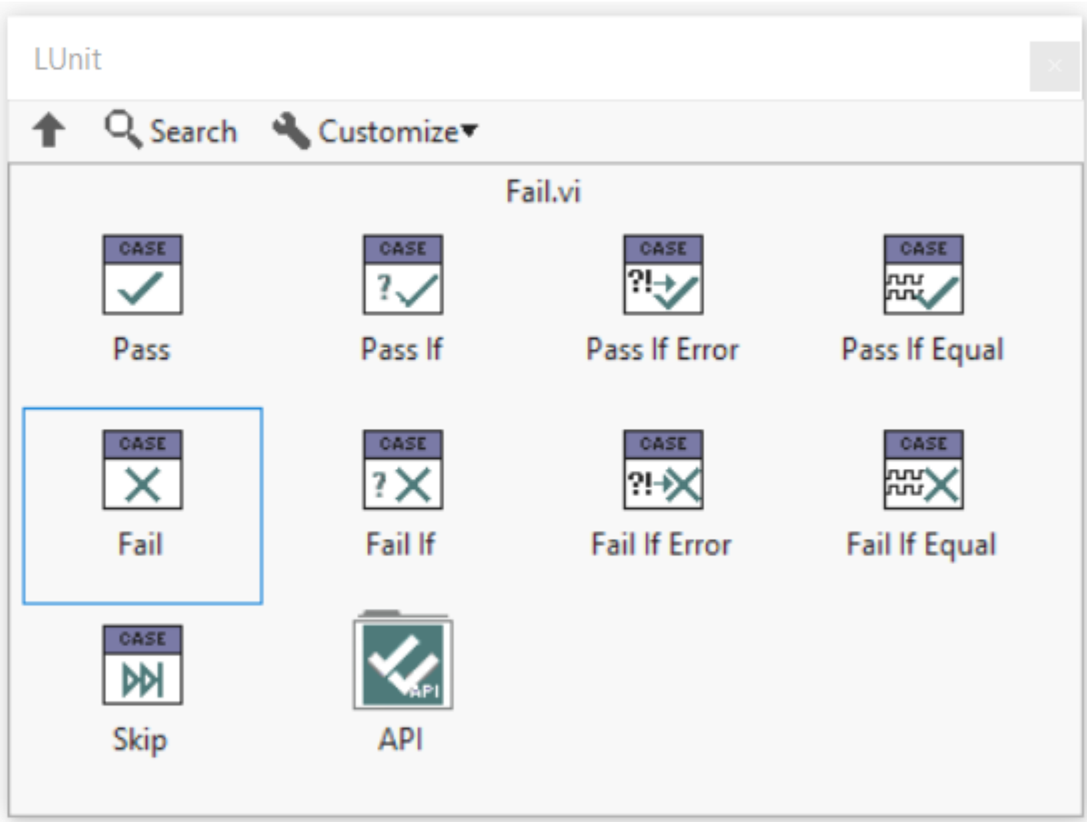
A dynamic test method will be executed for the test case where it is defined and for any test case inheriting from the test case defining it. For this to make sense, a dynamic dispatch test method will need some kind of mechanism for knowing the context which it runs within, *i.e.* what to test. A pattern used for this is illustrated in the example called `Test Inheritance`. In this pattern the class under test is configured by setting the class under test in the Setup.vi method

As of LUnit version 1.4, test cases using inheritance must descend from the `Inheriting Test Case.lvclass` located at *\<vi.lib>/Astemes/LUnit*. As the use of dynamic test methods is much less common than static methods, the complexity of handling test inheritance has been factored out into this class.



2.5 Assertions

Tests are evaluated by one or more assertions called in the test method. The assertions are available from the LUnit palette and the quick drop menu. Assertions are evaluated when the test case executes and the result of the assertions are reported by the framework. Multiple assertions may be used in a single test method and results from all assertions will be available in the test report. A test case will fail if one or more of the assertions fail. Likewise a test case will produce an error result if one or more of the assertions receives an error on the `Error In` terminal.



2.6 Test Runner

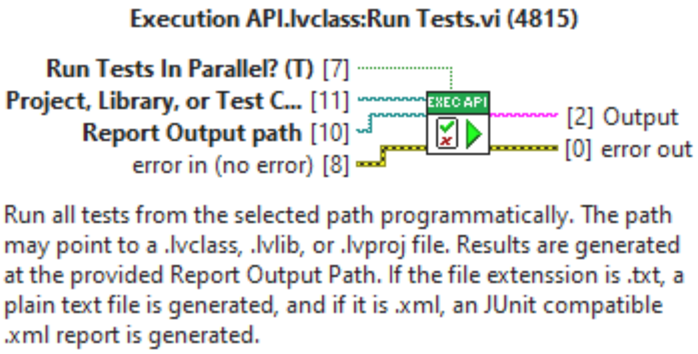
A test runner is a process executing a Test Suite and collecting the results. LUnit supports spawning multiple parallell test runners, which can significantly reduce the test time for large test suites. Test runners may run in separate threads and can leverage a multithreaded processor to run tests concurrently. When the Parallell Test Runner is enabled, tests are grouped into one suite for each Test Case class and all these are then executed in separate threads.

2.7 Test Finder

When launching the LUnit UI, the test finder searches for classes inheriting from the base Test Class within the current application instance. The result is saved into an index file and retrieved on subsequent runs to only search through classes which have changed since last time. To force the test finder to recreate the index, use the refresh test index button in the LUnit UI.

2.8 LabVIEW API

An API is provided for executing tests programatically from LabVIEW. The use of the API is illustrated in the `LUnit API Demo` example.



The API was simplified in version 1.2.6 and now only consists of one VI for running tests from a path. The reason for this change is that the low level API, provided earlier, had some sensitivity to internal changes and made updates more difficult.

The low level API is still used by the high level API method and the low level VIs may be used to alter the behavior of the test execution. Because of the low level nature, this API is more volatile and may break in later releases, while the high level method is very likely to remain stable.

2.9 Low Level API

Before going on, see the warning in the previous section. Now continue on yout own risk.

The low level API may be use to run tests in various ways. Tests are executed using the provided methods and results are returned using User Events, which may be registered for using the provided API method. To use the API methods, an API reference must first be obtained using the `LUnit Open API Reference.vi`. The configuration VI:s `LUnit Configure Reporting.vi` and `LUnit Configure Test Runner.vi` should be used before executing a test.

A test case is executed by calling one of the Run Test API VI:s. To observe the results of the test execution, the `LUnit Register for Events.vi` must be called before starting test execution. Results are returned using a data object through the user event registration.

When the execution has completed a result with type `Test Run` is generated. To abort a running test, use the `LUnit Abort.vi`. When done, use the `LUnit Close API Reference.vi` and unregister for any event obtained from `LUnit Register for Events.vi`.

