
dotPeek Documentation

Release 2018.1

Well Fired Development

Sep 12, 2018

1	Introduction	3
1.1	About .Peek	3
1.2	About the documentation	3
1.3	Organisation of the documentation	4
2	Frequently Asked Questions	5
2.1	Is .Peek representative of the final platform build	5
2.2	Can .Peek be used on Continuous Integration server	5
2.3	Can I use .Peek reports outside of the .Peek interface	5
3	Installing	7
3.1	Package Contents	7
3.2	Installing	7
4	Quick Start	9
4.1	Your first build report	9
4.2	Next Step	10
5	.Peek Front Page	11
6	Build Report Panels	13
6.1	Overview	13
6.2	Build Settings	15
6.3	Used and Unused Assets	15
7	Settings	19
8	Programmatic Control	21
8.1	Settings	21
8.2	Callbacks Order	22
8.3	Manually start .Peek metric counters	22
9	Continuous Integration	25
9.1	Getting informed when build report is ready	25
9.2	VCS Version	26
10	.Profile API	29
10.1	Classes	29

10.2	Interfaces	41
10.3	Namespaces	41
10.4	Enums	42

Welcome to the official documentation for .Peek, a Unity editor tool that allows you to automatically generate a report of the used (and unused) assets in your builds, and keep an eye on the evolution of your project assets in a nicely integrated and responsive interface.

If you are reading the .pdf version of this documentation, you may find the online version easier to navigate. It is accessible on <https://dotpeek-documentation.readthedocs.io/en/2018.1/>.

We recommend you read the *Introduction* to get an overview of what this documentation has to offer.

The table of contents below and in the sidebar should let you easily access the documentation for your topic of interest. You can also use the search function in the top left corner.

Note: Notice something wrong with our documentation? Feel free to submit a pull request.

If you have a technical question, please feel free to contact us through our keybase team [wellfiredltd.technicalsupport](https://keybase.io/wellfiredltd)

The main documentation for the site is organized into the following sections:

This page aims at giving a broad presentation of the tool and of the contents of this documentation, so that you know where to start if you are a beginner or where to look if you need info on a specific feature.

1.1 About .Peek

Most of Unity projects requires a large amount of assets (textures, audio files, prefabs, plugins...). These assets have a direct impact on the size of your build and the performance of your game. Tracking these assets to ensure your project stays up to quality standards and does not become a nightmare to maintain is very hard, especially when you are working in a team of people with different background.

.Peek is here to support you in quickly identifying the usual suspect, before it becomes a real problem. It is a must have for any development team who wants to control what goes into their build and react on time when undesirable assets are integrated to the project.

Here a list of the core features :

- Automatically generate and archive a build report each time a build is performed.
- Provide a nice interface to quickly jump between build reports and compare the content of each build.
- Provides a list of unused assets for each builds.
- Provide the possibility to share build reports on a VCS, or to save it in different location for each team member.
- Possibility to run build generation silently or to totally shut it off.
- Very responsive UI, even for projects with a large amount of assets, thanks to the usage of [.Guacamole](#), an open source MVVM framework for Unity.

1.2 About the documentation

This documentation is continuously written, corrected, edited and revamped by members of the .Peek team and community. It is edited via text files in the [reStructuredText](#) markup language and then compiled into a static website/offline

document using the open source [Sphinx](#) and [ReadTheDocs](#) tools.

Note: You can contribute to .Peek's documentation by opening issues through [YouTrack](#) or sending patches via pull requests on its [GitHub source repository](#).

1.3 Organisation of the documentation

This documentation is organised in five sections, the way it is split up should be relatively intuitive:

- The *General* section contains this introduction as well as the *Frequently Asked Questions*.
- The *Getting Started* section gives you a quick entry point to start using the tool.
- Finally, the *Class API reference* is the documentation of the .Peek API. It is generated automatically from files in the main repository, and the generated files of the documentation are therefore not meant to be modified.

Frequently Asked Questions

2.1 Is .Peek representative of the final platform build

.Peek is primarily focusing on tracking down what is included in your build and how it influences the size of it over time. The size reported for each asset is not necessarily representative of the final size of your build once it is fully packaged for the platform it is meant to run on. For example, the size of the final IPA produced for IOS is far different from the sum of the assets copied to the XCode project before it is compiled. But the size reported at the end of each Unity build is consistent with the previous builds. So it still gives you a nice overview of what was added, and how it influenced the size of the build.

2.2 Can .Peek be used on Continous Integration server

Yes! And we strongly encourage you to do so. By specifying a path relative to your unity project in .Peek settings, all your build reports can be generated on your CI machine and can be archived next to your other artifacts.

