

# Namespace Cobilas

## Classes

### [TypeUtilitarian](#)

Utility static class to obtain type or assembly.

## Structs

### [Interrupter](#)

Represents a list of switches.

# Cobilas Core

## Description

Cobilas Core Net4x is a utility library for CSharp.

## Json

(namespace:Cobilas.IO.Serialization.Json)

Only present in the NuGet version.

The static class `Json` grants static read and write functions.

## JsonContractResolver

Used by `JsonSerializer` to resolve a `JsonContract` for a given `Type`. Furthermore, `JsonContractResolver` determines how the fields of an `Object` will be serialized.

## ATLF(Arquivo de tradução de leitura facil)

ATLF (Easy to Read Translation File) can be used to create and load translations for apps.

```
#>Header
```

```
The use of the header is not mandatory.<#
```

```
#! version:/*std:1.0*/
```

```
#! encoding:/*utf-8*/
```

```
#> Comment <#
```

```
#> ATLF format(1.0) <#
```

```
#> Uni-line marking <#
```

```
#! Tag1:/*value1*/
```

```
#> Multi-line marking <#
```

```
#! Tag2:/*
```

```
value1
```

```
value2
```

```
value3
```

```
value4
```

```
*/
```

## How to read ATLF

```
static void Main(string[] args) {  
    using ATLFReader reader = ATLFReader.Create(@"C:\folder1\file.txt");  
    reader.Reader();  
}
```

```

    Console.WriteLine($"tag.value.1:{reader.GetTag("tag.value.1")}");
    Console.WriteLine($"tag.value.2:{reader.GetTag("tag.value.2")}");
    Console.WriteLine($"tag.value.3:{reader.GetTag("tag.value.3")}");
}

```

The other reading functions.

- The `ATLFNode[]:ATLFReader.GetHeader()` function allows you to get the header tags.
- The `ATLFNode[]:ATLFReader.GetAllComments()` function allows you to get all comments. The `ATLFNode[]:ATLFReader.GetTagGroup(string path)` function allows you to obtain tags that belong to the same path.

```

/*C:\folder1\file.txt
* #! version:/*std:1.0* /
* #! encoding:/*utf-8* /
*
* #! tag.value.cop1:/*value1* /
* #! tag.value.map.cop1:/*value1* /
* #! tag.value.map.cop2:/*value1* /
* #! tag.value.cop2:/*value1* /
* #! tag.value.cop3:/*value1* /
*/
static void Main(string[] args) {
    using ATLFReader reader = ATLFReader.Create(@"C:\folder1\file.txt");
    reader.Reader();
    foreach(var item in reader.GetTagGroup("tag.value.map"))
        Console.WriteLine(item);
}

```

## How to write ATLF

```

static void Main(string[] args) {
    using ATLFWriter writer = ATLFWriter.Create(File.OpenWrite(@"C:\folder1\file.txt"));
    writer.WriteHeader();//The header is not mandatory but if you add a header, call this
function first.
    writer.WriteComment("my tag1");
    writer.WriteNode("tag1", "value1");
    writer.WriteWhitespace("\r\n");//This function is called automatically when the `Indent`
property is `true`. By default the `Indent` property is `true`.
    writer.WriteComment("my tag2");
    writer.WriteNode("tag2", "value2");
    writer.WriteWhitespace(2, "\r\n");//This function is called automatically when the
`Indent` property is `true`. By default the `Indent` property is `true`.
    writer.WriteComment("my tag3");
}

```

```
writer.WriteNode("tag3", "value3");  
}
```

## Encoders and decoders

Regarding encoders and decoders, ATLF allows the creation of customized encoders and decoders. To use a custom encoder or decoder, assign a version to your custom encoder or decoder using the `Version` property and then assign the version of the custom encoder or decoder in the `TargetVersion` property of the `ATLFWriter` and `ATLFReader` classes.

### Creating a custom encoding class

To create a custom encoding class, the class must inherit the `ATLFVS10Encoding` class.

### Creating a custom decoding class

To create a custom decoding class, the class must inherit the `ATLFVS10Decoding` class.

## [Cobilas.Core.Net4x](#) is on nuget.org

To include the package, open the `.csproj` file and add it.

```
<ItemGroup>  
  <PackageReference Include="Cobilas.Core.Net4x" Version="1.6.1" />  
</ItemGroup>
```

Or use command line.

```
dotnet add package Cobilas.Core.Net4x --version 1.6.1
```

## [Cobilas.Core.Net4x](#) is on NPM

Include in npm package

```
"dependencies": {  
  "com.cobilas.unity.core.net4x": "1.6.1"  
}
```

Or use command line.

```
npm i com.cobilas.unity.core.net4x
```

# Namespace Cobilas.GodotEngine.Utility

## Classes

### [Coroutine](#)

This class represents a corrotine process.