2.10 Command Line Interface

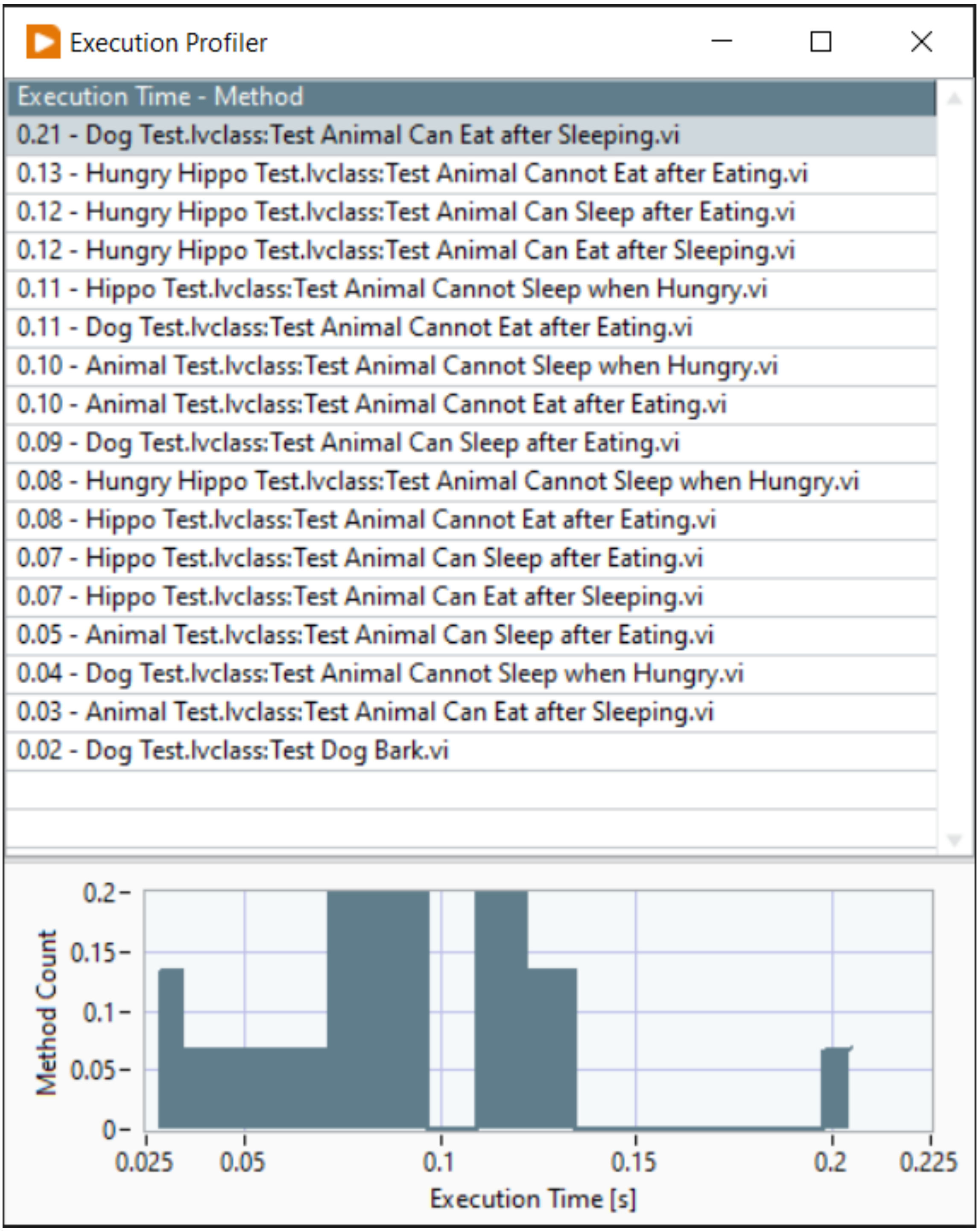
LUnit exposes a command line interface (CLI) to enable execution from continuous integration tools. LUnit adds an operation to the LabVIEWCLI when installed. Please note that LUnit must be installed for the version of LabVIEW called from the CLI.

3. Profiling Tools

LUnit has built in tools to help profile test suites. These tools are meant to be used to identify issues and locate parts needing improvement, but should probably not be used as hard benchmarking tools. The tools are enabled from the `Tools` menu of the LUnit User Interface and the results are displayed after test have been executed.

3.1 Execution Profiler

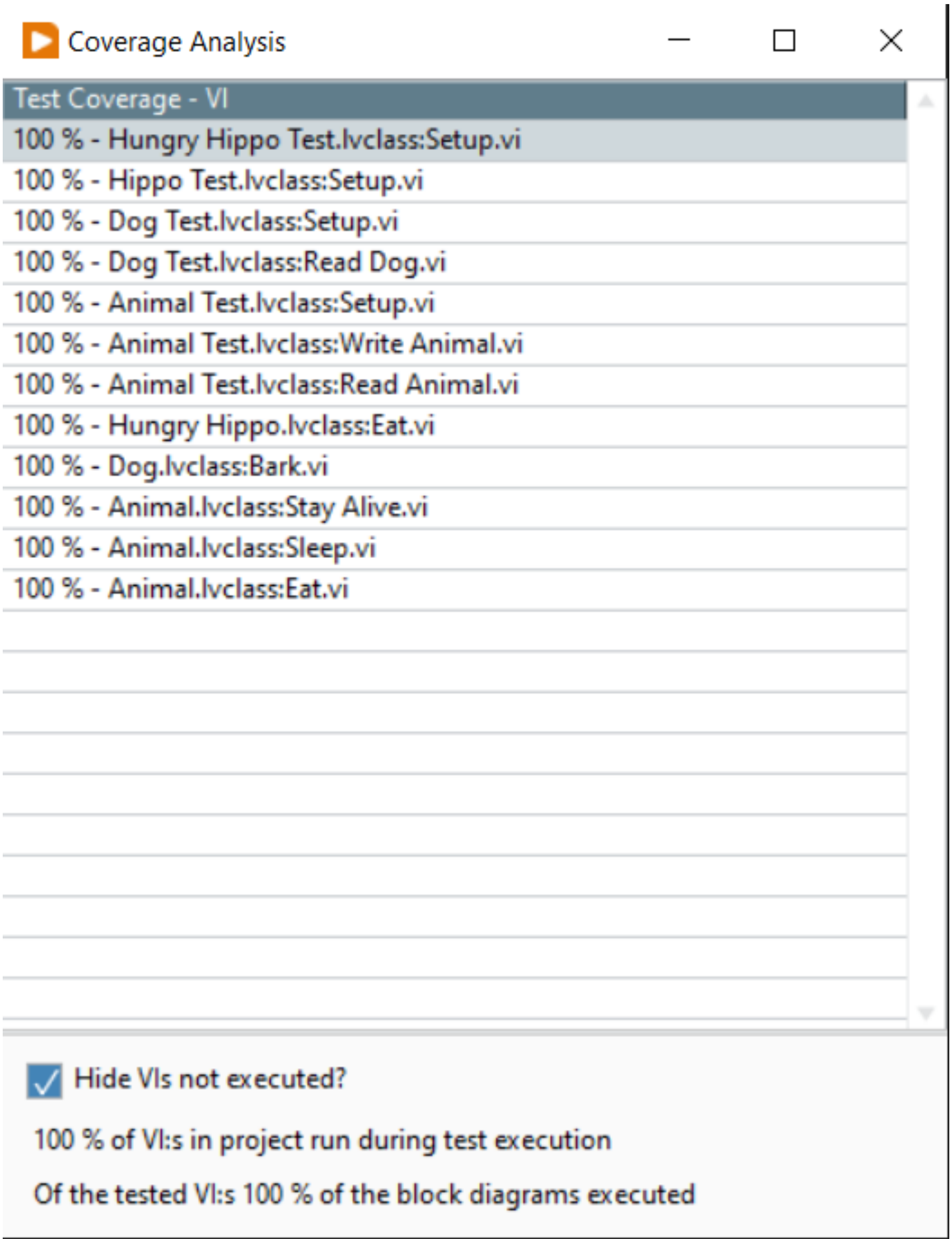
The execution profiler is used to profile the time each test method takes to execute. It is useful for identifying bottlenecks and improve the performance of a test suite. Once activated, the following window is shown after each test execution.



In the table, the execution time for the test and the test name are shown, ordered by test time. The histogram is useful to identify outliers and it is often these tests which have the biggest impact on the test execution time.

