Type.GetTypeFromCLSID Method

Reference

Feedback

In this article

Definition
Overloads
GetTypeFromCLSID(Guid)
GetTypeFromCLSID(Guid, Boolean)
Show 2 more

Definition

Namespace: System

Assembly: System.Runtime.dll

Gets the type associated with the specified class identifier (CLSID).

Overloads

Expand table

GetTypeFromCLSID(Guid)	Gets the type associated with the specified class identifier (CLSID).
GetTypeFromCLSID(Guid, Boolean)	Gets the type associated with the specified class identifier (CLSID), specifying whether to throw an exception if an error occurs while loading the type.
GetTypeFromCLSID(Guid, String)	Gets the type associated with the specified class identifier (CLSID) from the specified server.
GetTypeFromCLSID(Guid, String, Boolean)	Gets the type associated with the specified class identifier (CLSID) from the specified server, specifying whether to throw an exception if an error occurs while loading the type.

GetTypeFromCLSID(Guid)

Source: Type.cs

Gets the type associated with the specified class identifier (CLSID).

C# Copy

```
[System.Runtime.Versioning.SupportedOSPlatform("windows")]
public static Type? GetTypeFromCLSID (Guid clsid);
```

Parameters

clsid Guid

The CLSID of the type to get.

Returns

Type

`System.__ComObject` regardless of whether the CLSID is valid.

Attributes SupportedOSPlatformAttribute

Examples

The following example uses the CLSID of the Microsoft Word Application object to retrieve a COM type that represents the Microsoft Word application. It then instantiates the type by calling the Activator.CreateInstance method, and closes it by calling the Application.Quit method.

C# Copy

```
using System;
using System.Reflection;
using System.Runtime.InteropServices;

public class Example
{
   private const string WORD_CLSID = "{000209FF-0000-0000-C000-000000000046}";

   public static void Main()
   {
      // Start an instance of the Word application.
      var word = Type.GetTypeFromCLSID(Guid.Parse(WORD_CLSID));
      Console.WriteLine("Instantiated Type object from CLSID {0}",
                        WORD_CLSID);
      Object wordObj = Activator.CreateInstance(word);
      Console.WriteLine("Instantiated {0}",
                        wordObj.GetType().FullName);

      // Close Word.
      word.InvokeMember("Quit", BindingFlags.InvokeMethod, null,
                        wordObj, new object[] { 0, 0, false } );
   }
}
// The example displays the following output:
//    Instantiated Type object from CLSID {000209FF-0000-0000-C000-0000000000
//    Instantiated Microsoft.Office.Interop.Word.ApplicationClass
```

Remarks

The GetTypeFromCLSID method supports late-bound access to unmanaged COM objects from .NET Framework apps when you know the COM object's class identifier (CLSID). The class identifier for COM classes is defined in the HKEY_CLASSES_ROOT\CLSID key of the registry. You can retrieve the value of the IsCOMObject property to determine whether the type returned by this method is a COM object.

Tip

You can call the **GetTypeFromProgID** method for late-bound access to COM objects whose programmatic identifier (ProgID) you know.

Instantiating an unmanaged COM object from its CLSID is a two-step process:

1. Get a Type object that represents the `__ComObject` that corresponds to the CLSID by calling the GetTypeFromCLSID method.
2. Call the Activator.CreateInstance(Type) method to instantiate the COM object.

See the example for an illustration.

The GetTypeFromCLSID(Guid) overload ignores any exception that may occur when instantiating a Type object based on the `clsid` argument. Note that no exception is thrown if `clsid` is not found in the registry.

Notes to Callers

This method is intended for use when working with COM objects, not with .NET Framework objects. All managed objects, including those that are visible to COM (that is, their ComVisibleAttribute attribute is `true`) have a GUID that is returned by the GUID property. Although the method returns a Type object that corresponds to the GUID for .NET Framework objects, you can't use that Type object to create a type instance by calling the CreateInstance(Type) method, as the following example shows.

