



We use optional cookies to improve your experience on our websites, such as through social media connections, and to display personalized advertising based on your online activity. If you reject optional cookies, only cookies necessary to provide you the services will be used. You may change your selection by clicking “Manage Cookies” at the bottom of the page. [Privacy Statement](#) [Third-Party Cookies](#)

Accept

Reject

Manage cookies

Microsoft Ignite

Nov 19–22, 2024

Register now >



Learn

Discover ▾

Product documentation ▾

Development languages ▾

Topics ▾



Sign in

.NET

Languages ▾

Features ▾

Workloads ▾

APIs ▾

Troubleshooting

Resources ▾

Download .NET

Version

.NET 7 ▾

Search

Load

ReflectionOnlyGetAssemblies

SetCachePath

SetData

SetDynamicBase

SetPrincipalPolicy

SetShadowCopyFiles

SetShadowCopyPath

SetThreadPrincipal

ToString

Unload

> Events

> AppDomainSetup

> AppDomainUnloadedException

> ApplicationException

> ApplicationId

> ApplicationIdentity

> ArgIterator

> ArgumentException

> ArgumentNullException

> ArgumentOutOfRangeException

> ArithmeticException

> Array

> ArraySegment<T>.Enumerator

> ArraySegment<T>

> ArrayTypeMismatchException

> AssemblyLoadEventArgs

AssemblyLoadEventHandler

AsyncCallback

> Attribute

AttributeTargets

> AttributeUsageAttribute

... / [API browser](#) / [System](#) / [AppDomain](#) / [Methods](#) /

C# ▾



AppDomain.Load Method

Reference

[Feedback](#)

In this article

[Definition](#)

[Overloads](#)

[Load\(Byte\[\]\)](#)

[Load\(AssemblyName\)](#)

[Show 2 more](#)

Definition

Namespace: [System](#)

Assembly: System.Runtime.dll

Loads an [Assembly](#) into this application domain.

Overloads

[Expand table](#)

Load(Byte[])	Loads the Assembly with a common object file format (COFF) based image containing an emitted Assembly .
Load(AssemblyName)	Loads an Assembly given its AssemblyName .
Load(String)	Loads an Assembly given its display name.
Load(Byte[], Byte[])	Loads the Assembly with a common object file format (COFF) based image containing an emitted Assembly . The raw bytes representing the symbols for the Assembly are also loaded.

Load(Byte[])

Source: [AppDomain.cs](#) [↗](#)

Loads the [Assembly](#) with a common object file format (COFF) based image containing an emitted [Assembly](#).

 Download PDF

C#Copy

```
public System.Reflection.Assembly Load (byte[] rawAssembly);
```

Parameters

rawAssembly [Byte\[\]](#)

An array of type [byte](#) that is a COFF-based image containing an emitted assembly.

Returns

[Assembly](#)

The loaded assembly.

Exceptions

[ArgumentNullException](#)

`rawAssembly` is `null`.

[BadImageFormatException](#)

`rawAssembly` is not a valid assembly for the currently loaded runtime.

[AppDomainUnloadedException](#)

The operation is attempted on an unloaded application domain.

[FileLoadException](#)

An assembly or module was loaded twice with two different evidences.

Examples

The following sample demonstrates the use of loading a raw assembly.

For this code example to run, you must provide the fully qualified assembly name. For information about how to obtain the fully qualified assembly name, see [Assembly Names](#).

C#Copy

```
using System;
using System.IO;
using System.Reflection;
using System.Reflection.Emit;

class LoadRawSnippet {
    public static void Main() {
        AppDomain currentDomain = AppDomain.CurrentDomain;

        InstantiateMyType(currentDomain);    // Failed!

        currentDomain.AssemblyResolve += new ResolveEventHandler(MyResolver);

        InstantiateMyType(currentDomain);    // OK!
    }

    static void InstantiateMyType(AppDomain domain) {
        try {
            // You must supply a valid fully qualified assembly name here.
            domain.CreateInstance("Assembly text name, Version, Culture, PublicKeyToken");
        } catch (Exception e) {
            Console.WriteLine(e.Message);
        }
    }

    // Loads the content of a file to a byte array.
    static byte[] loadFile(string filename) {
        FileStream fs = new FileStream(filename, FileMode.Open);
```

```
byte[] buffer = new byte[(int) fs.Length];
fs.Read(buffer, 0, buffer.Length);
fs.Close();

return buffer;
}

static Assembly MyResolver(object sender, ResolveEventArgs args) {
    AppDomain domain = (AppDomain) sender;

    // Once the files are generated, this call is
    // actually no longer necessary.
    EmitAssembly(domain);

    byte[] rawAssembly = loadFile("temp.dll");
    byte[] rawSymbolStore = loadFile("temp.pdb");
    Assembly assembly = domain.Load(rawAssembly, rawSymbolStore);

    return assembly;
}

// Creates a dynamic assembly with symbol information
// and saves them to temp.dll and temp.pdb
static void EmitAssembly(AppDomain domain) {
    AssemblyName assemblyName = new AssemblyName();
    assemblyName.Name = "MyAssembly";

    AssemblyBuilder assemblyBuilder = domain.DefineDynamicAssembly(assemblyName, AssemblyBuilderAccess.RunAndSave);
    ModuleBuilder moduleBuilder = assemblyBuilder.DefineDynamicModule("MyModule", true);
    TypeBuilder typeBuilder = moduleBuilder.DefineType("MyType", TypeAttributes.Public);

    ConstructorBuilder constructorBuilder = typeBuilder.DefineConstructor(MethodAttributes.Public, CallingConventions.Standard, Type.EmptyTypes);
    ILGenerator ilGenerator = constructorBuilder.GetILGenerator();
    ilGenerator.EmitWriteLine("MyType instantiated!");
    ilGenerator.Emit(OpCodes.Ret);

    typeBuilder.CreateType();

    assemblyBuilder.Save("temp.dll");
}
}
```

