Low-Level Tests

Framework for lightweight, module-centric tests.

Low-Level Tests (LLTs) is a testing framework for lightweight, module-centric tests in Unreal Engine (UE). Low-Level Tests are written with the Catch2 test framework. You can build and execute Low-Level Tests for all platforms supported by UE. Low-Level Tests include the following test types:

- **Explicit**: Self-contained tests defined by a module and target pair.

Tests can use any of the following test methodologies:

- **Unit**: Test a standalone section or unit of code.
- **Integration**: Test multiple sections or units of code combined as a group.
- **Functional**: Test specific functionality of a feature or use case.
- **Smoke**: Quick validation of a feature or use case.

- **End-to-end**: Test several stages of a feature.
- **Performance**: Benchmark the running time of a feature.
- **Stress**: Try to break functionality by straining the system.

You can write Low-Level Tests with either of the following test paradigms:

- **Test-Driven Development (TDD)**
- **Behavior-Driven Development (BDD)**

Why Use Low-Level Tests

What sets Low-Level Tests apart from other UE test frameworks is their minimality with respect to compilation and runtime resources. LLTs are made to work with various UE features, including: UObjects, assets, engine components, and more. Low-Level Tests in Unreal Engine are written with Catch2 — a modern C++ test framework — extended to include test grouping and lifecycle events, as well as other features that work well with the modular architecture of Unreal Engine.

Guide to Low-Level Tests Documentation

The remainder of this page gives a brief overview of several common testing methodologies.

Below is an overview of what this documentation covers. Common to all use cases is the Build and Run documentation section, which is essential for learning how to run your tests.

Unreal Engine Low-Level Tests Documentation

The areas covered in the Unreal Engine Low-Level Test Documentation include:

- Types of Low-Level Tests
- Write Low-Level Tests
- Build and Run Low-Level Tests

Catch2 Documentation

This documentation does not provide a comprehensive resource for the features that Catch2 provides. For more information about Catch2, see the Catch2 repository on GitHub. For detailed information about Catch2, including writing tests in Catch2, see the Catch2 Documentation.

Testing Methodologies

This section gives you a brief overview of the different testing methods that Low-Level Tests can help you implement.

Unit tests

Unit tests test one unit of code, typically a single method within a class. They rely on mocking inputs and external dependencies such as servers or databases. Typically, you write one unit test suite per tested class. The goal of a unit test is to cover the public interface of a class, as well as the different code paths within a method.

Most tests are not unit tests because they are very restrictive in how they should be written, and the target code might not be testable in this strict way. Unit tests should only require minimal special global setup (set up mocks, stubs, and fakes) and teardown, and they should be able to be run independently between test suites. Unit tests should not have an order dependency on other unit tests. Unit tests are very fast — they take seconds or less to run.

Integration tests

Integration tests test multiple units of code together, typically two or more classes or methods. They can require global setup, such as loading modules or an external resource. They are less restrictive than unit tests and are more common, but it can be harder to cover branches in code (if conditions etc.). Integration tests are usually slower than unit tests — they take up to seconds to run.

Functional tests

Functional tests test a specific functionality, typically a single feature or a use case. They often require global setup and teardown to manage external resources. These are the most common types of tests, and they can vary greatly in complexity. Functional tests can take seconds to minutes and rarely hours to run. They are usually slower than integration tests.

Smoke tests

Smoke tests provide quick validation of a feature or use case. These tests cover minimum acceptance criteria. They can be run at the startup of an application if it takes a couple of seconds or in development builds. Typically included in continuous integration, iterative builds.

End to end tests

End-to-end tests run through several stages of a feature as opposed to just a segment. They are heavier tests that might require minutes or more to complete. End-to-end tests usually have checkpoints with preconditions that can stop the test.

Performance and stress tests

Performance tests are often benchmarks and are typically long running. **Stress tests** target one functionality, and they try to break it through repeated actions or by putting the system under strain. Both types typically capture performance metrics that are compared with baselines. Both types can be slow and usually take up to hours to complete but there's no general rule as to how much time they should run for. Some performance tests might only require seconds or a few minutes to complete.

Learn More

Types of Low-Level Tests

Learn more tests and how to structure them in your Unreal Engine code.

Types of Low-Level Tests

Determine which type of test is best for your use case.

Write Low-Level Tests

Learn how to write Low-Level Tests with Catch2 in Unreal Engine, including general guidelines and best practices.

Write Low-Level Tests

Learn how to write Low-Level Tests in Unreal Engine, including naming conventions and best practices.

Build and Run Low-Level Tests

Learn to build and run Low-Level Tests in Unreal Engine with Visual Studio, Unreal Build Tool, and BuildGraph.

Build and Run Low-Level Tests

Learn about the different ways to build and run Low-Level Tests.