3.2 Code Coverage Analyzer

Analysis of code coverage was added as a feature in LUnit version 1.0.6. The code coverage analyzer is useful for identifying parts of the code which has low test coverage. The coverate is reported for each VI as a percentage indicating how much of the block diagram is exercised during test execution.



There are a few caveats to be aware of when using the coverage tool.

- The tool measures how many of the diagrams are executed in a VI as a percentage
- The reported coverage is zero for VIs which do not have debugging activated
- VIs in vi.lib are ignored
- Any VI with a name starting with `test` is ignored

Further, it is important to understand that the number obtained does not tell anything about the quality of the tests. It only reveals how much of the code is exercised by the tests. For a deeper discussion of the usefulness of the metric, please see this blog post.

4. CI Integration

LUnit was designed to easily and natively integrate into continuous integration (CI) pipelines. To achieve this, a way of executing tests from the command line is needed and the results need to be available in a format which may be digested by the CI system.

4.1 Executing Tests from the Command Line

LUnit installs a command line operation using the LabVIEW native LabVIEWCLI by NI. This operation is named LUnit and may be called using LabVIEWCLI -OperationName LUnit. An example ilustrating the useage of the CLI i provided at `...\LabVIEW 20XX\examples\Astemes\LUnit\LUnit CLI Demo.vi`. A path to load tests from is provided using the -ProjectPath argument and the report directory is specified using the -ReportPath argument.

When executing tests from the command line, the test case index is cleared and re-created by default each time. This ensures that all inherited test methods are detected*, at the expense of some overhead for test discovery. The `-ClearIndex` flag may be used to override this behaviour and re-use the index to improve the execution time.

Argument	Description
`-ProjectPath`	The project containing the tests to be executed. The interface also accepts libraries or test case classes of types .lvlib or .lvclass.
`-TestRunners`	Specifies the number of parallell test runners to spawn. Default value is 1.
`-ReportPath`	The output path for the report file generated. The execution generates either a .txt-file or an .xml-file, based on the path specified.
`-ClearIndex`	Clear the index and force LUnit to rediscover all tests. Default is `True`. The index must be cleared to find new tests inherited for a Test Case.

The LabVIEW CLI uses VI Server and by default it is configured to work on port 3363. You will need to make sure that the connection is not blocked by firewalls.

4.2 Capturing the Test Results

Test results are saved in a text based format at the location specified when executing the command line operation.

LUnit has a built in xml-format for test reports which is using the same structure as the one used by JUnit testing framework and specified here. To use the JUnit xml format, you must provide a file path with the `.xml` extenssion. Once the tests have finished, the result file is available at the specified path. File may now be digested by most CI tools. For Jenkins this is done using the JUnit plugin.

4.3 Jenkins Example

Jenkins is a popular open source automation server used for continuous integration and delivery pipelines. A pipeline in Jenkins may be configured using a declarative Jenkinsfile which may be saved directly in the repository. Below is an example showing a basic configuration.

```
pipeline {
    agent any
    environment{
        LV_PROJECT_PATH = "Path to Your LabVIEW Project.lvproj"
        NUM_TEST_RUNNERS = "1"
        LV_PORT = "3363"
    }
    stages {
        stage('Unit Tests') {
            steps {
                bat "LabVIEWCLI -OperationName LUnit -ProjectPath \"${WORKSPACE}\\${LV_PROJECT_PATH}\" -TestRunners ${NUM_TEST_RUNNERS} -ReportPath \"$
{WORKSPACE}\\lunit_reports\\lunit.xml\" -ClearIndex TRUE -PortNumber ${LV_PORT} -LogFilePath \"${WORKSPACE}\\LabVIEWCLI_LUnit.txt\" -LogToConsole true -
Verbosity Default"

                junit "lunit_reports\\*.xml"
            }
        }
    }
}
```

The pipeline above declares three environment variables used to configure the call to LUnit using the LabVIEW CLI. The first is the path to the project file relative to the workspace, *i.e.* the path relative to the root of the repository where the Jenkinsfile is located. The second is the number of parallell test runners to spawn, here configured to one. The third parameter is the port configured for VI server in LabVIEW under Tools->Options->VI Server.

The report is saved in the path `lunit_reports` using the file name `lunit.xml` with incrementing index. After the execution of tests using the bat command the junit plugin is called to digest the report files generated. This requires that the Jenkins JUnit plugin is installed, which it is by using the recommended default settings when installing Jenkins.

Note that this is a minimal example meant to demonstrate the concept. It could be improved significantly to reduce the details in the Jenkinsfile using shared libraries. As an example, the build system used to build LUnit uses a simpler command `runLUnit "${LV_PROJECT_PATH}"` in the Jenkinsfile in stead of the rather detailed `bat` command.

4.3.1 * Footnote on Test Finder indexing

The test finder keeps an index of all test methods for all test classes in the project. When the test finder is started, it loads the index and compares all classes to the index. If the classes has changed since the index was created, the class will be re-indexed. As of version 1.0, the test indexer will however not re-index a class when a parent class has added a dynamic test method. To detect new inherited dynamic method the test index must be re-created, which happens when the `-ClearIndex` flag is left at default value `True`.

5. Real-Time Systems

A good rule of thumb is to only write tests for the code which you want to work. Thus, code running on a real time system does in general qualify for testing. And it is particularly useful to test the code outside production as debugging and troubleshooting a real-time target can be very painful.

5.1 Testing with hardware

Any kind of input/output (IO) is problematic when writing test code. In LabVIEW we typically have IO in the form of user interfaces, hardware drivers, and possibly some databases and files. When running tests on code coupled to IO of some sort, it is in general worth separating the code at the boundaries to test the IO communication separtely from the rest of the application. There are many reasons for this including:

- Testing with hardware requires you to have the hardware setup in order to run the tests. This might not be convenient or even feasible in many cases.
- Including hardware, and the configuration of it, adds a large chunk of complexity as there are more things that can fail.
- Hardware communication adds significant overhead to the test time.
- If the hardware is hidden behind an abstraction, it may easily be replaced.
- Replacing hardware IO with a test double ensures determinism when running the tests.
- Testing that the driver works does not add much value in an automated test environment.

How to use test doubles to avoid testing with real world IO is an interesting topic, but beyond the scope of this document.

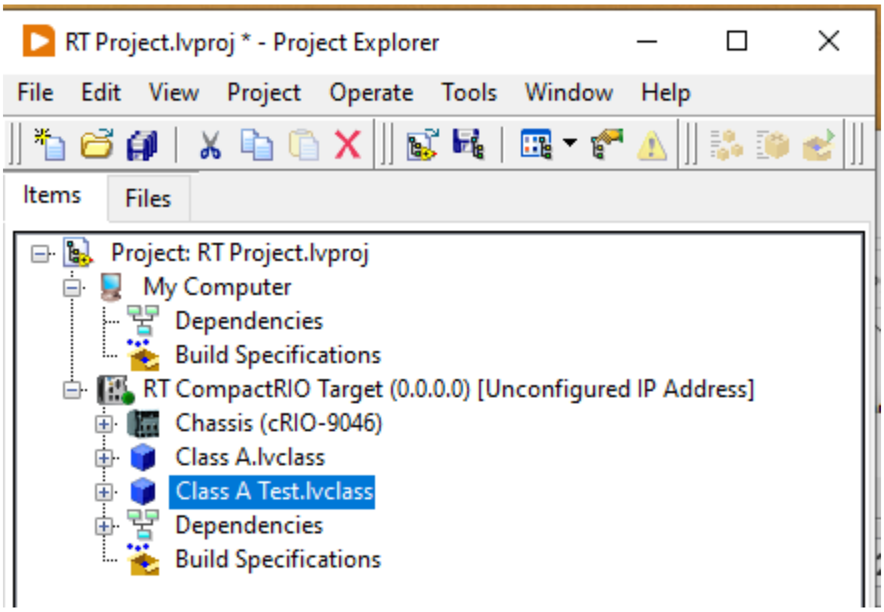
5.2 About running tests on a Real-Time target

