Process for IL2CPP

1. In your code replace:
   using System.Diagnostic.Process;
   with
   using KS.Diagnostic.Process;
   or add
   using Process = KS.Diagnostics.Process;
   using ProcessStartInfo = KS.Diagnostics.ProcessStartInfo;
2. Use Process as you would usually, for example see ProcessTest.cs script in Example folder or 2nd page in this document.
3. Data from OutputDataReceived may be returned in a different thread, add Dispatcher component to empty gameobject and run code using Dispatcher.Invoke to make sure the main thread is used. Check ProcessExample scene for more details.

You can request any missing methods/properties to be implemented here:
https://discord.gg/qdQQrjZneS

```
public class ProcessTest : MonoBehaviour
{
   [SerializeField] Text text;

   IEnumerator Start()
   {
       yield return new WaitForSeconds(1);

       var mainFolder = Application.dataPath;
       var exePath = Path.Combine(mainFolder, "Kamyker",
           "Process for IL2CPP", "Example", "NativeLibraryConsoleTest.exe");

       // this example work only in editor, for building application .exe could be
placed for example in StreamingAssets folder
https://docs.unity3d.com/Manual/StreamingAssets.html

       // var mainFolder = Application.streamingAssetsPath;
       // var exePath = Path.Combine(mainFolder,"NativeLibraryConsoleTest.exe");

       var proc = new Process()
       {
           StartInfo = new ProcessStartInfo()
           {
               FileName = exePath,
               Arguments = "",
               UseShellExecute = false,
               RedirectStandardOutput = true,
               CreateNoWindow = true,
           },

           //enables rising Exited event
           EnableRaisingEvents = true
       };
       proc.OutputDataReceived += (s, d) =>
           // Dispatcher is used to run on unity main thread
```

```
            Dispatcher.Invoke(() => text.text += Environment.NewLine + d.Data);

        proc.Exited += (s, d) =>
        {
            Dispatcher.Invoke(() => text.text += Environment.NewLine + "Exited");
            proc.CancelOutputRead();
            proc.Dispose();
        };

        proc.Start();
        proc.BeginOutputReadLine();
    }
}
```