2.3 Can I use .Peek reports outside of the .Peek interface

.Peek reports are serialized in JSON format. They can be parsed efficiently and used anywhere you find it useful. You can for example render them in your own web interface, or parse them on a CI machine. .Peek lastest version will always update the older build reports to the newest format, ensuring your build reports stay relevant at any time.

3.1 Package Contents

Each .unitypackage downloaded from the AssetStore or from the [WellFired](#) website will have the same contents.

- **/WellFired/WellFired.Peek/Editor** Here you'll find all code related to the .Peek project

3.2 Installing

1. Import the .unitypackage into your unity project.

4.1 Your first build report

1. After installing .Peek, .Peek window is accessible through the menu **Window > DotPeek** in Unity Editor. Click on it to open .Peek interface : the list of build reports is empty :(
2. Build your project for one of the supported platforms : Windows, Linux, Mac OS, iOS or Android.
3. At the end of the build, .Peek should open by itself and display the new build report.

Congratulation ! You generated your first build report !

Tip: If the .Peek does not open by itself at the end of the build it could be you disabled the option in .Peek. See [Settings](#) for more info.



	Overview	Build Settings	Unused Assets	Used Assets
All Builds				
1/27/2018 @ 21:44:59				
Saturday, January 27, 2018				
Unity Version				2017.3.0f3
Build Duration				00 min. 00 sec.
Commit Id				Invalid
Platform				Standalone OSX
Build Size				65.60 MB
Number of assets in Resources folder				1
Assets Breakdown				
Textures			1.80 MB	2.70 %
Meshes			7.00 MB	10.70 %
Animations			0.0 KB	0.00 %
Sounds			0.0 KB	0.00 %
Shaders			557.8 KB	0.80 %
Other Assets			18.8 KB	0.00 %
Levels			35.1 KB	0.10 %
Scripts			641.8 KB	1.00 %
Included DLLs			10.90 MB	16.70 %
File Headers			17.3 KB	0.00 %

4.2 Next Step

In the coming pages, you'll learn more about .Peek's User Interface, and how your build reports are stored and managed.

.Peek Front Page

This is the first page displayed when you access .Peek through Unity Editor **Window > DotPeek**.

It is divided in three parts :

1. The top navigation bar that gives you access to the *Settings*.
2. The left panel listing the different build reports based on the time it was generated.

Tip: You can right click on one of them to display a contextual menu. You can either delete it, or reveal the location of the report. Clicking on **Ctrl/Command** while right clicking allows you to select several reports for deletion.

3. The central view that displays the build report you selected on the left, or the settings page.

1.
⚙️

Overview

Build Settings

Unused Assets

Used Assets

2.

All Builds

- 8/14/2018 @ 01:29:42
- 8/14/2018 @ 01:15:04

Delete
Reveal
- 8/14/2018 @ 01:13:28
- 8/13/2018 @ 17:09:15
- 8/13/2018 @ 17:06:06
- 8/13/2018 @ 16:58:23
- 8/13/2018 @ 16:55:19
- 8/13/2018 @ 16:51:15
- 8/13/2018 @ 16:46:15
- 8/13/2018 @ 16:45:34
- 8/13/2018 @ 16:34:11
- 8/13/2018 @ 16:09:42
- 8/12/2018 @ 17:49:45
- 5/30/2018 @ 16:05:31
- 5/23/2018 @ 17:31:17

3.

Monday, August 13, 2018

Unity Version	2018.2.1f1
Build Duration	02 min. 12 sec.
Commit Id	Unknown
Platform	IOS
Build Size	638.60 MB
Number of assets in Resources folder	1

Assets Breakdown

Textures	3.00 MB	0.50 %
Meshes	7.00 MB	1.10 %
Animations	0.0 KB	0.00 %
Sounds	0.0 KB	0.00 %
Shaders	951.0 KB	0.10 %
Other Assets	19.6 KB	0.00 %
Levels	35.3 KB	0.00 %
Scripts	684.2 KB	0.10 %

Build Report Panels

After you select a build report on the left side of the UI, it will be loaded and displayed through 4 panels :

1. *Overview*
2. *Build Settings*
3. *Unused Assets*
4. *Used Assets*

6.1 Overview

The overview page gives you some general information about your build.



	Overview	Build Settings	Unused Assets	Used Assets	
All Builds 5/30/2018 @ 22:05:31 5/23/2018 @ 23:31:17 5/23/2018 @ 19:54:59 5/23/2018 @ 19:52:21 5/23/2018 @ 19:50:58 5/23/2018 @ 19:30:43 5/23/2018 @ 17:34:18 5/11/2018 @ 05:48:23 5/11/2018 @ 05:36:10 5/11/2018 @ 05:26:31 5/11/2018 @ 05:11:39 5/11/2018 @ 05:04:10 5/11/2018 @ 04:57:13 4/25/2018 @ 00:13:01 4/24/2018 @ 18:45:42	<h2>Wednesday, May 23, 2018</h2>				
	Unity Version	2017.3.1p3			
	Build Duration	00 min. 51 sec.			
	Commit Id	d64d74e-unsync			
	Platform	IOS			
	Build Size	604.80 MB			
	Number of assets in Resources folder	1			
	<h3>Assets Breakdown</h3>				
	Textures	3.00 MB	0.50 %		
	Meshes	7.00 MB	1.20 %		
	Animations	0.0 KB	0.00 %		
	Sounds	0.0 KB	0.00 %		
Shaders	884.9 KB	0.10 %			
Other Assets	20.0 KB	0.00 %			
Levels	35.1 KB	0.00 %			
Scripts	395.3 KB	0.10 %			

Unity Version The Unity version used.