### [CoroutineManager](#)

This class is responsible for managing all coroutines.

### [GDDirectory](#)

Represents a directory file.

### [GDFeature](#)

This class contains some Features pre-defined by the engine.

### [GDFile](#)

This class is a representation of a file.

### [GDFileBase](#)

This is a base class for other classes that represent files or directory files.

### [GDIONull](#)

This class is a representation of a null file.

### [Gizmos](#)

Gizmos are used to give visual debugging or setup aids in the Scene view.

### [NullNode](#)

A null representation of the Godot.Node class.

### [Randomico](#)

The class allows the creation of pseudo random numbers.

### [Screen](#)

Gets or changes game screen information.

## Structs

### [CustonResolutionList](#)

Stores custom resolutions.

### [DisplayInfo](#)

Contains information from a specific screen.

### [FixedRunTimeSecond](#)

This class represents a delay in seconds to methods that return [IEnumerator](#) and use the keyword Yield.

This class is performed in the [\\_PhysicsProcess\(float\)](#).

### [LastFixedRunTimeSecond](#)

This class represents a delay in seconds to methods that return [IEnumerator](#) and use the keyword Yield.

This class is performed in the [\\_PhysicsProcess\(float\)](#).

This class allows the corrotine to be called after the methods of updating the current scene.

### [LastRunTimeSecond](#)

This class represents a delay in seconds to methods that return [IEnumerator](#) and use the keyword Yield.

This class is performed in the [\\_Process\(float\)](#).

This class allows the corrotine to be called after the methods of updating the current scene.

### [Resolution](#)

Stores information about a screen resolution.

### [RunTimeSecond](#)

This class represents a delay in seconds to methods that return [IEnumerator](#) and use the keyword Yield.

This class is performed in the [\\_Process\(float\)](#).

## Interfaces

### [IYieldCoroutine](#)

A base interface for all Yield class.

### [IYieldFixedUpdate](#)

Yield Class to be excited in the [\\_PhysicsProcess\(float\)](#)

### [IYieldUpdate](#)

Yield Class to be excited in the [\\_Process\(float\)](#)

### [IYieldVolatile](#)

The IyieldVolatile interface allows the Yield class to change the type of process.

This interface allows you to change the type of update if the object will use the [\\_Process\(float\)](#) or [\\_PhysicsProcess\(float\)](#) [Coroutine](#) process.

## Enums

### [GDFileAttributes](#)

Represents the file attributes.

### [ScreenMode](#)

Represents screen modes.

# Cobilas Godot Utility

## Description

The package contains utility classes in csharp for godot engine(Godot3.5)

## RunTimeInitialization

(namespace: Cobilas.GodotEngine.Utility.Runtime)

The `RunTimeInitialization` class allows you to automate the `Project>Project Settings>AutoLoad` option.

To use the `RunTimeInitialization` class, you must create a class and make it inherit `RunTimeInitialization`.

```
using Cobilas.GodotEngine.Utility.Runtime;
//The name of the class is up to you.
public class RunTimeProcess : RunTimeInitialization {}
```

And remember to add the class that inherits `RunTimeInitialization` in `Project>Project Settings>AutoLoad` .

Remembering that the `RunTimeInitialization` class uses the virtual method `_Ready()` to perform the initialization of other classes.

And to initialize other classes along with the `RunTimeInitialization` class, the class must inherit the `Godot.Node` class or some class that inherits `Godot.Node` and use the `RunTimeInitializationClassAttribute` attribute.

```
using Godot;
using Cobilas.GodotEngine.Utility.Runtime;
[RunTimeInitializationClass]
public class ClassTest : Node {}
```

## RunTimeInitializationClass

```
/*
bootPriority: Represents the boot order
{ (enum Priority)values
    StartBefore,
    StartLater
}
name:The name of the object
subPriority: And the execution priority order.
*/
```



```
//RunTimeInitializationClassAttribute(string? name, Priority bootPriority =  
Priority.StartBefore, int subPriority = 0, bool lastBoot = false)  
[RunTimeInitializationClassAttribute(string?, [Priority:Priority.StartBefore],  
[int:0], [bool:false])]  
[RunTimeInitializationClass()]
```

## The [Cobilas Godot Utility](#) is on nuget.org

To include the package, open the `.csproj` file and add it.

```
<ItemGroup>  
  <PackageReference Include="Cobilas.Godot.Utility" Version="4.1.0" />  
</ItemGroup>
```

Or use command line.

```
dotnet add package Cobilas.Godot.Utility --version 4.1.0
```