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.

GoDiagram Win shares much of the design and implementation with GoDiagram Pocket and GoDiagram Web, which you use to build .NET Compact Framework and ASP.NET web applications, respectively. The User Guide provides details about Go, most of which apply to all three products. You will need to read this before you can really make good use of Go.

If you are interested in using GoDiagram Pocket for building PocketPC applications, please read the `GoPocketIntro.doc` file instead.

Installation kits

The installation kits are Windows Installer files, one for .NET version 1.1, the other for .NET version 2.0. They serve as both an evaluation kit as well as the full binary product kit both for GoDiagram Win and for GoDiagram Pocket, including optional libraries. When you have purchased and installed a full binary development license for an assembly, you will be able to compile and distribute applications using that assembly without getting evaluation reminder messages or watermarks.

Both the base Go assembly and the optional GoLayout and GoInstrument assemblies are included in the binary kit. GoDiagram Pocket assemblies are also provided in this kit. GoDiagram Web is available in a different installation kit.

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 six 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

The assemblies are in the **`lib`** subdirectory 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.

Detailed documentation on the types in these libraries is provided in the **`GoWin.chm`** compiled HTML help file. This file, along with other documentation, is in the **`docs`** subdirectory of the Go installation. You may find it instructive to see a listing of the differences between Windows Forms and Web Forms; this list is maintained in **`GoWinWebDiffs.doc`**. A similar, shorter list of differences between GoDiagram Win and GoDiagram Pocket is maintained in **`GoWinPocketDiffs.doc`.**

It also places some example code in the **Samples** and **SamplesVB** subdirectories. Both sample executable applications and the sample application source code are included. You can rebuild the sample applications in two different manners. You can run the **Samples\BuildSamples.bat** command file in a Visual Studio® Command Prompt window, after first editing that command file to make sure its directory paths are appropriate for your installation. This will recompile all of the sample applications and put the new EXEs in the installation root directory. The **SamplesVB\BuildSamplesVB.bat** will recompile the samples coded in VB.NET. Or you can open the project files in Visual Studio and compile and debug them individually.

Initial Experiences

If you haven't already run the sample applications, just to get a feel for what Go 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.chm**, in the **docs** subdirectory. Another source of inspiration can be the GoDiagram forum at http://www.nwoods.com/forum.

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.

Licensing and Unlock Codes

GoDiagram is licensed per developer. There are no additional fees for distributing or running applications that incorporate our GoDiagram products. Any developer who programs using the GoDiagram application programming interface (API) must be licensed.

Each developer machine must have an installed unlock code. A single paid developer license may have multiple unlock codes associated with it, allowing that developer to work on multiple machines (home machines, laptops, etc.). By default, each GoDiagram license has 2 unlock codes associated with it. If required, a reasonable number of additional unlock codes can be requested by sending e-mail to gosales@nwoods.com.

If no unlock code is installed on a machine, GoDiagram will run in "evaluation mode" on that machine. You will see a watermark in each GoDiagram window and you may see message boxes reminding you that you that this is an evaluation version. In all other respects the functionality is identical to a machine with a valid unlock code installed. You may build and test your applications in evaluation mode, but you will not be able distribute applications built in evaluation mode to other machines (see Deployment).

Requesting and Installing Unlock Codes with the License Manager

Unlock codes are managed using the License Manager. To license your development machine, you need to run the License Manager application (on the Start menu under Northwoods Software, GoDiagram) from that machine and follow the instructions for requesting and installing unlock codes.

If you have already purchased the software, simply select your product in the product list. This will automatically select the different assemblies that come with that product.

Click on "Request Unlock Codes" while connected to the web and fill out the requested information, which includes your e-mail address and the order number and buyer e-mail address. Note that your order number will have been sent to the buyer as the e-mail subject line when GoDiagram was purchased. Click on Submit and the unlock codes will be automatically e-mailed to you. If you have lost or forgotten your order number, click on "Get Order Information" in the License Manager , or if your development machine is not connected to the internet, please send e-mail to gosales@nwoods.com.

Finally, enter the unlock codes by pressing "Enter Unlock Codes". You will be prompted to enter the unlock codes for each assembly. Make sure you enter the matching unlock code for each requested assembly.

Each unlock code is only effective for a few days, so you should enter it in the LicenseManager application promptly, but you can always re-request another unlock code if you need to reinstall GoDiagram in the future on the same machine.

Note that when you successfully enter an unlock code into the GoDiagram LicenseManager, a license key is installed in the registry. If you frequently wipe out your disk drive to replace it with a standard disk image, you do not need to re-request unlock codes each time. You simply have to restore the registry key:

```
HKEY_CURRENT_USER\SOFTWARE\Northwoods Software\Go.NET
```

You must have read access to this registry key in order to develop using GoDiagram or to compile and link your license information into your executable (see Deployment).