While it is very important to test the code on actual hardware, it does not make sense in most cases to run unit tests on a real-time target. To do so would slow down test execution and requires access to the target during development. There are some differences and peculiarities to keep in mind when executing code in an RT environemnt, *e.g.* file paths and unsupported features, but apart from that the code should work the same as under Windows. This means that a failing test under Windows should fail on the RT system as well. Even if the reverse is not allways true, it is often best to assume it is and fall back on debugging when it is not. If running tests on the RT system is necessary, please see this section

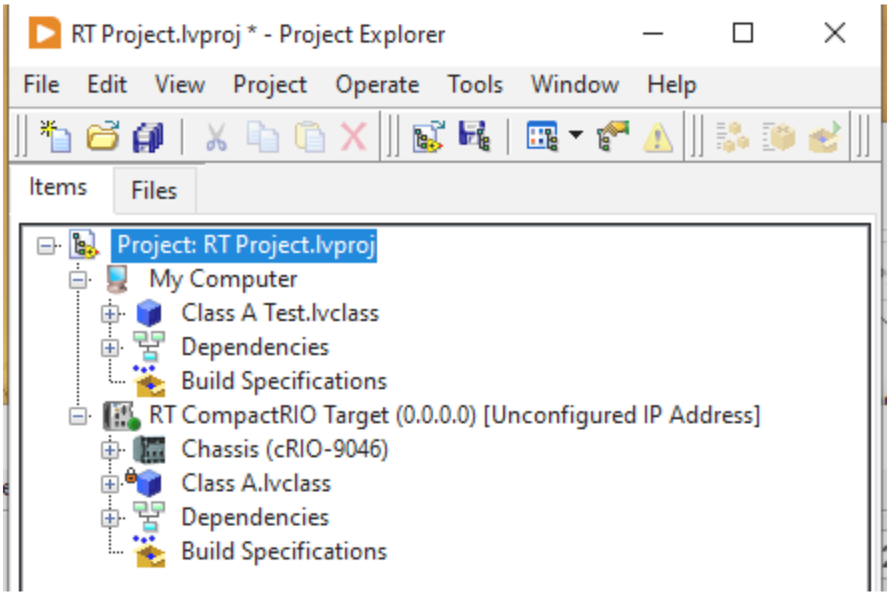
5.3 Working with tests in LabVIEW Real-Time

It is somewhat more tedious to work with tests for code on a LabVIEW Real-Time target compared to vanilla LabVIEW. One reason for this is that libraries gets locked when opened in multiple application instances, and this would happen frequently when developing code on an RT target and testing under Windows. Additionally the code needs to be recompiled for the different targets.

A direct approach would be to keep the test case under the target and run it as normal using the UI.

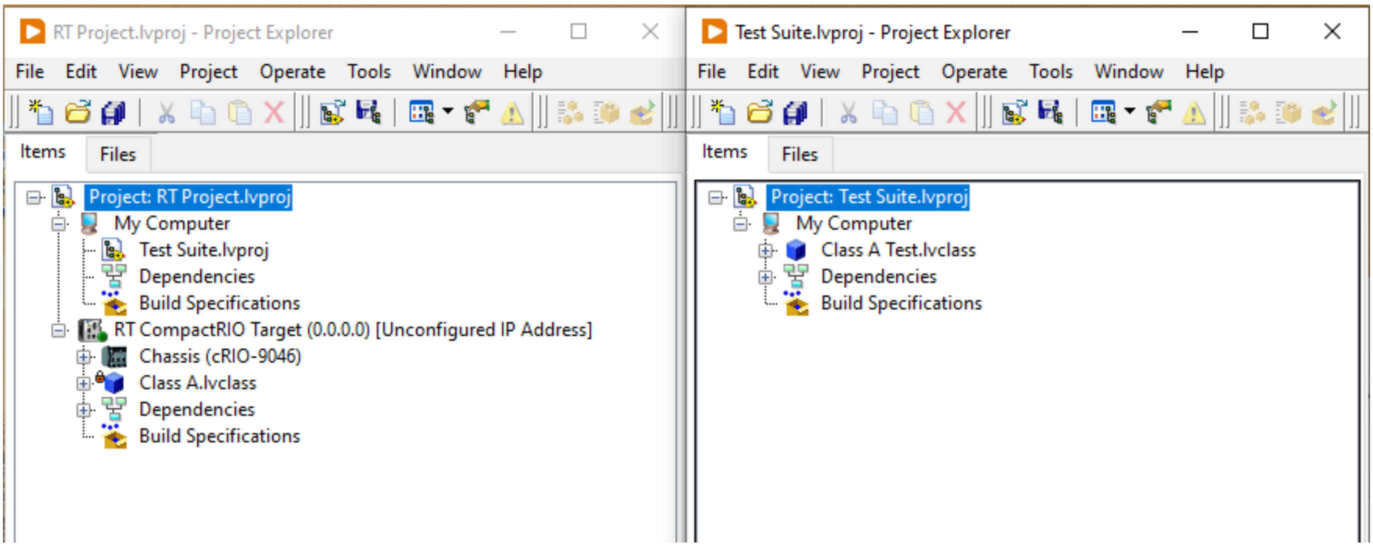


This works in principle, but LabVIEW will recompile the code for each test execution and is for this reason rather slow. This recompilation can be avoided by moving the test case to the My Computer target.

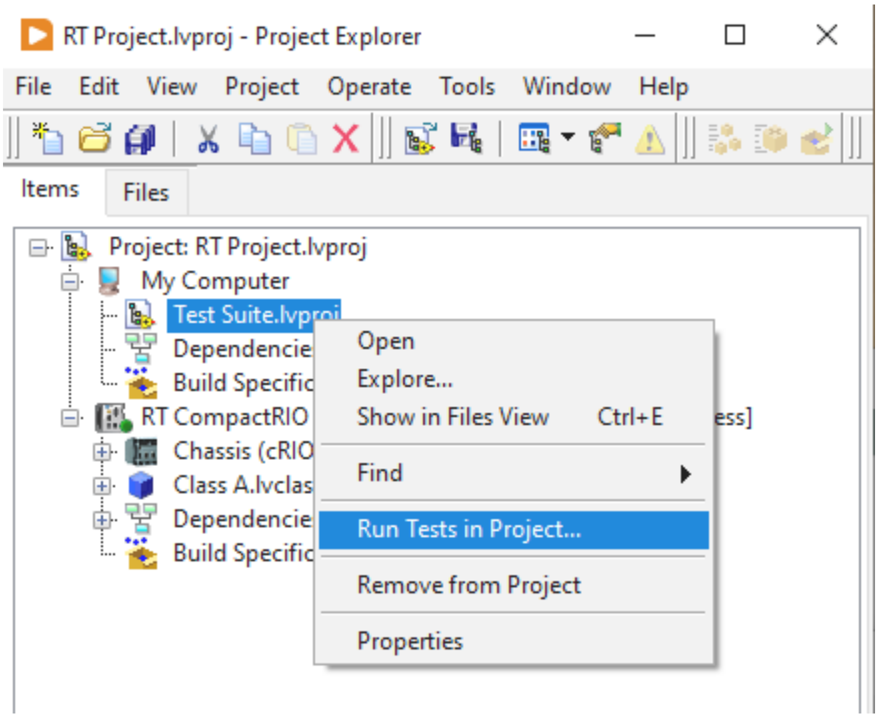


This speeds up the loading of the test as compilation is avoided, but it comes at the cost of locking the class under test. The lock occurs because the test case depends on the class under test, which is then loaded into two separate application instances.