C# Copy

```
using System;
using System.Runtime.InteropServices;

[assembly:ComVisible(true)]

// Define two classes, and assign one an explicit GUID.
[GuidAttribute("d055cba3-1f83-4bd7-ba19-e22b1b8ec3c4")]
public class ExplicitGuid
{ }

public class NoExplicitGuid
{ }

public class Example
{
   public static void Main()
   {
      Type explicitType = typeof(ExplicitGuid);
      Guid explicitGuid = explicitType.GUID;

      // Get type of ExplicitGuid from its GUID.
      Type explicitCOM = Type.GetTypeFromCLSID(explicitGuid);
      Console.WriteLine("Created {0} type from CLSID {1}",
                        explicitCOM.Name, explicitGuid);

      // Compare the two type objects.
      Console.WriteLine("{0} and {1} equal: {2}",
                        explicitType.Name, explicitCOM.Name,
                        explicitType.Equals(explicitCOM));

      // Instantiate an ExplicitGuid object.
      try {
         Object obj = Activator.CreateInstance(explicitCOM);
         Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
      }
      catch (COMException e) {
         Console.WriteLine("COM Exception:\n{0}\n", e.Message);
      }

      Type notExplicit = typeof(NoExplicitGuid);
      Guid notExplicitGuid = notExplicit.GUID;

      // Get type of ExplicitGuid from its GUID.
```

```
      Type notExplicitCOM = Type.GetTypeFromCLSID(notExplicitGuid);
      Console.WriteLine("Created {0} type from CLSID {1}",
                        notExplicitCOM.Name, notExplicitGuid);

      // Compare the two type objects.
      Console.WriteLine("{0} and {1} equal: {2}",
                        notExplicit.Name, notExplicitCOM.Name,
                        notExplicit.Equals(notExplicitCOM));

      // Instantiate an ExplicitGuid object.
      try {
         Object obj = Activator.CreateInstance(notExplicitCOM);
         Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
      }
      catch (COMException e) {
         Console.WriteLine("COM Exception:\n{0}\n", e.Message);
      }
   }
}
// The example displays the following output:
//       Created __ComObject type from CLSID d055cba3-1f83-4bd7-ba19-e22b1b8e
//       ExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {D055CBA3-1F83-4BD7-BA19-E22B1B8EC3C4} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
//
//       Created __ComObject type from CLSID 74f03346-a718-3516-ac78-f351c745
//       NoExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {74F03346-A718-3516-AC78-F351C7459FFB} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
```

Instead, the GetTypeFromCLSID(Guid, String, Boolean) should only be used to retrieve the GUID of an unmanaged COM object, and the resulting Type object that is passed to the CreateInstance(Type) method must represent an unmanaged COM object.

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

GetTypeFromCLSID(Guid, Boolean)

Source: Type.cs

Gets the type associated with the specified class identifier (CLSID), specifying whether to throw an exception if an error occurs while loading the type.

C# Copy

```
[System.Runtime.Versioning.SupportedOSPlatform("windows")]
public static Type? GetTypeFromCLSID (Guid clsid, bool throwOnError);
```

Parameters

clsid Guid

The CLSID of the type to get.

`throwOnError` Boolean

`true` to throw any exception that occurs.

-or-

`false` to ignore any exception that occurs.

Returns

Type

`System.__ComObject` regardless of whether the CLSID is valid.

Attributes SupportedOSPlatformAttribute

Examples

The following example uses the CLSID of the Microsoft Word Application object to retrieve a COM type that represents the Microsoft Word application. It then instantiates the type by calling the Activator.CreateInstance method, and closes it by calling the Application.Quit method. An exception is thrown if an error occurs while loading the type.

C# Copy

```
using System;
using System.Reflection;
using System.Runtime.InteropServices;

public class Example
{
   private const string WORD_CLSID = "{000209FF-0000-0000-C000-000000000046}'

   public static void Main()
   {
      try {
         // Start an instance of the Word application.
         var word = Type.GetTypeFromCLSID(Guid.Parse(WORD_CLSID), true);
         Console.WriteLine("Instantiated Type object from CLSID {0}",
                           WORD_CLSID);
         Object wordObj = Activator.CreateInstance(word);
         Console.WriteLine("Instantiated {0}",
                           wordObj.GetType().FullName, WORD_CLSID);

         // Close Word.
         word.InvokeMember("Quit", BindingFlags.InvokeMethod, null,
                           wordObj, new object[] { 0, 0, false } );
      }
      catch (Exception) {
         Console.WriteLine("Unable to instantiate an object for {0}", WORD_Cl
      }
   }
}
// The example displays the following output:
//    Instantiated Type object from CLSID {000209FF-0000-0000-C000-0000000006
//    Instantiated Microsoft.Office.Interop.Word.ApplicationClass
```

Remarks

The GetTypeFromCLSID method supports late-bound access to unmanaged COM objects from .NET Framework apps when you know the COM object's class identifier (CLSID). The class identifier for COM classes is defined in the HKEY_CLASSES_ROOT\CLSID key of the registry. You can retrieve the value of the IsCOMObject property to determine whether the type returned by this method is a COM object.

