GoDiagram Win Introduction

GoDiagram™ Win for Microsoft® .NET Windows Forms ("Go") is a .NET class library containing a set of Windows Forms controls for easily building interactive diagrams in .NET-based applications.

The User Guide provides details about Go. You will need to read this before you can really make good use of Go.

Installation kits

GoDiagram Win 6.0 is provided only in ZIP format. It includes with DLLs for .NET 3.5, 4.0, 4.5, 4.6, 4.7.1, and 4.8.2 and .NET Core 3.0 as well as compiled examples and the license manager.

For an MSI installation, use GoDiagram 5.2 kits.

Kits serve as both an evaluation kit as well as the full binary product kit both for GoDiagram. When you have purchased and installed a full binary development license, you will be able to compile and distribute applications using that assembly without getting evaluation watermarks.

Before you install Go, you should already have installed the version of the .NET Framework SDK that the kit depends upon.

GoDiagram Win Files

GoDiagram Win consists of seven assemblies:

- **`Northwoods.Go.dll`**, holding the **Northwoods.Go** namespace
- **`Northwoods.Go.Layout.dll`**, holding the **Northwoods.Go.Layout** namespace
- **`Northwoods.Go.Instruments.dll`**, holding the **Northwoods.Go.Instruments** namespace
- **`Northwoods.Go.Xml.dll`**, holding the **Northwoods.Go.Xml** namespace
- **`Northwoods.Go.Svg.dll`**, holding the **Northwoods.Go.Svg** namespace
- **`Northwoods.Go.Draw.dll`**, holding the **Northwoods.Go.Draw** namespace
- **`Northwoods.Go.Pdf.dll`**, holding the **Northwoods.Go.Pdf** namespace

The assemblies are in the **`libx.y`** subdirectories of the Go installation. They only depend on the Microsoft .NET System, Windows Forms, and Drawing assemblies. They do not include any unmanaged code and do not require any particular permission beyond what any Windows Forms application would need.

.NET Core 3.0 assemblies are in the netcoreapp3.0 folder.

Detailed documentation on the types in these libraries is provided in the compiled HTML help file. This file, along with other documentation, is in the **`docs`** subdirectory of the Go installation.

It also places some example code in the **`Samples`** subdirectories. Both sample executable applications and the sample application source code are included. You can rebuild the sample applications by opening the project files in Visual Studio and compile and debug them individually. Or you can open the Samples3.sln or

for VS2008 (or Samples4.sln or SamplesVB4.sln for VS2010 and “45” for VS2012/2013, “46” for VS 2015, and “47” for VS 2017) to see all the samples in one place.

Samples for Visual Basic have been removed from the 6.0 kit. Install an older 5.3 kit for those samples.

Initial Experiences

If you haven’t already run the sample applications, just to get a feel for what GoDiagram can do, please try them. Reading the source code for the applications will really help you understand how easily you can implement different kinds of features. Remember that these are sample applications. Sometimes functionality is implemented just for the sake of demonstration—no real application would want to have that combination of features, or so many different ways to achieve the same kind of functionality.

If you have certain features you know you want to implement, but are not sure how to do so, it might help to read the Frequently Asked Questions (FAQ) document, **`GoDiagramFAQ.htm`**, in the **`docs`** subdirectory. Another source of inspiration can be the GoDiagram forum at http://forum.nwoods.com/.

It might also help to read the entire User Guide, **`GoUserGuide.doc`**, because it discusses much of the programming model embodied in Go. If you don’t have that much time, at least read the *Go Concepts* chapter in the User Guide.

Customizing Visual Studio

If you are using Visual Studio, you’ll want to customize your Toolbox to include the three controls provided by the **`Northwoods.Go.dll`** assembly.

1. Start up Visual Studio
2. View the Toolbox, if it isn’t already visible.
3. Open up the tab that you want to hold the Go controls. You may want to create a new tab, or you may want to use an existing tab of Windows Forms controls.
4. Context click (right-mouse click) in the desired toolbox tab window. Choose the “Add/Remove Items” or “Choose Items…” context menu command. The Toolbox customization dialog will appear.
5. Select the “.NET Framework Components” tab.
6. Scroll down until you find the **GoView**, **GoPalette**, and **GoOverview** controls, in the Go assembly. If you do not see these controls, you may need to click the “Browse…” button to open the assembly in the **`lib`** subdirectory of the Go installation. Make sure all three controls have check marks by them.
7. Click OK for this dialog. The three controls should appear in your toolbox.

You can now drag any of the controls onto your Form that you are designing. The Properties window will let you specify many of the properties and events to customize the appearance and behavior of the selected view.