To solve this we can create a new project file to contain the tests and add this to our main project as a project item.



As the project file is treated as a text file by LabVIEW, it will not load anything into memory when added to the project. This implies that the class under test will not be locked unless the test suite is also in memory. The test suite may then be run from the right click menu and tests will not need to be recompiled on subsequent test executions. There will however be a smaller overhead from loading the project into memory when running the tests. Please note that this feature requires LUnit version 1.0.5 or higher.



5.4 Running LUnit on a Real-Time target

There is nothing preventing LUnit from running in a Real-Time environment. However, the Test Execution UI will not be able to run on the target as it uses front panel events not available under LabVIEW Real-Time. Instead the LUnit LabVIEW API should be used to execute tests and collect results when the tests must run on the Real-Time target.

6. Creating Report Plugins

LUnit was designed to be extendable through plugins and a lot of the features of LUnit are implemented as plugins behind the scenes. If you'd like to create your own reporting plugin, this can be done following the guide below. The only requirements for a reporting plugin is that it implements two specific interfaces and is placed in a specific folder within vi.lib.

6.1 Getting started

To create a custom reporting plugin, you will need to create a class inheriting from `LUnit Report.lvclass`. This parent class implements the `LUnit Report Interface.lvclass` and `LUnit Plugin.lvclass` and these will therefore be inherited. The later plugin is a legacy dependency and the plugin methods does not need to be implemented. The dependency on `LUnit Plugin.lvclass` will be deprecated in a future release, but this will not break integration of classes implementing it. The easiest way to get started is to copy one of the existing plugins, which are located at `C:\Program Files (x86)\National Instruments\LabVIEW 202X\vi.lib\Astemes\LUnit\plugins`. It is recommended to use LabVIEW 2020 to ensure backward compatibility.

6.2 Implementing the Report Interface

The report interface is very simple and is called into by the report generator. The `LUnit Open Report File.vi` is called when a new test execution is started, `LUnit Handle Result.vi` is called on the fly for each result generated, and `LUnit Close Report File.vi` is called when the execution is done.

It is important to understand the different types of results which are sent to the `LUnit Handle Result.vi`. Please refer to the Framework Architecture document and the unit tests in the LUnit project fot the native report formats.

6.3 Deploying your plugin

To deploy the plugin, it should be placed in the plugin directory at `C:\Program Files (x86)\National Instruments\LabVIEW 202X\vi.lib\Astemes\LUnit\plugins`. The plugin name should begin with the `LUnit` prefix. The `.lvclass` file should be directly in the `plugins` directory, but the member VI:s of the class will need to be placed in a sub directory to avoid naming conflicts.

After doing this, the plugin should appear in the drop-down menu in the reporting configuration dialog. To automate this, making a VIPM package is recommended and sharing it with the community through VIPM is encouraged.

7. License

7.1 Astemes LUnit

MIT License

Copyright (c) 2021 Anton Sundqvist - Astemes

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

7.2 JKI Flat UI Controls

The JKI Flat UI Controls 2.0 are released under the JKI Toolkits License and include some icons licensed under the Apache License 2.0 and SIL Open Font License 1.1 licenses. To redistribute software you create with the JKI Flat UI Controls 2.0, please be sure to place a copy of this license file, into your distributable application's help docs. Place a copy of this file into your distributable application's help docs.