Remarks

For information that is common to all overloads of this method, see the [Load\(AssemblyName\)](#) method overload.

Beginning with the .NET Framework 4, the trust level of an assembly that is loaded by using this method is the same as the trust level of the application domain.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Load(AssemblyName)

Source: [AppDomain.cs](#) ↗

Loads an [Assembly](#) given its [AssemblyName](#).

```
public System.Reflection.Assembly Load (System.Reflection.AssemblyName assemblyRef);
```

Parameters

assemblyRef [AssemblyName](#)

An object that describes the assembly to load.

Returns

[Assembly](#)

The loaded assembly.

Exceptions

[ArgumentNullException](#)

`assemblyRef` is `null`.

[FileNotFoundException](#)

`assemblyRef` is not found.

[BadImageFormatException](#)

`assemblyRef` is not a valid assembly for the currently loaded runtime.

[AppDomainUnloadedException](#)

The operation is attempted on an unloaded application domain.

[FileLoadException](#)

An assembly or module was loaded twice with two different evidences.

Remarks

This method should be used only to load an assembly into the current application domain. This method is provided as a convenience for interoperability callers who cannot call the static [Assembly.Load](#) method. To load assemblies into other application domains, use a method such as [CreateInstanceAndUnwrap](#).

If a version of the requested assembly is already loaded, this method returns the loaded assembly, even if a different version is requested.

Supplying a partial assembly name for `assemblyRef` is not recommended. (A partial name omits one or more of culture, version, or public key token. For overloads that take a string instead of an [AssemblyName](#) object, "MyAssembly, Version=1.0.0.0" is an example of a partial name and "MyAssembly, Version=1.0.0.0, Culture=neutral, PublicKeyToken=18ab3442da84b47" is an example of a full name.) Using partial names has a negative effect on performance. In addition, a partial assembly name can load an assembly from the global assembly cache only if there is an exact copy of the assembly in the application base directory ([BaseDirectory](#) or [AppDomainSetup.ApplicationBase](#)).

If the current [AppDomain](#) object represents application domain `A`, and the [Load](#) method is called from application domain `B`, the assembly is loaded into both application domains. For example, the following code loads `MyAssembly` into the new application domain `ChildDomain` and also into the application domain where the code executes:

C#

 Copy

```
AppDomain ad = AppDomain.CreateDomain("ChildDomain");
```

```
ad.Load("MyAssembly");
```

The assembly is loaded into both domains because [Assembly](#) does not derive from [MarshalByRefObject](#), and therefore the return value of the [Load](#) method cannot be marshaled. Instead, the common language runtime tries to load the assembly into the calling application domain. The assemblies that are loaded into the two application domains might be different if the path settings for the two application domains are different.

ⓘ **Note**

If both the [AssemblyName.Name](#) property and the [AssemblyName.CodeBase](#) property are set, the first attempt to load the assembly uses the display name (including version, culture, and so on, as returned by the [Assembly.FullName](#) property). If the file is not found, the [CodeBase](#) property is used to search for the assembly. If the assembly is found using [CodeBase](#), the display name is matched against the assembly. If the match fails, a [FileLoadException](#) is thrown.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Load(String)

Source: [AppDomain.cs](#) ↗

Loads an [Assembly](#) given its display name.

C#

📄 Copy

```
public System.Reflection.Assembly Load (string assemblyString);
```

Parameters

assemblyString [String](#)

The display name of the assembly. See [FullName](#).

Returns

[Assembly](#)

The loaded assembly.

