Clang Sanitizers

Instrument Clang sanitizers in your builds for Android and Linux to quickly diagnose problems.

UnrealBuildTool (UBT) supports **Clang sanitizers** for Linux and Android. This page contains information about which sanitizers are supported and how to use them in your builds.

Overview

There are many types of programming errors that are not normally caught at compile time, as the compiler recognizes them as valid use of C++ syntax. Even though the syntax is correct, gaps in the logic can lead to undesired behavior, such as:

- Memory leaks
- Race conditions
- Uninitialized memory
- Out-of-bound access in arrays
- Integer overflow

Normally, these can only be diagnosed at runtime through trial-and-error debugging. By building your application with Clang's sanitizers enabled, you can find these errors more rapidly with a log output that directly reports them.

You can find thorough details on Clang's sanitizers and their benefits in Clang's documentation.

Running Build Tool With Sanitizers

To instrument a sanitizer in your build, run UnrealBuildTool with the UBT argument for the desired sanitizer, and the sanitizer will be linked in your final executable. For example, the following command line builds your project with **Address Sanitizer (ASan)** linked in your build:

Commandline

```
Build\BatchFiles\Build.bat MyGame Linux Development -WaitMutex -FromMsBuild -EnableASan
```

Copy full snippet

When you run your application, you will see the sanitizer's error detection output in your logs. You can run these builds directly through a command line, or you can open your project's **Properties** in **Visual Studio**, then add the sanitizer's command to **NMake** > **Build Command Line**.

Clang's sanitizers typically consume extra memory and slow down the programs they are linked in by at least 2x and sometimes as much as 15x, depending on the sanitizer. Review Clang's specifications for each sanitizer for more information.

Supported Sanitizers

The following sanitizers are supported by UBT:

Sanitizer	UBT Command	Description
Address Sanitizer (ASan)	-EnableASan	Detects a variety of memory access problems, including out-of-bound access errors and memory leaks. This is deprecated since Android 14.
HWASan (Android only)	-EnableHWASan	Hardware-accelerated version of ASan for Android. Should use 20-30% less memory. This is preferred for Android 14 and newer, and needs NDK 26.1.10909125 or higher.
Thread Sanitizer (TSan) (Doesn't work on Android)	-EnableTSan	Detects threading issues like race conditions.

Sanitizer	UBT Command	Description
Undefined Behavior Sanitizer (UBSan)	-EnableUBSan	Detects anything that C++ recognizes as undefined behavior, like out-of-bounds errors, integer overflow, or uninitialized memory.
MinUBSan (Android only)	-EnableMinUBSan	Minimalistic version of UBSan for Android.
Memory Sanitizer (MSan) (Linux only)	-EnableMSan	Detects attempts to read uninitialized memory.