Build Duration The duration Unity took to make the build.

Tip: Note that .Peek records the time based on callbacks triggered by Unity Editor before and after it produced a build. Since several scripts in your project may make use of these callbacks, Unity provides a way to order them. .Peek by default will try to receive the prebuild callback as early as possible, and the postbuild callback as late as possible. You can programmatically change these settings, see XXXXXXXXXX

Commit Id If your Unity project is linked to a VCS repository, then the commit id at build time will be saved and display here. Note that .Peek supports only GIT and SVN at the moment. Windows users may need to ensure these VCS are installed on the command line.

Platform The platform the build was done for. For the moment .Peek support generating build reports only for MacOS, Linux, Windows, Android and iOS.

Build Size The size of the build before it is packaged to be run on the final platform.

Warning: The size reported here and the final size of your build once it is fully packaged for the final platform can differ a lot. For example, the size of the final IPA produced for IOS is far different from the sum of the assets copied to the XCode project before it is compiled.

Size of Resources This is the size of the assets added to Resources folders. It is recommended to keep this size low as it directly impacts the startup time of your game. You can get more information about this in this [Unity article](#).

Assets Breakdown This indicates you the size a category of assets occupies in the build.

6.2 Build Settings

The build settings panel shows you which settings were used for the platform you were building for. The settings displayed are relevant to the platform. So if a setting is specific to Android, like the target API Level, then it won't be displayed if the platform targeted was Standalone.

The screenshot shows the Build Settings panel. On the left is a sidebar titled "All Builds" with a list of build timestamps. The main panel has four tabs: "Overview", "Build Settings", "Unused Assets", and "Used Assets". The "Build Settings" tab is active, displaying a settings table.

Debug And Crash Reporting	
Unhandled Exception	Crash
Log Obj C Exception	True
Enable Crash Report	False
Rendering	
Color Space	Gamma
Auto Graphics API	True
APIs	Metal Open GLES2
Force Hard Shadow On Metal	False
Static Batching	True
Dynamic Batching	True
GPU Skinning	False

6.3 Used and Unused Assets

These two panels give you a detailed list of the assets included or not in your build.

Tip: You can click on any asset. If the asset is located in your project **Assets** folder, the asset will be selected in your project panel.

Overview

Build Settings

Unused Assets

Used Assets

All Builds

- 5/30/2018 @ 22:05:31
- 5/23/2018 @ 23:31:17
- 5/23/2018 @ 19:54:59
- 5/23/2018 @ 19:52:21
- 5/23/2018 @ 19:50:58
- 5/23/2018 @ 19:30:43
- 5/23/2018 @ 17:34:18
- 5/11/2018 @ 05:48:23
- 5/11/2018 @ 05:36:10
- 5/11/2018 @ 05:26:31
- 5/11/2018 @ 05:11:39
- 5/11/2018 @ 05:04:10
- 5/11/2018 @ 04:57:13
- 4/25/2018 @ 00:13:01
- 4/24/2018 @ 18:45:42

Total Size

9.66 MB

Type to filter...

Asset Path	Imported Size	Raw Size	Percentage
Assets/Models/Car/fbx.fbx	7.00 MB	10.90 MB	10.70 %
Assets/Models/Car/Materials/Textures/Parts.	682.8 KB	1.81 MB	1.00 %
Resources/unity_built_in_extra	559.4 KB	N/A	0.80 %
Assets/Models/Car/Materials/Textures/Licen:	341.5 KB	54.3 KB	0.50 %
Assets/Models/Car/Materials/Textures/Lighs	170.8 KB	4.8 KB	0.30 %
Assets/Textures/thibault.jpg	170.8 KB	173.6 KB	0.30 %
Assets/Models/Car/Materials/Wheel_Mats/Nc	85.5 KB	86.1 KB	0.10 %
Assets/Models/Car/Flares/Textures/Indicator	85.5 KB	25.9 KB	0.10 %
Assets/Models/Car/Flares/Textures/TailFlare.	85.5 KB	25.6 KB	0.10 %
Assets/Models/Car/Flares/Textures/FrontFlar	85.5 KB	26.8 KB	0.10 %
Assets/Models/Car/Flares/Textures/BrakeFlai	85.5 KB	47.8 KB	0.10 %
Assets/Textures/texture.png	78.7 KB	238.8 KB	0.10 %
Assets/Models/Car/Materials/Wheel_Mats/Di	42.8 KB	0.7 KB	0.10 %
Assets/Models/Car/Materials/Wheel_Mats/Sp	42.8 KB	20.2 KB	0.10 %

6.3.1 Assets Information

The **Total Size** on top of the list is the sum of all the assets *Imported Size*.