Tip

You can call the **GetTypeFromProgID** method for late-bound access to COM objects whose programmatic identifier (ProgID) you know.

Instantiating an unmanaged COM object from its CLSID is a two-step process:

1. Get a Type object that represents the `__ComObject` that corresponds to the CLSID by calling the GetTypeFromCLSID method.
2. Call the Activator.CreateInstance(Type) method to instantiate the COM object.

See the example for an illustration.

Exceptions such as OutOfMemoryException will be thrown when specifying `true` for `throwOnError`, but it will not fail for unregistered CLSIDs.

Notes to Callers

This method is intended for use when working with COM objects, not with .NET Framework objects. All managed objects, including those that are visible to COM (that is, their ComVisibleAttribute attribute is `true`) have a GUID that is returned by the GUID property. Although the method returns a Type object that corresponds to the GUID for .NET Framework objects, you can't use that Type object to create a type instance by calling the CreateInstance(Type) method, as the following example shows.

C# Copy

```
using System;
using System.Runtime.InteropServices;

[assembly:ComVisible(true)]

// Define two classes, and assign one an explicit GUID.
[GuidAttribute("d055cba3-1f83-4bd7-ba19-e22b1b8ec3c4")]
public class ExplicitGuid
{ }

public class NoExplicitGuid
{ }

public class Example
{
   public static void Main()
   {
      Type explicitType = typeof(ExplicitGuid);
      Guid explicitGuid = explicitType.GUID;

      // Get type of ExplicitGuid from its GUID.
      Type explicitCOM = Type.GetTypeFromCLSID(explicitGuid);
      Console.WriteLine("Created {0} type from CLSID {1}",
                        explicitCOM.Name, explicitGuid);

      // Compare the two type objects.
      Console.WriteLine("{0} and {1} equal: {2}",
                        explicitType.Name, explicitCOM.Name,
                        explicitType.Equals(explicitCOM));

      // Instantiate an ExplicitGuid object.
      try {
         Object obj = Activator.CreateInstance(explicitCOM);
         Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
      }
      catch (COMException e) {
         Console.WriteLine("COM Exception:\n{0}\n", e.Message);
      }

      Type notExplicit = typeof(NoExplicitGuid);
      Guid notExplicitGuid = notExplicit.GUID;

      // Get type of ExplicitGuid from its GUID.
      Type notExplicitCOM = Type.GetTypeFromCLSID(notExplicitGuid);
```

```
      Console.WriteLine("Created {0} type from CLSID {1}",
                        notExplicitCOM.Name, notExplicitGuid);

      // Compare the two type objects.
      Console.WriteLine("{0} and {1} equal: {2}",
                        notExplicit.Name, notExplicitCOM.Name,
                        notExplicit.Equals(notExplicitCOM));

      // Instantiate an ExplicitGuid object.
      try {
         Object obj = Activator.CreateInstance(notExplicitCOM);
         Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
      }
      catch (COMException e) {
         Console.WriteLine("COM Exception:\n{0}\n", e.Message);
      }
   }
}
// The example displays the following output:
//       Created __ComObject type from CLSID d055cba3-1f83-4bd7-ba19-e22b1b8e
//       ExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {D055CBA3-1F83-4BD7-BA19-E22B1B8EC3C4} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
//
//       Created __ComObject type from CLSID 74f03346-a718-3516-ac78-f351c745
//       NoExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {74F03346-A718-3516-AC78-F351C7459FFB} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
```

Instead, the GetTypeFromCLSID(Guid, String, Boolean) should only be used to retrieve the GUID of an unmanaged COM object, and the resulting Type object that is passed to the CreateInstance(Type) method must represent an unmanaged COM object.

