## **Unity Supports**

Put the ZeroFormatter.dll and ZeroFormatter.Interfaces.dll, modify Edit -> Project Settings -> Player -> Optimization -> Api Compatibillity Level to .NET 2.0 or higher.

[](https://cloud.githubusercontent.com/assets/46207/20293228/d3a4add2-ab37-11e6-878b-24daad4dc2c1.png)

ZeroFormatter.Unity works on all platforms(PC, Android, iOS, etc...). But it can 'not' use dynamic serializer generation due to IL2CPP issue. But pre code generate helps it. Code Generator is located in packages\ZeroFormatter.Interfaces.\*.\*.\*\tools\zfc.exe. zfc is using [Roslyn](https://github.com/dotnet/roslyn) so analyze source code, pass the target csproj.

zfc arguments help:

-i, --input=VALUE [required]Input path of analyze csproj

-o, --output=VALUE [required]Output path(file) or directory base(in separated mode)

-s, --separate [optional, default=false]Output files are separated

-u, --unuseunityattr [optional, default=false]Unuse UnityEngine's RuntimeInitializeOnLoadMethodAttribute on ZeroFormatterInitializer

-t, --customtypes=VALUE [optional, default=empty]comma separated allows custom types

-c, --conditionalsymbol=VALUE [optional, default=empty]conditional compiler symbol

-r, --resolvername=VALUE [optional, default=DefaultResolver]Register CustomSerializer target

-d, --disallowinternaltype [optional, default=false]Don't generate internal type

-e, --propertyenumonly [optional, default=false]Generate only property enum type only

-m, --disallowinmetadata [optional, default=false]Don't generate in metadata type

-g, --gencomparekeyonly [optional, default=false]Don't generate in EnumEqualityComparer except dictionary key

-n, --namespace=VALUE [optional, default=ZeroFormatter]Set namespace root name

-f, --forcedefaultresolver [optional, default=false]Force use DefaultResolver

Note: Some options is important for reduce code generation size and startup speed on IL2CPP, especially -f is recommend if you use only DefaultResolver.

// Simple Case:

zfc.exe -i "..\src\Sandbox.Shared.csproj" -o "ZeroFormatterGenerated.cs"

// with t, c

zfc.exe -i "..\src\Sandbox.Shared.csproj" -o "..\unity\ZfcCompiled\ZeroFormatterGenerated.cs" -t "System.Uri" -c "UNITY"

// -s

zfc.exe -i "..\src\Sandbox.Shared.csproj" -s -o "..\unity\ZfcCompiled\"

zfc.exe can setup on csproj's PreBuildEvent(useful to generate file path under self project) or PostBuildEvent(useful to generate file path is another project).

Note: zfc.exe is currently only run on Windows. It is .NET Core's [Roslyn](https://github.com/dotnet/roslyn) workspace API limitation but I want to implements to all platforms...

Generated formatters must need to register on Startup. By default, zfc generate automatic register code on RuntimeInitializeOnLoad timing.

For Unity Unit Tests, the generated formatters must be registered in the SetUp method:

[SetUp]

public void RegisterZeroFormatter()

{

ZeroFormatterInitializer.Register();

}

ZeroFormatter can not serialize Unity native types by default but you can make custom formatter by define pseudo type. For example create Vector2 to ZeroFormatter target.

#if INCLUDE\_ONLY\_CODE\_GENERATION

using ZeroFormatter;

namespace UnityEngine

{

[ZeroFormattable]

public struct Vector2

{

[Index(0)]

public float x;

[Index(1)]

public float y;

public Vector2(float x, float y)

{

this.x = x;

this.y = y;

}

}

}

#endif

INCLUDE\_ONLY\_CODE\_GENERATION is special symbol of zfc, include generator target but does not include compile.

If you encounter InvalidOperationException such as

InvalidOperationException: Type is not supported, please register Vector3[]

It means not generated/registered type. Especially collections are not automatically registered if they are not included in the property. You can register manually such as Formatter.RegisterArray<UnityEngine.Vector3>() or create hint type for zfc.

using ZeroFormatter;

namespace ZfcHint

{

[ZeroFormattable]

public class TypeHint

{

// zfc analyzes UnityEngine.Vector3[] type and register it.

[Index(0)]

public UnityEngine.Vector3[] Hint1;

}

}