Imported Size This is the size of the asset after it was imported in the player. For example, if the asset is a .psd file and is imported under the format ETC 4 bits in the player, then Imported Size will display the size of the later.

Raw Size This is the original size of the asset imported in Unity. This is the size that directly impacts your VCS repository size.

Percentage This is the percentage that represents the *Imported Size* over the total size of the build you can read on the *Overview* panel.

Note: Note that you can re-order the assets by clicking on the columns header.

6.3.2 Filtering

Simple search

You can filter which assets should be listed by entering a value in the search field. .Peek will list all the assets which path contains all of the word you input. For example, if you enter `.png car`, all the asset with a path containing `.png` and `car` will be displayed. **The search field is case insensitive.**

Typed search

You can also filter the assets by type. You simply need to input `t:[asset type]`. This can be used on top of the simple search, like for example `: car t:texture renault`. The values accepted are:

t:texture Displays files with extention : `.png, .tga, .psd, .tif, .jpg, .jpeg, .gif, .bmp, .iff, .pict`

t:audio Displays files with extention : `.mp3, .ogg, .wav, .aiff, .aif, .mod, .it, .s3m, .xm`


t:model Displays files with extention : `.fbx, .dae, .3ds, .dxf, .obj, .skp, .ma, .mb, .max, .c4d, .blend`

t:plugin Displays files with extention : `.dll, .jar, .so, .aar, .a`

t:shader Displays files with extention : `.shader, .cginc`

These extensions are based on what is supported in Unity. Don't hesitate to create a pull request or open an issue on Github if you find out relevant to add an extension here.

Settings

The settings page is accessible from the **Settings** button on the top of .Peek window. Some settings are personal settings, which means they are saved in the folder `[Unity Project]/.wellfired` which should be ignored from your VCS. Some settings are team shared and are saved in `[Unity Project]/WellFired` which should be added to your VCS repository, they are marked with .

Tip: Note that these settings can be controlled programmatically if you want to enforce .Peek behaviour on different computers (for Continuous Integration for example).



Auto generate report

Automatically open

Build reports location



Track VCS version


Activate .Peek Logs

Auto generate report If this option is turned on, .Peek will generate a report at the end of a build. You have the possibility to turn off this option and programmatically ask DotPeek to generate a report. This is a personal


setting.

Automatically open When turned on, .Peek window will automatically open at the end of a build and display the new generated report. This is a personal setting.

Build reports location This is the location where reports are saved.

- Relative paths are team shared . Therefore, if you input “*../BuildReports*” as report location, then all of your team members will have their build reports saved in *[Unity Project]/../BuildReports*.
- Absolute path are personal, then it will not affect other team members reports location. Note that if you indicate a location inside the Unity project, it will be automatically converted to a relative location.

Tip: Different Unity projects can save their reports in the same folder. Indeed, .Peek will create a subfolder with the GUID it assigned to your Unity project. The GUID is a team shared setting.

Track VCS version  If enable and SVN or GIT is installed on your computer, the version of the commit you are building will appear in the report.

If not all your modification are committed, then *-unsync* will be added to the version (such as : **Commit Id** *4baa424-unsync*).

Windows users may need to ensure these VCS are installed on the command line. Computers not supporting it will simply display **Commit Id** *unknow* no matter this option is enabled or not.

Activate .Peek logs This will activate logs when .Peek is running. This will decrease .Peek performances and should be activated only if necessary to debug an issue you are meeting with .Peek.

Team icon on this page is provided for free by Icons8

8.1 Settings

All the settings you can access through .Peek UI are available through code and can be read or overwritten.

To do so :

1. Add the required using :

```
using WellFired.Peek.Application.Unity.Editor;
```

2. Access and modify your .Peek personal options (stored on your computer only, see *Settings* for more information) :

```
DotPeek.Storage.PersonalOptions.BuildReportPath = "../DotPeekReports";  
DotPeek.Storage.PersonalOptions.AutomaticallyShowReportAfterBuild = false;
```

3. Access and modify your .Peek team-shared options :

```
DotPeek.Storage.TeamOptions.TrackVCSVersion = false;
```

4. Save your changes to ensure .Peek can access updated settings :

```
DotPeek.Storage.Save();
```

Warning: Saving step cannot be omitted. Indeed, it will ensure settings that were modified are written to the disk and are accessible to the whole .Peek application.

