Automation Test Framework

Overview of the automation system used for unit testing, feature testing, and content stress testing.

The **Automation System** is built on top of the **Functional Testing Framework**, which is designed to do gameplay level testing, which works by performing one or more automated tests. Most tests that are written will be functional tests, low-level core or **Editor** tests that need to be written for using the **Automation Framework** system. These tests that are written can be broken down into the following categories depending on their purpose or function:

Test Type	Description
Unit	API level verification tests. See `TimespanTest.cpp` or `DateTimeTest.cpp` for examples of these.

Test Type	Description
Feature	System-level tests that verify such things as PIE, in-game stats, and changing resolution. See `EditorAutomationTests.cpp` or `EngineAutomationTests.cpp` for examples of these.
Smoke	Smoke tests are just considered a speed promise by the implementer. They are intended to be fast so they can run *every time* the Editor, game, or commandlet starts. They are also selected by default in the UI. All Smoke tests are intended to complete within 1 second. Only mark Unit Tests or fast Feature Tests as Smoke Tests.
Content Stress	More thorough testing of a particular system to avoid crashes, such as loading all maps or loading and compiling all Blueprints. See `EditorAutomationTests.cpp` or `EngineAutomationTests.cpp` for examples of these.
Screenshot Comparison	This enables your QA testing to quickly compare screenshots to identify potential rendering issues between versions or builds.

Overview

You can use the **Automation Test Framework** to perform automated unit, feature, and content stress tests on your project.

The Automation Test Framework is:

- Built directly in C++.
- Not associated with the `UObject` environment.
- Designed to function directly in Unreal Engine's core modules.

- Relies on Unreal Engine systems to work properly, so it is not ideal for pure unit testing. See Low-Level Tests for more information on pure unit testing.

Interfaces

Several interfaces are built on the framework to make writing tests easier, depending on your goals.

- Automation Spec: Supports Behavior Driven Design (BDD) methodology.
- Automation Driver: Supports user input simulation.
- Functional Testing: Supports Level testing with Blueprint.
- Screenshot Comparison Tool: Supports screenshot capture and comparison.
- FBX Test Builder: Supports FBX file testing.
- Editor Testing with Blueprint: Create Editor tests with Blueprints.
- Editor Testing with Python: Create Editor tests with Python.
- CQTest: Supports test syntax simplification for asynchronous execution and provides test fixtures.

Tests in Plugins

Containing tests in plugins has the following benefits. You can:

- Enable them individually.
- Choose to include them with the project packaged builds when compiled.
- Store content outside of the project's Content folder.

Enabling Automation Test Plugins

To enable the automation test plugins, do the following:

1. Select **Edit > Plugins** to open the **Plugin** panel.
2. In the **Plugin** panel, select the **Testing** filter on the left.
3. Enable the desired test plugins.
4. Restart the **Unreal Editor**.

Test Design Guidelines

Epic Games follows the following guidelines for our automation testing:

- Do not assume the state of the game or the Editor. Tests can run out of order or parallel across machines.
- Leave the state of the files on disk the way you found them. If a test generates a file, delete it when the test completes.
- Assume the test was left in a bad state the last time it ran. A good habit is generating and deleting files before the test starts.

Running Automation Tests

1. Open any project.
2. Enable the Plugins available to use for testing and restart the Editor.
3. In the Editor, go to **Window** > **Test Automation**.

   For this menu option to be visible, you must first enable at least one automation tests plugin.

4. In the Automation tab of the Sessions Frontend under the **Test Name** column, enable the following:
   - Editor

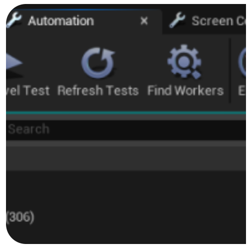
    - Project
    - System
5. In the Automation tab toolbar, click the **Start Tests** button. As the tests complete, you can follow the progress in the Test Name window.

Essentials

The **Automation System** provides the ability to perform Unit Testing, Feature Testing, and Content Stress Testing using the power of the **Unreal Message Bus** to increase stability.

Setting up an Automation Test Report Server

Instructions for setting up an Automation Test Report Server.



Automation System User Guide

A guide to using the Automation System for Running Tests.

Gauntlet Automation Framework

A framework to run sessions of projects in Unreal Engine that perform tests and validate results.