THIS IS A BETA FEATURE

This feature is still in beta and subject to constant change. We encourage you to use it in test projects and provide your feedback to us.

However, DO NOT use it in production where it creates dependencies! Always back up your projects to make sure that you can go back to a previous version.

The CRYENGINE plugin system exposes support for creating plugins in both C++ and C# and allows users to simply drop a plugin into their project and to see the benefits instantaneously.

Intended Benefits

- **Buy/Sell Plugins on the Marketplace**
  In the future we want to offer a variety of plugins on the Marketplace - these will not only be developed by Crytek, but by 3rd parties too and will give you all the functionality you need to create your project.

- **Simplify game code creation**
  Size down the scope of code you have to touch when implementing elemental features for your project. You can use C++ as well as C# to achieve this.

- **Create a custom project without once touching code**
  The days of having to modify legacy game projects are in the past, with the new system you can create a project by just adding pre-written plugins to it.

- **Making more parts of the Engine optional**
  Gain performance by removing systems that are not required by your project. No need to maintain dead code.

- **More control/info about features used by your project**
  Get proper information about the plugins loaded at runtime, increase iteration time by being able to hot-reload any plugin while the Engine is running.

- **Easier Engine customization for your project**
  Make your project changes even more accessible to your team members by using a completely dynamic working environment for the plugins.

Changes

Changes to ExtensionSystem:

There are no compatibility issues with existing extensions.

Changes to CryMonoBridge & MonoRuntime:

CryMonoBridge is now part of every Windows project. This will increase compile times because of the swig-process, but it also ensures that C# support can be easily enabled without any further changes or special configurations. The MonoRuntime has changed a lot. The MonoLauncher is no longer loaded, but all the C# plugins are managed directly from the plugin manager. This allows users to keep control over every single binary loaded.

Changes to C# projects/plugins:

To make your C# binary compatible with the new plugin system, just write a wrapper class around it that inherits from ICryEnginePlugin. ICryEnginePlugin offers you all the necessary interfaces to CRYENGINE. The plugin system takes care of calling the methods mentioned below:

- **void Initialize()**
  Gets called whenever the plugin manager initializes your plugin for the first time.

- **void OnGameStart()**
  Gets called whenever the Engine starts your game (either after initialization in the Launcher or when switching to in-game mode in the Sandbox Editor)

- **void OnGameStop()**
  The equivalent for OnGameStart()

- **void Shutdown()**
  Gets called whenever the plugin system is trying to shutdown your plugin. Every Engine-related allocation should be removed here. After shutdown the class will await clean up by the framework garbage collector.

On top of this, the plugin manager takes care of setting up a link to the global DomainHandler as well as setting the specified Binary and Asset directories for this plugin. Can be accessed within the plugin through the following methods:

- **const char* GetBinaryDirectory()**
  Retrieves the binary directory for this plugin on your local machine.

- **const char* GetAssetDirectory()**
  Retrieves the asset directory for this plugin on your local machine.

All these methods are also available to native C++ plugins. In the future, we plan to match them even closer in functionality and have an identical workflow on both sides.

Feedback

As this is a Beta Feature and is still in development, then we would love to hear what you think about it. Please provide us with any feedback you have through the **CRYENGINE Community forum**!