Tip: .Peek Storage is thread safe, so the thread you are accessing it from does not matter.

8.2 Callbacks Order

DotPeek build generation is driven by two callbacks automatically called by Unity when building :

- `IPreprocessBuild.OnPreprocessBuild` which happens just before the build starts.

When it is called, .Peek will start counting the time used to build and get the VCS commit ID of your project. Note that if beforehand some files were not committed, .Peek will add `-unsync` behind the commit ID.

- `IPostprocessBuild.OnPostprocessBuild` which is called once the build is done.

When it is called, the time elapsed to make the build is saved and the report generated. Note that at that moment, all the files that are in the project, but were not added to the build are considered as unused assets.

Different modules in your project may use these callbacks, and Unity will decide which one to call first based on `IPreprocessBuild.callbackOrder` and `IPostprocessBuild.callbackOrder`

.Peek by default try to be called as early as possible and as late as possible, but you can programmatically change this behaviour by :

1. Adding the required using :

```
using WellFired.PEEK.Application.Unity.Editor;
```

2. Modifying the callback order :

```
DotPeek.Storage.TeamOptions.PrebuildCallbackOrder = int.MinValue + 1;  
DotPeek.Storage.TeamOptions.PostbuildCallbackOrder = int.MaxValue - 1;
```

3. Saving your changes to ensure .Peek can access updated settings :

```
DotPeek.Storage.Save();
```

8.3 Manually start .Peek metric counters

There is some situations you might not want to rely on the `IPreprocessBuild.OnPreprocessBuild`.

Indeed, when this Unity callback is called, .Peek will check your VCS status and starts counting how much time your build takes.

If ever your build pipeline is based on a custom script that executes some operations before Unity `BuildPipeline.BuildPlayer` is called, then when `OnPreprocessBuild` is called, your VCS may already have some local changes, or your build may already have taken several minutes, which will lead .Peek to display wrong information at the end of the build.

To avoid this situation you can force .Peek to start tracking these information before `BuildPipeline.BuildPlayer` is called. Here the steps :

1. Add the required using :

```
using WellFired.PEEK.Application.Unity.Editor;
```

2. Start .Peek session with the Unity build target you are building for :

```
DotPeek.StartSession(BuildTarget.Android);
```

That's it !

Continuous Integration

.Peek was designed to provide a quick feedback to Unity users about the status of their project, but also to be nicely integrated on any Continuous Integration pipeline.

Through the *Programmatic Control* of DotPeek, you can ensure that your CI run .DotPeek with consistent settings.

Hereunder are more ways of optimizing .Peek usage on a CI machine.

9.1 Getting informed when build report is ready

For optimized performances, .Peek generate your build report on threads. It means that when you call Unity on the command line, you should not do it with the parameter *-quit*. This would quit the editor as soon as the Unity editor main thread finished the task it was assigned on the command line.

To allow you to quit unity after the report was generated, .Peek provides you the interface *IDotPeekListener*. You first need to implement it :

```
private class DotPeekListener : IDotPeekListener
{
    public void DoBuildReportGenerated(string reportAbsolutePath)
    {
        //do here what you want after the build is generated
    }
}
```

And then to provide this listener to .Peek :

```
DotPeek.Listener = new DotPeekListener();
```

Below is a whole functional usage example :

```
using UnityEditor;
using WellFired.PEEK.Application.Unity.Editor;
```

(continues on next page)

(continued from previous page)

```
[InitializeOnLoad]
public static class DotPeekInitializer
{
    static DotPeekInitializer()
    {
        DotPeek.Listener = new DotPeekListener();
    }

    private class DotPeekListener : IDotPeekListener
    {
        public void DoBuildReportGenerated(string reportAbsolutePath)
        {
            //Will quit Unity with error code 0, indicating no error happened.
            EditorApplication.Exit(0);
        }
    }
}
```

The parameter `reportAbsolutePath` is the path to your freshly generated report. It can be useful if you want to send the report to an other machine for example (website server, NAS, ...)

9.2 VCS Version

If `.Peek` option is enabled and your project versions are tracked through GIT or SVN, the commit id at build time will be automatically saved with your generated report. This is done through the terminal of your machine.