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

GetTypeFromCLSID(Guid, String)

Source: Type.cs

Gets the type associated with the specified class identifier (CLSID) from the specified server.

C# Copy

```
[System.Runtime.Versioning.SupportedOSPlatform("windows")]
public static Type? GetTypeFromCLSID (Guid clsid, string? server);
```

Parameters

clsid Guid

The CLSID of the type to get.

`server` String

The server from which to load the type. If the server name is `null`, this method automatically reverts to the local machine.

Returns

Type

`System.__ComObject` regardless of whether the CLSID is valid.

Attributes SupportedOSPlatformAttribute

Examples

The following example uses the CLSID of the Microsoft Word Application object to retrieve a COM type that represents the Microsoft Word application from a server named computer17.central.contoso.com. It then instantiates the type by calling the Activator.CreateInstance method, and closes it by calling the Application.Quit method.

C# Copy

```
using System;
using System.Reflection;
using System.Runtime.InteropServices;

public class Example
{
   private const string WORD_CLSID = "{000209FF-0000-0000-C000-000000000046}"

   public static void Main()
   {
      // Start an instance of the Word application.
      var word = Type.GetTypeFromCLSID(Guid.Parse(WORD_CLSID), "computer17.ce
      Console.WriteLine("Instantiated Type object from CLSID {0}",
                        WORD_CLSID);
      try {
         Object wordObj = Activator.CreateInstance(word);
         Console.WriteLine("Instantiated {0}",
                           wordObj.GetType().FullName, WORD_CLSID);

         // Close Word.
         word.InvokeMember("Quit", BindingFlags.InvokeMethod, null,
                           wordObj, new object[] { 0, 0, false } );
      }
      catch (COMException) {
         Console.WriteLine("Unable to instantiate object.");
      }
   }
}
// The example displays the following output:
//    Instantiated Type object from CLSID {000209FF-0000-0000-C000-0000000000
//    Instantiated Microsoft.Office.Interop.Word.ApplicationClass
```

Remarks

The GetTypeFromCLSID method supports late-bound access to unmanaged COM objects from .NET Framework apps when you know the COM object's class identifier (CLSID). The class identifier for COM classes is defined in the HKEY_CLASSES_ROOT\CLSID key of the registry. You can retrieve the value of the IsCOMObject property to determine whether the type returned by this method is a COM object.

Tip

You can call the **GetTypeFromProgID** method for late-bound access to COM objects whose programmatic identifier (ProgID) you know.

Instantiating an unmanaged COM object from its CLSID is a two-step process:

1. Get a Type object that represents the `__ComObject` that corresponds to the CLSID by calling the GetTypeFromCLSID method.
2. Call the Activator.CreateInstance(Type) method to instantiate the COM object.

Notes to Callers

This method is intended for use when working with COM objects, not with .NET Framework objects. All managed objects, including those that are visible to COM (that is, their ComVisibleAttribute attribute is `true`) have a GUID that is returned by the GUID property. Although the method returns a Type object that corresponds to the GUID for .NET Framework objects, you can't use that Type object to create a type instance by calling the CreateInstance(Type) method, as the following example shows.