Moving or Retiring Unlock Codes

If you are no longer using a particular machine for GoDiagram development, the name of the development machine is changing, or you wish to move GoDiagram development from one machine to another, click on "Remove All Licenses" while running the License Manager from your development machine. Doing so will remove the unlock codes from your current machine, causing GoDiagram to run in evaluation mode once again. You will also be presented with a form to report the deactivation of the machine. Enter your order number and buyer e-mail and click on Submit to send this information to Northwoods. This is important, as it will return your unlock code to the available pool for that order number, allowing you to successfully request a new unlock code for another machine in the future.

Note that if you are changing machine names, it is important to "Remove All Licenses" *before* renaming your machine to allow that machine name to be deactivated.

After retiring the unlock code from the previous machine, simply install your GoDiagram kit on the new machine and run the License Manager as before to request and install unlock codes for the new machine.

Deployment

To deploy an application containing GoDiagram, you must compile and link the license(s) into your application. This will allow your application (including GoDiagram DLLs) to be run on machines other than your development machine without the need for your end users to be take any action, or even be aware of GoDiagram licensing.

Your license information is included in a LICX file that is included in your project (typically LICENSES.LICX). If you are using Microsoft Visual Studio, the LICENSES.LICX file is automatically created for you when you drag and drop a licensed component, such as **GoView**, from the Toolbox onto a Form. This file will be visible using the Visual Studio Solution Explorer (assuming the Show-All-Files property is selected on the Visual Studio Project menu). If this file is not present or does not contain Northwoods information, you may copy and paste the information from the LICENSES.LICX file from one of the GoDiagram sample applications (such as LayoutDemo). Because licensed components other than GoDiagram may also make use of the LICENSES.LICX file, be careful not to delete the information for any other component that you are using.

The contents of the LICENSES.LICX file should be similar to the following:

```
Northwoods.Go.GoView, Northwoods.Go, Version=9.8.7.6, Culture=neutral, PublicKeyToken=a4e3b7b70161cfe8
Northwoods.Go.GoPalette, Northwoods.Go, Version=9.8.7.6, Culture=neutral, PublicKeyToken=a4e3b7b70161cfe8
Northwoods.Go.GoOverview, Northwoods.Go, Version=9.8.7.6, Culture=neutral, PublicKeyToken=a4e3b7b70161cfe8
Northwoods.Go.Layout.GoLayoutForceDirected, Northwoods.Go.Layout, Version=9.8.7.6, Culture=neutral, PublicKeyToken=a4e3b7b70161cfe8
Northwoods.Go.Layout.GoLayoutLayeredDigraph, Northwoods.Go.Layout, Version=9.8.7.6, Culture=neutral, PublicKeyToken=a4e3b7b70161cfe8
Northwoods.Go.Layout.GoLayoutTree, Northwoods.Go.Layout, Version=9.8.7.6, Culture=neutral, PublicKeyToken=a4e3b7b70161cfe8
Northwoods.Go.Instruments.GoInstruments, Northwoods.Go.Instruments, Version=9.8.7.6, Culture=neutral, PublicKeyToken=a4e3b7b70161cfe8
```

Your LICENSES.LICX file should contain one entry for each of the kinds of licensed GoDiagram components or controls used in your application.

You will need to substitute the actual four-part version number that you want to use. The four-part version number is formatted as follows:

m.n.b.f

where:

- m is the GoDiagram major version number
- n is the GoDiagram minor version number
- b is the GoDiagram baselevel
- f is a number identifying the .NET Framework that GoDiagram is linked with (0 for .NET 1.0, 1 for .NET 1.1, 2 for .NET 2.0)

To determine the correct four-part version number that you are using in your Visual Studio project, open the Solution Explorer and examine the References section looking for references with the **Northwoods** prefix. Open the properties window for these references and examine the Version property.

Note that the version numbers specified in the LICENSES.LICX file, the References section of the Solutions Explorer, and the actual GoDiagram dlls distributed with your application must all match exactly!

Also note that the name of the EXE is included when the license key is compiled and linked into your application. If you rename the EXE you may get licensing errors.

You may also use the command line based Microsoft License Compiler (**lc.exe**) to compile the license into your application. This technique is useful for creating batch build files and is used in the `BuildSamples.bat` file included in the `Samples` folder of your GoDiagram installation.

Finally, remember to do your deployment testing on a machine where GoDiagram is *not* installed.

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

When you compile a license with the Microsoft license compiler, it also needs **Read** access to the registry key:

`HKEY_LOCAL_MACHINE\SOFTWARE\Northwoods Software\Go.NET`

This key is created and written by the GoDiagram **`LicenseManager.exe`**.

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
- DLLs with embedded licenses

Contact us for additional licensing instructions and 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.