If for any reason `.Peek` cannot access the terminal of the machine it is running on, you can provide an implementation of `IVCS` and return to `.Peek` the value you want :

```
private interface IVCS
{
    string GetCommitId();
}
```

This can be useful for example if the access to the commit id can be done only through the parameters passed to Unity through the command line. Here an illustration :

The command being called :

```
/Applications/Unity/Unity.app/Contents/MacOS/Unity -VCS aa2e32w -batchmode -
↳executeMethod MyEditorScript.PerformBuild
```

The implementation of the static function being called in Unity :

```
using UnityEditor;
class MyEditorScript
{
    static void PerformBuild ()
    {
        DotPeek.CustomVCS = new DotPeekVCS();

        string[] scenes = { "Assets/MyScene.unity" };
        BuildPipeline.BuildPlayer(scenes, ...);
    }
}
```

with DotPeekVCS implemented this way :

```
private class DotPeekVCS : IVCS
{
    public string GetCommitId()
    {
        var args = Environment.GetCommandLineArgs().ToList();
        var optionPosition = args.IndexOf("-VCS");
        var vcsCommitId = args[optionPosition + 1];

        return vcsCommitId;
    }
}
```


10.1 Classes

10.1.1 BuildReportHelper

Namespace: WellFired.Peek

Description

Public Static Methods

async Task< BuildReport >	<i>GenerateAndSaveReport</i> (IBuildReportGenerator buildReportGenerator, IBuildReportStorage reportStorage, string savingPath, string commitId, Stopwatch stopwatch)
---------------------------	---

Breakdown

- async Task< BuildReport > **GenerateAndSaveReport** (IBuildReportGenerator buildReportGenerator, IBuildReportStorage reportStorage, string savingPath, string commitId, Stopwatch stopwatch)

10.1.2 DotPeekSession

Namespace: WellFired.Peek

Implements: WellFired.Peek.Application.IDotPeekSession

Description

Properties

IDotPeekSessionListener	<i>Listener</i> { get; set; }
-------------------------	-------------------------------

Public Methods

	<i>DotPeekSession</i> (IVCS vcs, IStorage storage, IPlatformTools platformTools, IBuildReportGenerator buildReportGenerator, IBuildReportStorage buildReportStorage, IWindowLauncher windowLauncher)
void	<i>PreProcessBuild</i> (Platform platform)
void	<i>PostProcessScene</i> (string scenePath)
void	<i>PostProcessBuild</i> ()
void	<i>OpenWindow</i> ()
async Task	<i>PostProcessBuildTask</i> ()

Breakdown

- IDotPeekSessionListener **Listener** { get; set; }
- **DotPeekSession** (IVCS vcs, IStorage storage, IPlatformTools platformTools, IBuildReportGenerator buildReportGenerator, IBuildReportStorage buildReportStorage, IWindowLauncher windowLauncher)
- void **PreProcessBuild** (Platform platform)
- void **PostProcessScene** (string scenePath)
- void **PostProcessBuild** ()
- void **OpenWindow** ()
- async Task **PostProcessBuildTask** ()

10.1.3 BuildPostProcessor

Namespace: WellFired.PEEK.Application.Unity.Editor

Description

Properties

int	<i>callbackOrder</i> { get; set; }
-----	------------------------------------

Public Methods

void	<i>OnPostprocessBuild</i> (BuildTarget target, string path)
------	---

Breakdown

- int **callbackOrder** { get; set; }
- void **OnPostprocessBuild** (BuildTarget target, string path)

10.1.4 BuildPreProcessor

Namespace: WellFired.Peek.Application.Unity.Editor

Description

Properties

int	<i>callbackOrder</i> { get; set; }
-----	------------------------------------

Public Methods

void	<i>OnPreprocessBuild</i> (BuildTarget target, string path)
------	--

Breakdown

- int **callbackOrder** { get; set; }
- void **OnPreprocessBuild** (BuildTarget target, string path)

10.1.5 OpenDotPeek

Namespace: WellFired.Peek.Application.Unity.Editor

Description

Public Static Methods

void	<i>Launch</i> ()
------	------------------

Breakdown

- void **Launch** ()

10.1.6 SceneProcessor

Namespace: WellFired.Peek.Application.Unity.Editor

Description

Public Static Methods

void	<i>PostProcessSceneAttribute</i> ()
------	--------------------------------------

Breakdown

- void **PostProcessSceneAttribute** ()

10.1.7 DotPeek

Namespace: WellFired.Peek.Application.Unity

Implements: WellFired.Peek.Application.IDotPeekSessionListener

Description

This is a public wrapper around `.:ref:Peek<namespacewellfired_peek>` application. It gives access to different utilities allowing a total control of `.:ref:Peek<namespacewellfired_peek>`.

Properties

IDotPeekListener	<i>Listener</i> { get; set; }
IVCS	<i>CustomVCS</i> { get; set; }
Storage	<i>Storage</i> { get; set; }

public-static-attrib

bool	<i>SessionStarted</i>
IDotPeekSession	<i>CurrentSession</i>

Public Static Methods

void	<i>StartSession</i> (BuildTarget target)
void	<i>EndSession</i> ()
void	<i>OpenWindow</i> ()

Public Methods

void	<i>DoBuildReportGenerated</i> (string reportAbsolutePath)
------	---

Breakdown

- IDotPeekListener **Listener** { get; set; }

Description

Give access to `.:ref:Peek<namespacewellfired_peek>` callbacks, like when the report is generated and where it is stored for example.