C# Copy

```
using System;
using System.Runtime.InteropServices;

[assembly:ComVisible(true)]

// Define two classes, and assign one an explicit GUID.
[GuidAttribute("d055cba3-1f83-4bd7-ba19-e22b1b8ec3c4")]
public class ExplicitGuid
{ }

public class NoExplicitGuid
{ }

public class Example
{
   public static void Main()
   {
      Type explicitType = typeof(ExplicitGuid);
      Guid explicitGuid = explicitType.GUID;

      // Get type of ExplicitGuid from its GUID.
      Type explicitCOM = Type.GetTypeFromCLSID(explicitGuid);
      Console.WriteLine("Created {0} type from CLSID {1}",
                        explicitCOM.Name, explicitGuid);

      // Compare the two type objects.
      Console.WriteLine("{0} and {1} equal: {2}",
                        explicitType.Name, explicitCOM.Name,
                        explicitType.Equals(explicitCOM));

      // Instantiate an ExplicitGuid object.
      try {
         Object obj = Activator.CreateInstance(explicitCOM);
         Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
      }
      catch (COMException e) {
         Console.WriteLine("COM Exception:\n{0}\n", e.Message);
      }

      Type notExplicit = typeof(NoExplicitGuid);
      Guid notExplicitGuid = notExplicit.GUID;

      // Get type of ExplicitGuid from its GUID.
      Type notExplicitCOM = Type.GetTypeFromCLSID(notExplicitGuid);
      Console.WriteLine("Created {0} type from CLSID {1}",
                        notExplicitCOM.Name, notExplicitGuid);

      // Compare the two type objects.
      Console.WriteLine("{0} and {1} equal: {2}",
                        notExplicit.Name, notExplicitCOM.Name,
```

```
                        notExplicit.Equals(notExplicitCOM));

      // Instantiate an ExplicitGuid object.
      try {
         Object obj = Activator.CreateInstance(notExplicitCOM);
         Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
      }
      catch (COMException e) {
         Console.WriteLine("COM Exception:\n{0}\n", e.Message);
      }
   }
}
// The example displays the following output:
//       Created __ComObject type from CLSID d055cba3-1f83-4bd7-ba19-e22b1b8e
//       ExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {D055CBA3-1F83-4BD7-BA19-E22B1B8EC3C4} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
//
//       Created __ComObject type from CLSID 74f03346-a718-3516-ac78-f351c745
//       NoExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {74F03346-A718-3516-AC78-F351C7459FFB} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
```

Instead, the GetTypeFromCLSID(Guid, String, Boolean) should only be used to retrieve the GUID of an unmanaged COM object, and the resulting Type object that is passed to the CreateInstance(Type) method must represent an unmanaged COM object.

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

GetTypeFromCLSID(Guid, String, Boolean)

Source: Type.cs

Gets the type associated with the specified class identifier (CLSID) from the specified server, specifying whether to throw an exception if an error occurs while loading the type.

C# Copy

```
[System.Runtime.Versioning.SupportedOSPlatform("windows")]
public static Type? GetTypeFromCLSID (Guid clsid, string? server, bool
throwOnError);
```

Parameters

clsid Guid

The CLSID of the type to get.

server String

The server from which to load the type. If the server name is `null`, this method automatically reverts to the local machine.

`throwOnError` Boolean

`true` to throw any exception that occurs.

-or-

`false` to ignore any exception that occurs.

Returns

Type

`System.__ComObject` regardless of whether the CLSID is valid.

Attributes SupportedOSPlatformAttribute

Examples

The following example uses the CLSID of the Microsoft Word Application object to retrieve a COM type that represents the Microsoft Word application from a server named computer17.central.contoso.com. It then instantiates the type by calling the Activator.CreateInstance method, and closes it by calling the Application.Quit method. An exception is thrown if an error occurs while loading the type.

C# Copy

```
using System;
using System.Reflection;
using System.Runtime.InteropServices;

public class Example
{
   private const string WORD_CLSID = "{000209FF-0000-0000-C000-000000000046}";

   public static void Main()
   {
      try {
         // Start an instance of the Word application.
         var word = Type.GetTypeFromCLSID(Guid.Parse(WORD_CLSID),
                                          "computer17.central.contoso.com",
                                          true);
         Console.WriteLine("Instantiated Type object from CLSID {0}",
                           WORD_CLSID);
         Object wordObj = Activator.CreateInstance(word);
         Console.WriteLine("Instantiated {0}",
                           wordObj.GetType().FullName, WORD_CLSID);

         // Close Word.
         word.InvokeMember("Quit", BindingFlags.InvokeMethod, null,
                           wordObj, new object[] { 0, 0, false } );
      }
      // The method can throw any of a variety of exceptions.
      catch (Exception e) {
         Console.WriteLine("{0}: Unable to instantiate an object for {1}",
                           e.GetType().Name, WORD_CLSID);
      }
   }
}
// The example displays the following output:
//    Instantiated Type object from CLSID {000209FF-0000-0000-C000-0000000000
//    Instantiated Microsoft.Office.Interop.Word.ApplicationClass
```

Remarks

The GetTypeFromCLSID method supports late-bound access to unmanaged COM objects from .NET Framework apps when you know the COM object's class identifier (CLSID). The class identifier for COM classes is defined in the HKEY_CLASSES_ROOT\CLSID key of the

registry. You can retrieve the value of the IsCOMObject property to determine whether the type returned by this method is a COM object.