SOFTWARE LICENSE AGREEMENT JKI TOOLKITS FOR LabVIEW JAMES KRING, INC. NOTICE TO CUSTOMER: PLEASE READ THIS CONTRACT CAREFULLY. BY DOWNLOADING, INSTALLING OR USING ALL OR ANY PORTION OF THE JKI PRODUCT, INCLUDING ANY UPDATE THERETO, YOU AKNOWLEDGE AND AGREE THAT YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. YOU AGREE THAT THIS AGREEMENT, INCLUDING ALL ATTACHMENTS AND EXHIBITS, CONSTITUTES THE COMPLETE AND EXCLUSIVE UNDERSTANDING BETWEEN US, AND SUPERSEDES ALL PRIOR SALES PROPOSALS, NEGOTIATIONS, AGREEMENTS AND OTHER REPRESENTATIONS OR COMMUNICATIONS, WHETHER ORAL OR WRITTEN. YOU FURTHER AGREE THAT THIS AGREEMENT IS ENFORCEABLE AGAINST YOU AND ANY LEGAL ENTITY THAT OBTAINED THE SOFTWARE OR ON WHOSE BEHALF IT IS USED. IF YOU DO NOT AGREE TO THESE TERMS, DO NOT INSTALL, DOWNLOAD NOR USE THIS SOFTWARE. This Software License Agreement (the "Agreement") is made by and between James Kring, Inc. and you as the customer. In this Agreement, you, as an individual as well as any legal entity that obtained the Software or on whose behalf it is used, will be referred to as the "Customer", "you" and "your"; James Kring, Inc. will be referred to as "JKI" or as "we", "us", and "our". WHEREAS, JKI has developed and is willing to supply the computer software, documentation, and related materials that you are downloading and which form all or a part of the JKI Products known as the "JKI Toolkits for LabVIEW" ("the JKI Product") subject to the terms and conditions stated herein; WHEREAS, you desire to have access to the JKI Product, and you are willing to use the JKI Product in accordance with the terms of this Agreement; NOW, THEREFORE, in consideration of the foregoing and the mutual covenants hereinafter set forth, you agree as follows: 1. License to the JKI Product; License Restrictions. 1.1 Scope of License. JKI grants to you a personal, worldwide, non-exclusive, non-transferable (except as permitted under Section 13(a)), perpetual (except as revocable under Section 7) license to use the JKI Product that you obtain under this Agreement, in accordance with the documentation and instructions supplied by JKI, and as follows: (a) for your own internal use, development, testing and evaluation purposes; and/or (b) to copy, modify, create derivative works of, and distribute the JKI Product, through multiple tiers of licensees, only in executable form and only as part of or together with a software or other product developed by you or on your behalf (the "Customer Product"). These grants of license are contingent upon your payment of the associated license fee for the JKI Product (the "License Fee") and your compliance with the terms of this Agreement. For purposes of this Agreement and as reasonably applicable, all modifications, adjustments, enhancements, bug fixes, error corrections or other updates (including any major or minor revisions, which are the "Updates") made by or on behalf of JKI to the JKI Product will become part of the JKI Product. 1.2 License Restrictions. YOU ARE NOT AUTHORIZED TO DISTRIBUTE THE JKI PRODUCT YOU OBTAIN HEREUNDER ON A STANDALONE BASIS (I.E., FOR USE OR DISTRIBUTION INDEPENDENT OF THE CUSTOMER PRODUCT). THE JKI PRODUCT SHALL BE INCORPORATED INTO THE CUSTOMER PRODUCT SUCH THAT JKI TRADE SECRET AND/OR SOURCE CODE INFORMATION IS NOT NOR IS LIKELY TO BE EXPOSED TO SUBLICENSEES NOR ANY THIRD PARTY; FOR EXAMPLE, THE CUSTOMER PRODUCT SHALL NOT BE A SOFTWARE LIBRARY OR "THIN WRAPPER" ON TOP OF THE JKI PRODUCT THAT EXPOSES THE FUNCTIONALITY OR SOURCE CODE OF THE JKI PRODUCT TO ANY THIRD PARTY. YOU WILL NOT MODIFY, DISTRIBUTE OR COMBINE THE JKI PRODUCT WITH ANY OTHER SOFTWARE SO AS TO (I) CREATE, OR PURPORT TO CREATE, OBLIGATIONS, LIMITATIONS, OR RESTRICTIONS ON THE PART OF JKI; OR (II) REQUIRE OR CONDITION THE USE OR DISTRIBUTION OF SUCH SOFTWARE OR PRODUCT ON, THE DISCLOSURE, LICENSING, DELIVERY OR DISTRIBUTION OF ANY SOURCE CODE FOR ALL OR ANY PORTION OF THE JKI PRODUCT. YOU AGREE THAT YOU WILL IMPOSE SIMILAR RESTRICTIONS TO THOSE CONTAINED IN THIS AGREEMENT ON ANY RESELLER, SUBLICENSEE, OR OTHER THIRD PARTY TO WHOM YOU REDISTRIBUTE, SUBLICENSE OR OTHERWISE MAKE AVAILABLE THE JKI PRODUCT. 1.3 Ownership; Proprietary Rights. You acknowledge that the JKI Product, any Updates thereto and their structure and organization are owned by JKI and its suppliers. Accordingly, and except as expressly allowed under this Agreement, you agree (a) not to remove, alter or obscure in any way any proprietary rights notices (including copyright notices and messages indicating the code is JKI property) of JKI or its suppliers on or within the copies of the JKI Product furnished to you by JKI, and (b) that the Customer Product will include in its About Box or other applicable written documentation the notice that the Customer Product "includes the JKI Toolkits for LabVIEW, © (year) JKI. All rights reserved." The JKI Product is licensed, not sold, to you and any and all rights not specifically granted to you by this Agreement, remain in JKI and its suppliers. The JKI Product is protected by copyright, trademark, trade secret and other proprietary rights of JKI, and you do not acquire any rights, express or implied, in the JKI Product, other than those specified in this Agreement. No title to or ownership of the JKI Product, nor any copyright, trademark, trade secret or other proprietary rights in the JKI Product, are transferred to you under this Agreement. All modifications, adjustments, enhancements, bug fixes, error corrections or other updates (including any "Updates") to the JKI Product will become part of the JKI Product and will remain the exclusive property of JKI. 2. Use of Third Party Software You understand that the JKI Product may (a) contain or be distributed with computer programs that are distributed as part of the JKI Product and for which the source code is written by persons or entities other than employees of JKI or contractors under the direction of JKI, and/or (b) include tools that access, interact with and/or utilize software object and/or source code obtained by JKI from third parties and that are separate from the JKI Product (in each case, the "Third Party Software"). (For purposes of this Agreement, the JKI Product and any Third Party Software shall be referred to as the "Software".) Together with its distribution to you of the JKI Product, JKI may make some Third Party Software available to you via download or other distribution. In addition, following your installation of the JKI Product, the JKI Product may be able to, based on your instruction, connect to the internet and identify additional Third Party Software for download and installation on your computer on your behalf. This identification and installation process will require you to provide certain information, including information about the JKI Product as installed on your computer, all of which information will be gathered and used by JKI in accordance with its Privacy Policy then in effect. JKI will endeavor to provide you with a list of the Third Party Software and notice of the associated Third Party Software license(s) and terms as of your receipt of the JKI Product; for this information, please refer to the JKI Product documentation available at http://jkisoft.com/manuals/. YOU ACKNOWLEDGE AND AGREE THAT YOUR USE AND DISTRIBUTION OF ANY SUCH THIRD PARTY SOFTWARE IS SUBJECT TO THE TERMS OF THE APPLICABLE THIRD PARTY SOFTWARE LICENSE(S), AND THAT YOU ARE RESPONSIBLE FOR YOUR COMPLIANCE WITH THE TERMS OF SUCH THIRD PARTY SOFTWARE LICENSE(S). YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT, PRIOR TO USING THE THIRD PARTY SOFTWARE FOR ANY OTHER PURPOSE, AND IN ANY CASE BEFORE COPYING, MODIFYING, OR DISTRIBUTING ANY THIRD PARTY SOFTWARE, YOU WILL CONFIRM THAT YOU HAVE ALL NECESSARY RIGHTS AND PERMISSIONS TO DO SO FROM THE APPLICABLE THIRD PARTY LICENSOR (THE "LICENSOR"), WHICH CONFIRMATION MAY INCLUDE OBTAINING A SEPARATE LICENSE FROM THE LICENSOR EXPRESSLY AUTHORIZING YOU TO DO SO. 3. You Will Not Use nor Disclose Our Confidential Information. Confidential Information hereunder includes, but is not limited to: JKI Product specifications, JKI Product source code, trade secrets, know-how, inventions (whether or not patentable), techniques, processes, programs, ideas, algorithms, schematics, testing procedures, software design and architecture, internal documentation, design and function specifications, product requirements, analysis and performance information, benchmarks, software documents, and other non-public technical, business, product, marketing and financial information, plans and data of JKI (the "Confidential Information"). You agree that all Confidential Information is the confidential property of JKI and, except with JKI's prior written consent or as required (and then only to the extent required) by law, you agree to use any Confidential Information you obtain only as permitted by this Agreement, and not to disclose any Confidential Information to third parties. Notwithstanding the foregoing, you may disclose Confidential Information only to those of your employees and consultants as is necessary for the use expressly and unambiguously licensed hereunder, and only after such employees and contractors have agreed in writing to be bound by the obligations of nondisclosure at least as restrictive as those contained in this Agreement. Your nondisclosure obligations hereunder shall not apply to information you can document: (i) is generally available to the public other than through breach of this Agreement; (ii) is rightfully disclosed to you by a third party without any associated obligation of confidentiality; or (iii) is independently developed by you without use of or reference to any JKI Confidential Information. Because of the unique and proprietary nature of the Confidential Information, you understand and agree that JKI's remedies at law for your breach of your obligations under this Section may be inadequate and that JKI shall be entitled to seek equitable relief (including without limitation provisional and permanent injunctive relief and specific performance). Nothing stated herein shall limit any other remedies provided under this Agreement or available to JKI at law. Upon expiration or termination of this Agreement for any reason, you will return or destroy all copies of all JKI Confidential Information in your possession or control. 4. Communications and Feedback. 4.1 Feedback. In the course of your use of the JKI Product and in connection with any related support or other services that may be offered to you by JKI (the "Services), you may provide JKI with comments and feedback regarding your use and evaluation of the JKI Product, including any defects found therein and any recommendations for changes or modifications to the JKI Product (the "Feedback"). Such Feedback may include, but will not be limited to, any communications from you to JKI, including (i) any messages, content, materials or other communications posted to the http://jkisoft.com, http://jameskring.com or any other website(s) owned or maintained by JKI; and/or (ii) relating to your use and evaluation of the JKI Product. Feedback may include communications regarding: (1) which portions of the JKI Product have been used, (2) the nature of that use, (3) the extent or amount of use, (4) any errors or difficulties discovered and (5) the characteristic conditions and symptoms of the errors and difficulties. You acknowledge and agree that (i) JKI may use, in any manner and for any purpose, the information gained as a result of your use and evaluation of the JKI Product, including but not limited to the Feedback; (ii) any corrections, modifications, upgrades or improvements to the JKI Product based on such Feedback or other input shall be owned and retained entirely by JKI; and (iii) JKI shall have no obligation to correct, upgrade, modify, or otherwise support or maintain the JKI Product pursuant to this license. 4.2 Assignment. If you are ever held or deemed to hold any right, title or interest (including, without limitation, any intellectual property rights, moral rights or trade secret rights) in or to: (a) the JKI Product (including any changes, modifications or corrections thereto) and/or (b) the Feedback, whether by virtue of your provision of Feedback to JKI or otherwise, then you hereby irrevocably assign to JKI all such right, title and interest. Such assignment includes all rights in or to any invention, work of authorship, mask work, idea, information, feedback or know-how (whether or not patentable) that is conceived, learned or reduced to practice in the course of performance under this Agreement and any patent rights, copyrights (including moral rights; provided that any non-assignable moral rights are waived to the extent permitted by law), trade secret rights, mask work rights, sui generis database rights and all other intellectual and industrial property rights of any sort with respect thereto that in any way relate to or constitute the Feedback or the JKI Product. In the event that any such rights (including, by way of example and without limitation, "moral rights," or other similar rights) cannot be assigned, you hereby agree to waive enforcement worldwide of such rights against JKI and hereby grant to JKI an exclusive, fully paid, worldwide, irrevocable, perpetual license, with right to sublicense through multiple tiers of sub-licensees, to use, reproduce, create derivative works of, publicly perform, publicly display, transfer, assign and distribute in any medium or format, whether now known or later developed, any and all property that is subject to such rights. You agree to take any action reasonably requested by JKI to evidence, perfect, obtain, maintain, enforce or defend the foregoing, including executing any and all documents necessary to implement and confirm the letter and intent of this Agreement. 5. NO WARRANTY OR PROMISES; HIGH RISK ACTIVITIES 5.1 NO WARRANTY OR PROMISES. Your right of termination and refund described in Section 7 constitutes your sole and exclusive remedy with respect to any dissatisfaction with the JKI Product. THE JKI PRODUCT, ANY UPDATES THERETO, AND ALL THIRD PARTY SOFTWARE IS DISTRIBUTED BY JKI ON AN "AS IS" BASIS, WITHOUT ANY WARRANTY PROVIDED BY OR ON BEHALF OF JKI. PLEASE REFER TO THE APPLICABLE THIRD PARTY SOFTWARE LICENSE FOR ANY WARRANTY THAT MAY BE OFFERED OR DISCLAIMED BY THE LICENSOR OF SUCH SOFTWARE. WE MAKE NO OTHER WARRANTIES OR REPRESENTATIONS AS TO ANY SOFTWARE PROVIDED HEREUNDER, AND WE HEREBY DISCLAIM ALL EXPRESS AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. WE DO NOT WARRANT THAT ANY JKI PRODUCT OR ANY THIRD PARTY SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE. 5.2 HIGH RISK ACTIVITIES. THE JKI PRODUCT IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH HAZARDOUS OR "HIGH RISK" ACTIVITIES OR WITH