- IVCS **CustomVCS** { get; set; }

Description

Allows to provide a custom commit id to `.:ref:Peek<namespacewellfired_peek>` when it is generating the build report.

- Storage **Storage** { get; set; }

Description

Allows to read or modify `.:ref:Peek<namespacewellfired_peek>` settings on the disk.

- bool **SessionStarted**

Description

Returns true if a session was started already.

- IDotPeekSession **CurrentSession**

Description

Returns the current IDotPeekSession.

- void **StartSession** (BuildTarget target)

Description

Creates a new IDotPeekSession that will receive the different callbacks from the game engine when build is being processed. When a new session is started, then the previous one is not referenced anymore.

- void **EndSession** ()

Description

Finishes a IDotPeekSession.

- void **OpenWindow** ()

Description

Open the *DotPeek* window in Unity.

- void **DoBuildReportGenerated** (string reportAbsolutePath)

Description

This is called after the build report was generated and saved on the disk.

10.1.8 WindowLauncher

Namespace: WellFired.Peek.Application.Unity

Implements: WellFired.Peek.Application.IWindowLauncher

Description

Public Methods

void	<i>Launch</i> (string companyName, string applicationName, string applicationTitle)
------	---

Breakdown

- void **Launch** (string companyName, string applicationName, string applicationTitle)

10.1.9 GIT

Namespace: *WellFired.PEEK.Application.VCS*

Implements: WellFired.PEEK.Application.VCS.IVCS

Description

Public Methods

string	<i>GetCommitId</i> ()
--------	-----------------------

Breakdown

- string **GetCommitId** ()

Description

Provide the current commit id.

10.1.10 GITException

Namespace: *WellFired.PEEK.Application.VCS*

Description

Public Properties

override string	<i>Message</i>
-----------------	----------------

Public Methods

	<i>GITException</i> (string command, string error)
--	--

Breakdown

- override string **Message**
- **GITException** (string command, string error)

10.1.11 GITInspector

Namespace: *WellFired.PEEK.Application.VCS*

Implements: WellFired.PEEK.Application.VCS.IVCSInspector

Description

Public Methods

bool	<i>IsRepository</i> (string location)
<i>RepositoryInfo</i>	<i>GetRepositoryInfo</i> (string location)

Breakdown

- bool **IsRepository** (string location)

Description

Detect if this IVCSInspector is compatible with the *VCS* used at the location specified.

Parameters

location

- *RepositoryInfo* **GetRepositoryInfo** (string location)

Description

Get information about the state of the repository at the location specified.

Parameters

location

10.1.12 NoVCS

Namespace: WellFired.PEEK.Application

Implements: WellFired.PEEK.Application.VCS.IVCS

Description

This is used when no *VCS* used by the project could be detected.

Public Methods

string	<i>GetCommitId</i> ()
--------	-----------------------

Breakdown

- string **GetCommitId** ()

Description

Provide the current commit id.

10.1.13 OSEnvironment

Namespace: WellFired.Peek.Application

Description

Public Static Methods

string	<i>RunCommand</i> (string command, string args, out string retErrors)
string	<i>RunCommand</i> (string command, string args, string workingDirectoy, out string retErrors)
string	<i>GetWorkingDirectory</i> ()

Breakdown

- string **RunCommand** (string command, string args, out string retErrors)
- string **RunCommand** (string command, string args, string workingDirectoy, out string retErrors)
- string **GetWorkingDirectory** ()

10.1.14 RepositoryInfo

Namespace: WellFired.Peek.Application

Description

Info about the status of a local repository

Public Properties

string	<i>CommitID</i>
RepositoryStatus	<i>Status</i>

Breakdown

- string **CommitID**

Description

The commit Id checked out.

- RepositoryStatus **Status**

Description

Indicates if the local repository is synchronized with the remote one or not.

10.1.15 SVN

Namespace: *WellFired.PEEK.Application.VCS*

Implements: WellFired.PEEK.Application.VCS.IVCS

Description

Public Methods

string	<i>GetCommitId</i> ()
--------	-----------------------

Breakdown

- string **GetCommitId** ()

Description

Provide the current commit id.

10.1.16 SVNException

Namespace: *WellFired.PEEK.Application.VCS*

Description

Public Properties

override string	<i>Message</i>
-----------------	----------------

Public Methods

	<i>SVNException</i> (string command, string error)
--	--

Breakdown

- override string **Message**
- **SVNException** (string command, string error)

10.1.17 SVNInspector

Namespace: *WellFired.PEEK.Application.VCS*

Implements: WellFired.PEEK.Application.VCS.IVCSInspector

Description

Public Methods

bool	<i>IsRepository</i> (string location)
<i>RepositoryInfo</i>	<i>GetRepositoryInfo</i> (string location)

Breakdown

- bool **IsRepository** (string location)

Description

Detect if this IVCSInspector is compatible with the *VCS* used at the location specified.