> **Tip**
>
> You can call the **GetTypeFromProgID** method for late-bound access to COM objects whose programmatic identifier (ProgID) you know.

Instantiating an unmanaged COM object from its CLSID is a two-step process:

1. Get a Type object that represents the `__ComObject` that corresponds to the CLSID by calling the GetTypeFromCLSID method.
2. Call the Activator.CreateInstance(Type) method to instantiate the COM object.

Exceptions such as OutOfMemoryException will be thrown when specifying `true` for `throwOnError`, but it will not fail for unregistered CLSIDs.

Notes to Callers

This method is intended for use when working with COM objects, not with .NET Framework objects. All managed objects, including those that are visible to COM (that is, their ComVisibleAttribute attribute is `true`) have a GUID that is returned by the GUID property. Although the GetTypeFromCLSID(Guid, String, Boolean) method returns a Type object that corresponds to the GUID for a particular managed object, you can't use that Type object to create a type instance by calling the CreateInstance(Type) method, as the following example shows.

C# Copy

```
using System;
using System.Runtime.InteropServices;

[assembly:ComVisible(true)]

// Define two classes, and assign one an explicit GUID.
[GuidAttribute("d055cba3-1f83-4bd7-ba19-e22b1b8ec3c4")]
public class ExplicitGuid
{ }

public class NoExplicitGuid
{ }

public class Example
{
   public static void Main()
   {
      Type explicitType = typeof(ExplicitGuid);
      Guid explicitGuid = explicitType.GUID;

      // Get type of ExplicitGuid from its GUID.
      Type explicitCOM = Type.GetTypeFromCLSID(explicitGuid);
      Console.WriteLine("Created {0} type from CLSID {1}",
                        explicitCOM.Name, explicitGuid);

      // Compare the two type objects.
      Console.WriteLine("{0} and {1} equal: {2}",
                        explicitType.Name, explicitCOM.Name,
                        explicitType.Equals(explicitCOM));

      // Instantiate an ExplicitGuid object.
      try {
         Object obj = Activator.CreateInstance(explicitCOM);
         Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
      }
      catch (COMException e) {
         Console.WriteLine("COM Exception:\n{0}\n", e.Message);
      }
```

```
        Type notExplicit = typeof(NoExplicitGuid);
        Guid notExplicitGuid = notExplicit.GUID;

        // Get type of ExplicitGuid from its GUID.
        Type notExplicitCOM = Type.GetTypeFromCLSID(notExplicitGuid);
        Console.WriteLine("Created {0} type from CLSID {1}",
                          notExplicitCOM.Name, notExplicitGuid);

        // Compare the two type objects.
        Console.WriteLine("{0} and {1} equal: {2}",
                          notExplicit.Name, notExplicitCOM.Name,
                          notExplicit.Equals(notExplicitCOM));

        // Instantiate an ExplicitGuid object.
        try {
           Object obj = Activator.CreateInstance(notExplicitCOM);
           Console.WriteLine("Instantiated a {0} object", obj.GetType().Name);
        }
        catch (COMException e) {
           Console.WriteLine("COM Exception:\n{0}\n", e.Message);
        }
     }
}
// The example displays the following output:
//       Created __ComObject type from CLSID d055cba3-1f83-4bd7-ba19-e22b1b8e
//       ExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {D055CBA3-1F83-4BD7-BA19-E22B1B8EC3C4} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
//
//       Created __ComObject type from CLSID 74f03346-a718-3516-ac78-f351c745
//       NoExplicitGuid and __ComObject equal: False
//       COM Exception:
//       Retrieving the COM class factory for component with CLSID
//       {74F03346-A718-3516-AC78-F351C7459FFB} failed due to the following e
//       80040154 Class not registered
//       (Exception from HRESULT: 0x80040154 (REGDB_E_CLASSNOTREG)).
```

Instead, the GetTypeFromCLSID(Guid, String, Boolean) should only be used to retrieve the GUID of an unmanaged COM object, and the resulting Type object that is passed to the CreateInstance(Type) method must represent an unmanaged COM object.

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