Exceptions

[ArgumentNullException](#)

`assemblyString` is `null`

[FileNotFoundException](#)

`assemblyString` is not found.

BadImageFormatException

`assemblyString` is not a valid assembly for the currently loaded runtime.

AppDomainUnloadedException

The operation is attempted on an unloaded application domain.

FileLoadException

An assembly or module was loaded twice with two different evidences.

Remarks

For information that is common to all overloads of this method, see the [Load\(AssemblyName\)](#) method overload.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Load(Byte[], Byte[])

Source: [AppDomain.cs](#) ↗

Loads the [Assembly](#) with a common object file format (COFF) based image containing an emitted [Assembly](#). The raw bytes representing the symbols for the [Assembly](#) are also loaded.

C# Copy

```
public System.Reflection.Assembly Load (byte[] rawAssembly, byte[]? rawSymbolStore);
```

Parameters

`rawAssembly` [Byte\[\]](#)

An array of type `byte` that is a COFF-based image containing an emitted assembly.

`rawSymbolStore` [Byte\[\]](#)

An array of type `byte` containing the raw bytes representing the symbols for the assembly.

Returns

[Assembly](#)

The loaded assembly.

Exceptions

[ArgumentNullException](#)

`rawAssembly` is `null`.

[BadImageFormatException](#)

`rawAssembly` is not a valid assembly for the currently loaded runtime.

[AppDomainUnloadedException](#)

The operation is attempted on an unloaded application domain.

[FileLoadException](#)

An assembly or module was loaded twice with two different evidences.

Examples

The following sample demonstrates the use of loading a raw assembly.

For this code example to run, you must provide the fully qualified assembly name. For information about how to obtain the fully qualified assembly name, see [Assembly Names](#).

C# Copy

```
using System;
using System.IO;
using System.Reflection;
using System.Reflection.Emit;

class LoadRawSnippet {
    public static void Main() {
        AppDomain currentDomain = AppDomain.CurrentDomain;

        InstantiateMyType(currentDomain);    // Failed!

        currentDomain.AssemblyResolve += new ResolveEventHandler(MyResolver);

        InstantiateMyType(currentDomain);    // OK!
    }

    static void InstantiateMyType(AppDomain domain) {
        try {
            // You must supply a valid fully qualified assembly name here.
            domain.CreateInstance("Assembly text name, Version, Culture, PublicKeyToken");
        } catch (Exception e) {
            Console.WriteLine(e.Message);
        }
    }

    // Loads the content of a file to a byte array.
    static byte[] loadFile(string filename) {
        FileStream fs = new FileStream(filename, FileMode.Open);
        byte[] buffer = new byte[(int) fs.Length];
        fs.Read(buffer, 0, buffer.Length);
        fs.Close();

        return buffer;
    }

    static Assembly MyResolver(object sender, ResolveEventArgs args) {
        AppDomain domain = (AppDomain) sender;

        // Once the files are generated, this call is
        // actually no longer necessary.
        EmitAssembly(domain);

        byte[] rawAssembly = loadFile("temp.dll");
        byte[] rawSymbolStore = loadFile("temp.pdb");
        Assembly assembly = domain.Load(rawAssembly, rawSymbolStore);

        return assembly;
    }

    // Creates a dynamic assembly with symbol information
    // and saves them to temp.dll and temp.pdb
    static void EmitAssembly(AppDomain domain) {
        AssemblyName assemblyName = new AssemblyName();
        assemblyName.Name = "MyAssembly";

        AssemblyBuilder assemblyBuilder = domain.DefineDynamicAssembly(assemblyName, EmitAssemblyOptions.Default);
        AssemblyBuilder assemblyBuilder2 = domain.DefineDynamicAssembly(assemblyName, EmitAssemblyOptions.Default);
    }
}
```

```
ModuleBuilder moduleBuilder = assemblyBuilder.DefineDynamicModule("MyModule", "MyModule.dll");
TypeBuilder typeBuilder = moduleBuilder.DefineType("MyType", TypeAttributes.Public);

ConstructorBuilder constructorBuilder = typeBuilder.DefineConstructor(MethodAttributes.Public,
    Type.EmptyTypes, null);
ILGenerator ilGenerator = constructorBuilder.GetILGenerator();
ilGenerator.EmitWriteLine("MyType instantiated!");
ilGenerator.Emit(OpCodes.Ret);

typeBuilder.CreateType();

assemblyBuilder.Save("temp.dll");
}
```

Remarks

For information that is common to all overloads of this method, see the [Load\(AssemblyName\)](#) method overload.

Beginning with the .NET Framework 4, the trust level of an assembly that is loaded by using this method is the same as the trust level of the application domain.

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	2.0, 2.1

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)