APPLICATIONS THAT REQUIRE FAIL-SAFE PERFORMANCE (TOGETHER, "HIGH RISK" ACTIVITIES). HIGH RISK ACTIVITIES INCLUDE BUT ARE NOT LIMITED TO ACTIVITIES OR APPLICATIONS RELATING TO THE OPERATION OF NUCLEAR FACILITIES, AIR TRAFFIC CONTROL, AEROSPACE OPERATIONS, OR DIRECT LIFE SUPPORT MACHINES, AND ANY OTHER ACTIVITIES OR APPLICATIONS IN WHICH THE FAILURE OF THE SOFTWARE COULD LEAD DIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE. ACCORDINGLY, WE SPECIFICALLY DISCLAIM ANY AND ALL EXPRESS OR IMPLIED WARRANTIES OF FITNESS FOR HIGH RISK ACTIVITIES, AND YOU AGREE THAT JKI WILL HAVE NO LIABILITY OR RESPONSIBILITY RELATING TO YOUR USE OR OPERATION OF THE SOFTWARE IN CONNECTION WITH HIGH RISK ACTIVITIES. 6. INDEMNIFICATION BY YOU. YOU AGREE TO, AT YOUR EXPENSE, INDEMNIFY AND HOLD HARMLESS JKI FROM AND AGAINST ANY AND ALL LIABILITIES, LOSSES, ACTIONS, DAMAGES, OR CLAIMS (INCLUDING ALL REASONABLE EXPENSES, COSTS, AND ATTORNEYS FEES) THAT RESULT FROM YOUR DIRECT OR INDIRECT MISAPPROPRIATION OF ANY INTELLECTUAL PROPERTY RIGHTS CONTAINED IN ANY SOFTWARE; YOUR VIOLATION OF ANY APPLICABLE LAW; OR YOUR USE OF THE SOFTWARE IN CONNECTION WITH ANY HIGH RISK ACTIVITIES. 7. Termination. You may terminate your license(s) to the JKI Product within thirty (30) days of your receipt of a purchased license to the JKI Product for any reason or no reason and receive a refund for the License Fee you have paid with respect to such purchased license (a "Customer Termination"). This Agreement may be terminated by JKI immediately upon notice of any breach by you of the provisions of this Agreement. Upon any termination, all licenses granted hereunder shall terminate and you shall immediately cease all use and redistribution of the JKI Product. Upon any termination, you shall immediately destroy all copies of the JKI Product, together with any and all documentation regarding the JKI Product, any other Confidential Information and any and all copies and extracts of the foregoing. All other terms of this Agreement shall remain in effect following termination. 8. OUR LIABILITY IS LIMITED. BY DOWNLOADING, INSTALLING AND/OR USING THE JKI PRODUCT, YOU AGREE THAT, DESPITE ANY OTHER PROVISION OF THIS AGREEMENT OR OTHERWISE, JKI WILL NOT BE LIABLE OR OBLIGATED WITH RESPECT TO THE SUBJECT MATTER OF THIS AGREEMENT UNDER ANY CONTRACT, NEGLIGENCE, STRICT LIABILITY OR OTHER LEGAL OR EQUITABLE THEORY: (I) FOR ANY AMOUNTS IN EXCESS OF THE TOTAL OF THE LICENSE FEES PAID TO US HEREUNDER IN THE TWELVE (12) MONTHS PRECEDING ANY CLAIM; (II) FOR ANY COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES OR RIGHTS; OR (III) FOR INTERRUPTION OF USE OR LOSS OR CORRUPTION OF DATA. DESPITE ANY OTHER PROVISION OF THIS AGREEMENT, WE SHALL NOT BE LIABLE NOR OBLIGATED WITH RESPECT TO THE SUBJECT MATTER OF THIS AGREEMENT OR UNDER ANY CONTRACT, NEGLIGENCE, STRICT LIABILITY OR OTHER LEGAL OR EQUITABLE THEORY: (I) FOR ANY MATTER BEYOND OUR REASONABLE CONTROL, OR (II) FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES, LOST DATA OR LOST PROFITS, EVEN IF WE ARE INFORMED OF THEIR POSSIBILITY. THESE LIMITATIONS SHALL APPLY DESPITE THE FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDY. ANY ACTIONS BASED ON OR ARISING OUT OF THIS AGREEMENT, OR RELATING TO THE SOFTWARE SUPPLIED HEREUNDER, MUST BE BROUGHT WITHIN ONE YEAR OF THE DATE OF TERMINATION OF THIS AGREEMENT. 9. Payment. You agree to pay us the License Fee(s) for the JKI Products you are purchasing concurrently, at the time you accept the terms of this Agreement, or as JKI may otherwise agree in writing. All payments are non-cancelable and non-refundable, except for your right of refund upon Customer Termination as described in Section 7. Fees charged by us do not include any sales, use, excise, value-added, or similar taxes, and do not include any duties or fees payable on the delivery of software in countries other than the United States. Any such taxes, duties, or fees shall be either added to our invoice or paid directly by you. You will not, however, be liable for taxes imposed on us based on our income. 10. Publicity. Except as provided under Section 4 hereof, neither party may issue press releases or endorsements which reference the other party or make any use of the other party's name, logo or trademark without the prior written consent of the other party. 11. You Will Comply with Export Regulations and other Applicable Laws; Transfer of Personal Data. On your own behalf and on behalf of your sublicensees and any third parties to whom you redistribute or make available the JKI Product: You agree that the Software will not be shipped, transferred or exported into any country or used in any manner prohibited by the United States Export Administration Act or any other applicable laws, restrictions or regulations (collectively the "Laws") In addition, if all or any portion of the Software is identified as an export controlled item under any export Laws, you represent and warrant that you are not a citizen, or otherwise located within, an embargoed nation (including without limitation Iran, Iraq, Syria, Sudan, Libya, Cuba, North Korea, and Serbia) and that you are not otherwise prohibited under any Laws from receiving or using the Software. If you reside in any part of the European Union or any other jurisdiction in which the transfer of your personal data may apply, you expressly consent to the transfer of any personal or other data identifying or relating to you or the entity on whose behalf you are accepting this Agreement. You agree that you will impose similar restrictions to those contained in this Agreement on any reseller, sublicensee, or other third party to whom you redistribute, sublicense or otherwise make available the JKI Product. 12. U.S. Government Users. The JKI Product is a "Commercial Item," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 and 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the JKI Product is being provided to U.S. Government end users (1) only as a Commercial Item, and (2) with only those rights as are granted to all other end users pursuant to the terms and conditions of this Agreement. 13. Miscellaneous. (a) You may not assign, transfer, or sublicense any obligations or benefit under this Agreement without the written consent of JKI. This restriction shall not apply to any redistribution of software as provided under a separate Third Party Software license allowing for such redistribution. (b) We agree that we are independent contractors and neither of us has the right or authority to assume or create any obligation or responsibility on behalf of the other. (c) All notices under this Agreement shall be in writing, and shall be deemed given when personally delivered or three (3) days after being sent by prepaid certified or registered mail to the address of the party to be noticed as set forth herein or such other address as such party has provided to the other. (d) No failure or delay in exercising any right hereunder will operate as a waiver thereof, nor will any partial exercise of any right or power hereunder preclude further exercise. (e) If any provision of this Agreement shall be adjudged by any court of competent jurisdiction to be unenforceable or invalid, that provision shall be limited or eliminated to the minimum extent necessary so that this Agreement shall otherwise remain in full force and effect and enforceable. (f) This Agreement shall be deemed to have been made in, and shall be construed pursuant to the laws of, the State of California and the United States without regard to the conflict of law provisions thereof. The United Nation's Convention on Contracts for the International Sale of Goods is expressly excluded from application to this Agreement. The sole venue for all disputes relating to this Agreement shall be in San Francisco County, California. (g) This Agreement may be executed in any number of counterparts, each of which shall be considered an original, but all of which together will constitute one and the same instrument. (h) This Agreement constitutes the entire agreement between us pertaining to the subject matter hereof, and any and all written or oral agreements previously existing between the parties are expressly cancelled. This Agreement may be modified, replaced or rescinded only in writing, and signed by a duly authorized representative of each party. (i) In any action to enforce this Agreement the prevailing party will be entitled to reasonable costs and attorneys' fees. In the event that any of the provisions of this Agreement shall be held by a court or other tribunal of competent jurisdiction to be unenforceable, such provisions shall be limited or eliminated to the minimum extent necessary so that this Agreement shall otherwise remain in full force and effect and enforceable. (j) You acknowledge and agree that JKI will treat any information it gathers about or from you in accordance with its Privacy Policy currently in effect and available at http://jkisoft.com/legal/privacy/. 14. BASIS OF BARGAIN. EACH PARTY RECOGNIZES AND AGREES THAT THE WARRANTY DISCLAIMERS AND LIABILITY AND REMEDY LIMITATIONS IN THIS AGREEMENT ARE MATERIAL, BARGAINED FOR BASES OF THIS AGREEMENT AND THAT THEY HAVE BEEN TAKEN INTO ACCOUNT AND REFLECTED IN DETERMINING THE CONSIDERATION TO BE GIVEN BY EACH PARTY UNDER THIS AGREEMENT AND IN THE DECISION BY EACH PARTY TO ENTER INTO THIS AGREEMENT. February 2008