Parameters

location

- *RepositoryInfo* **GetRepositoryInfo** (string location)

Description

Get information about the state of the repository at the location specified.

Parameters

location

10.1.18 VCSUtils

Namespace: WellFired.PEEK.Application

Description

Public Static Methods

IVCS	<i>GetVCSInUse</i> ()
------	------------------------

Breakdown

- **IVCS GetVCSInUse ()**

Description

Detect which *VCS* is being used at the location the dll is being executed and return the relevant IVCS.

10.1.19 Constants

Namespace: WellFired.Peek

Description

Public Properties

const string	<i>ApplicationName</i>
const string	<i>CompanyName</i>
const string	<i>ApplicationTitle</i>
const string	<i>BuildReportExtention</i>

Breakdown

- const string **ApplicationName**
- const string **CompanyName**
- const string **ApplicationTitle**
- const string **BuildReportExtention**

10.1.20 FileExtensions

Namespace: WellFired.Peek

Description

public-static-attrib

readonly string[]	<i>Animation</i>
readonly string[]	<i>Texture</i>
readonly string[]	<i>Model</i>
readonly string[]	<i>Prefab</i>
readonly string[]	<i>Asset</i>
readonly string[]	<i>Material</i>
readonly string[]	<i>Audio</i>
readonly string[]	<i>Plugin</i>
readonly string[]	<i>Script</i>
readonly string[]	<i>Shader</i>
readonly string[]	<i>Scene</i>
readonly string[]	<i>Ignored</i>

Public Static Methods

IEnumerable< string >	<i>GetExtensions</i> (string value)
-----------------------	---------------------------------------

Breakdown

- readonly string[] **Audio**
- readonly string[] **Animation**
- readonly string[] **Model**
- readonly string[] **Prefab**
- readonly string[] **Asset**
- readonly string[] **Material**
- readonly string[] **Texture**
- readonly string[] **Plugin**
- readonly string[] **Script**
- readonly string[] **Shader**
- readonly string[] **Scene**
- readonly string[] **Ignored**
- IEnumerable< string > **GetExtensions** (string value)

10.1.21 FileSizeComparer

Namespace: WellFired.Peek

Description**Public Methods**

int	<i>Compare</i> (FileSize x, FileSize y)
-----	---

Breakdown

- int **Compare** (FileSize x, FileSize y)

10.2 Interfaces

10.3 Namespaces

10.3.1 VCS

Namespace: WellFired.Peek

Description**Breakdown**

10.3.2 Data

Namespace: WellFired

Description**Breakdown**

10.3.3 Model

Namespace: WellFired

Description**Breakdown**

10.3.4 ProjectSettings

Namespace: WellFired.Peek

Description

Breakdown

10.4 Enums

10.4.1 RepositoryStatus

Namespace: *WellFired.PEEK.Application.VCS*

Description

Status of the repository

NotSync	The repository contains modifications not pushed to the remote repository.
SyncToCommit	The checked out version is not the latest one, but files does not contains any modification.
SyncToHead	The checked out version is the latest one, and files does not contains any modification.

10.4.2 Category

Namespace: *WellFired.PEEK.Data*

Description

Undefined
Textures
Meshes
Animations
Sounds
Shaders
OtherAssets
Levels
Scripts
IncludedDLLs
FileHeaders
StreamingAssets
Settings
IndividualBuildReport
NoBuildReports
Overview
UsedAssets
UnusedAssets
BuildSettings
HasBuildReport
Android
IOS
WindowsStandalone
WindowsStandalone_64
StandaloneOSX
MacStandalone_x86
MacStandalone_x86_64
LinuxStandalone
LinuxStandalone_64
LinuxStandaloneUniversal

10.4.3 PreprocessorOrigin

Namespace: *WellFired.PEEK.Model*

Description

Log
Editor

10.4.4 VertexCompression

Namespace: *WellFired.PEEK.Model.ProjectSettings*

Description

Nothing
Position
Normal
Color
Uv0
Uv1
Uv2
Uv3
Tangent
Everything
SlowAndSafe
Fast
ArMv7
Arm64
Universal
IPhone
IPad
IPhoneIPad
Mono
IL2CPP
Net2
Net2Subset
Net4_6
Net35
Net46
Metal
OpenGLES3
OpenGLES2
Vulkan
Direct3D_11
Direct3D_9
Direct3D_12
OpenGLCore
Gamma
Linear
Crash
SilentExit
Disabled
StripAssemblies
StripByteCode
Internal
External
Automatic
External
Internal
FAT
ARMv7
x86