Creating a Standard MDI Windows Forms Application

It is very easy to use Visual Studio and its Form Designer to create a new single document interface application that uses Go. However, when you want to create a “standard” multiple document interface application, there are a lot of details that you must implement.

We have already done much of the work for you, in the form of the ProtoApp sample application. The ProtoApp sample is a complete, runnable application, but you’ll want to customize it.
You can use this project to get started on your own application.

1. Depending on whether you want to code in VB.NET or C#, copy the appropriate directory, **`Samples\ProtoApp`** or **`SamplesVB\ProtoApp`** to your own directory.
2. Replace "ProtoApp" with your own project name in all of the files, and then edit the code to suit your needs.
3. Search for "TODO" to find some of the places you'll want to customize.

You might consider starting off with some of the other sample applications, if they are closer to what you are looking for. The organization chart, state diagram, and flow chart editing programs are popular starting points. The class hierarchy browser is also representative of part of many Go applications, and is useful for learning Go in its own right.

Permissions

The **`Northwoods.Go.dll`** assembly requires at least the following permission:

UIPermissionWindow.SafeSubWindows

Depending on the functionality you use, it may also need the following permissions:

UIPermissionWindow.AllWindows
UIPermissionClipboard.AllClipboard
PrintingPermissionLevel.AllPrinting
FileIOPermissionAccess.Read, for image files your application uses, if any

The **`Northwoods.Go.dll`** assembly does not on its own need to call unmanaged code, perform serialization (except through the clipboard), manipulate threads, or use reflection. Files are only read when you specify file names for **GoImage**s. No network I/O occurs except when **GoImage** loads an image from the URI given by **GoImage.Name** when **GoImage.NameIsUri** is true.

Special Deployments

Once you have a binary development license for GoDiagram, you will also be able to produce:

- No-touch or one-click deployment applications that run with reduced trust levels
- Web pages with embedded DLLs that use GoDiagram

Contact us for a replacement DLLs that will work in a reduced-trust environment.

But note that features will fail or not be available in reduced-trust environments. For example, the user will not be able to read or write local files in a medium-trust environment, and the user will not be able to drag-and-drop between windows or do in-place text editing in a low-trust environment.

This is a list of known restrictions that have been found to affect using GoDiagram Win, and how the restrictions are handled:

- Trying to set **Control.AllowDrop** to true when constructing a **GoView**, **GoPalette**, or **GoOverview** may signal a **SecurityException.** This is handled silently by those constructors (except for a Trace message) and they set **GoView.AllowDragOut** to false. Users will not be able to do drag-and-drop; dragging between windows is disabled, and dragging within a **GoView** is handled by tracking mouse up/move/down events.
- Calling **Control.DoDragDrop**; **DoDragDrop** should not be called from **GoToolDragging.Start** when **GoView.AllowDragOut** is false, but if it is called anyway because **AllowDragOut** was set to true, the **SecurityException** is handled silently (except there is a Trace message) and the effect is to only do the internal "dragging", as described above.

- **GoView.CopyToClipboard** and **GoView.PasteFromClipboard** might not work. There is no message or exception.
- **GoView.Print** might not work.
- Creating **Control**s for in-place editing may signal a **SecurityException** in **GoText.DoBeginEdit**; this is handled silently (except there is a Trace message) as if cancelling the edit. Users might not be able to do in-place text editing.
- **GoView.DrawXorLine** and **GoView.DrawXorRectangle** may signal an uncaught **SecurityException**. **GoView.DrawXorBox** will handle the **SecurityException** by drawing a gray rectangle instead.
- **Control.Focus** may cause a **SecurityException**; internally any call to **Focus** will handle any SecurityException silently, again with a Trace message.
- There is no definition of **GoView.ProcessCmdKey**, which in the standard DLL will allow characters that would be treated as menu accelerators be passed on to any in-place editor control that has focus.
- There is no definition of **GoView.IsInputKey**, which in the standard DLL will handle the arrow keys to allow scrolling or moving of the selection, depending on the value of **GoView.DisableKeys**.

Of course there are many other restrictions not involving Go, such as the inability to create MDI child windows.

In any case, you are likely to run into **SecurityException**s as you try to run your application in an environment with restricted permissions. You should first test that your application runs correctly when manually copied to a non-development machine, running with full-trust. Then when running your app as "http://localhost/…" or "http://127.0.0.1/…", we suggest you use exception handlers that display the stack, to give you a better idea of what is failing and when. (Patience is also a virtue.)

To repeat: you should do your deployment testing on a machine where GoDiagram Win is *not* installed. That is just wise testing policy.