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/ TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION 1. Definitions. "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document. "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License. "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity. "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License. "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files. "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types. "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below). "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof. "Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution." "Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work. 2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form. 3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed. 4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions: 1. You must give any other recipients of the Work or Derivative Works a copy of this License; and 2. You must cause any modified files to carry prominent notices stating that You changed the files; and 3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and 4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License. 5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions. 6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file. 7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License. 8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages. 9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability. END OF TERMS AND CONDITIONS

SIL OPEN FONT LICENSE Version 1.1 - 26 February 2007 PREAMBLE The goals of the Open Font License (OFL) are to stimulate worldwide development of collaborative font projects, to support the font creation efforts of academic and linguistic communities, and to provide a free and open framework in which fonts may be shared and improved in partnership with others. The OFL allows the licensed fonts to be used, studied, modified and redistributed freely as long as they are not sold by themselves. The fonts, including any derivative works, can be bundled, embedded, redistributed and/or sold with any software provided that any reserved names are not used by derivative works. The fonts and derivatives, however, cannot be released under any other type of license. The requirement for fonts to remain under this license does not apply to any document created using the fonts or their derivatives. DEFINITIONS "Font Software" refers to the set of files released by the Copyright Holder(s) under this license and clearly marked as such. This may include source files, build scripts and documentation. "Reserved Font Name" refers to any names specified as such after the copyright statement(s). "Original Version" refers to the collection of Font Software components as distributed by the Copyright Holder(s). "Modified Version" refers to any derivative made by adding to, deleting, or substituting — in part or in whole — any of the components of the Original Version, by changing formats or by porting the Font Software to a new environment. "Author" refers to any designer, engineer, programmer, technical writer or other person who contributed to the Font Software. PERMISSION & CONDITIONS Permission is hereby granted, free of charge, to any person obtaining a copy of the Font Software, to use, study, copy, merge, embed, modify, redistribute, and sell modified and unmodified copies of the Font Software, subject to the following conditions: 1) Neither the Font Software nor any of its individual components, in Original or Modified Versions, may be sold by itself. 2) Original or Modified Versions of the Font Software may be bundled, redistributed and/or sold with any software, provided that each copy contains the above copyright notice and this license. These can be included either as stand-alone text files, human-readable headers or in the appropriate machine-readable metadata fields within text or binary files as long as those fields can be easily viewed by the user. 3) No Modified Version of the Font Software may use the Reserved Font Name(s) unless explicit written permission is granted by the corresponding Copyright Holder. This restriction only applies to the primary font name as presented to the users. 4) The name(s) of the Copyright Holder(s) or the Author(s) of the Font Software shall not be used to promote, endorse or advertise any Modified Version, except to acknowledge the contribution(s) of the Copyright Holder(s) and the Author(s) or with their explicit written permission. 5) The Font Software, modified or unmodified, in part or in whole, must be distributed entirely under this license, and must not be distributed under any other license. The requirement for fonts to remain under this license does not apply to any document created using the Font Software. TERMINATION This license becomes null and void if any of the above conditions are not met. DISCLAIMER THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.