

# GoDiagram™ Win and GoDiagram Pocket version 2.3.1 Release Notes

Copyright © 2002-2005 Northwoods Software Corporation

This kit includes sets of assemblies and samples for building interactive diagram applications using two different versions of Windows Forms: the full .NET Framework running on Windows and for the .NET Compact Framework on the Pocket PC.

GoDiagram™ Win requires the Microsoft® .NET Framework 1.0 or 1.1 SDK. It also works with interactive development environments such as Visual Studio .NET 2002 or 2003 or 2005, or with Borland C#Builder. At the time of this writing GoDiagram Win assemblies appear to work with .NET 2.0 beta releases, although we have only done some casual testing.

GoDiagram Pocket for Microsoft .NET Compact Framework requires the Microsoft .NET Compact Framework 1.1 SDK or Visual Studio .NET 2003.

GoDiagram Web for ASP.NET Web Forms is a separate product in a separate installation kit. GoDiagram Web and its additional assemblies share most of the same functionality as the Windows Forms components but are designed for ASP.NET to run on web servers.

This kit also includes assemblies and samples for several optional libraries that extend GoDiagram:

- GoLayout, for automatic positioning of nodes
- GoInstruments, for displaying instruments such as dials, meters, gauges, and rulers
- GoXml, for customized reading and writing of XML documents
- GoSvg, for generating customized interactive SVG files (but not available for GoDiagram Pocket)

This installation is initially licensed under the terms of the evaluation license agreement, `WinEvalLicense.rtf` in the `docs` subdirectory. Upon purchasing a valid binary development license for a GoDiagram assembly on a particular computer, and after installing the license with the `LicenseManager` application, you may compile a license for that assembly into your application and distribute that assembly. The binary development license agreement is `docs\WinBinaryLicense.rtf`. Each product has its own redistributable assembly:

- GoDiagram Win: `Northwoods.Go.dll`
- GoLayout Win: `Northwoods.Go.Layout.dll`
- GoInstruments Win: `Northwoods.Go.Instruments.dll`
- GoDiagram Pocket: `Northwoods.GoPocket.dll`
- GoLayout Pocket: `Northwoods.GoPocket.Layout.dll`
- GoInstruments Pocket: `Northwoods.GoPocket.Instruments.dll`

The GoXml and GoSvg assemblies are licensed under the terms of their respective GoDiagram assemblies and can be used at no additional cost:

- GoXml Win: `Northwoods.Go.Xml.dll`
- GoSvg Win: `Northwoods.Go.Svg.dll`
- GoXml Pocket: `Northwoods.GoPocket.Xml.dll`

For GoDiagram Win, please read the **GoDiagram Win Introduction** document,

docs\GoWinIntro.doc.

For GoDiagram Pocket, please read the **GoDiagram Pocket Introduction** document,

docs\GoPocketIntro.doc.

Both documents include a section about how to distribute your applications with the assemblies for which you have purchased a license.

You will find many answers to your questions in the **Frequently Asked Questions (FAQ)** list that is in docs\GoDiagramFAQ.chm.

You might also want to search the **GoDiagram Forum** at <http://www.nwoods.com/forum>.

## Installation Directory Contents

- .\
  - Release Notes (this README file)
- .\lib *assemblies compiled for .NET Framework version 1.0*
  - Northwoods.Go.dll, containing the GoDiagram Win controls and classes
  - Northwoods.Go.Layout.dll, the automatic layout library components (GoLayout) for GoDiagram Win
  - Northwoods.Go.Instruments.dll, the instruments library (GoInstruments) for GoDiagram Win
  - Northwoods.Go.Xml.dll, the library for the customizable reading and writing of XML files
  - Northwoods.Go.Svg.dll, the library for the customizable generation of SVG files
  - \*.xml, XML documentation for code editor tooltips in Visual Studio .NET
- .\lib1.1 *assemblies compiled for .NET Framework version 1.1 and for PocketPC*
  - Northwoods.GoPocket.dll, containing the GoDiagram Pocket controls and classes
  - Northwoods.GoPocket.Layout.dll, the automatic layout library components (GoLayout) for GoDiagram Pocket
  - Northwoods.GoPocket.Instruments.dll, the instruments library (GoInstruments) for GoDiagram Pocket
  - Northwoods.GoPocket.Xml.dll, the library for the customizable reading and writing of XML files
  - Northwoods.Go.dll, containing the GoDiagram Win controls and classes
  - Northwoods.Go.Layout.dll, the automatic layout library components (GoLayout) for GoDiagram Win
  - Northwoods.Go.Instruments.dll, the instruments library (GoInstruments) for GoDiagram Win
  - Northwoods.Go.Xml.dll, the library for the customizable reading and writing of XML files
  - Northwoods.Go.Svg.dll, the library for the customizable generation of SVG files
  - \*.xml, XML documentation for code editor tooltips in Visual Studio .NET
- .\docs
  - GoWinIntro.doc, introduction to GoDiagram Win
  - GoPocketIntro.doc, introduction to GoDiagram Pocket
  - GoDiagramFAQ.chm, Frequently Asked Questions list for GoDiagram (Win, Pocket, and Web)

- `GoUserGuide.doc`, User Guide for GoDiagram (Win, Pocket, and Web)
- `GoLayoutUserGuide.doc`, User Guide for GoLayout (Win, Pocket, and Web)
- `GoInstrumentsUserGuide.doc`, User Guide for GoInstruments (Win, Pocket, and Web)
- `GoWin.chm`, API Reference Manual for GoDiagram, GoXml, GoSvg, GoLayout, and GoInstruments (Win and Pocket)
- `GoWinWebDiffs.doc`, a listing of the differences in GoDiagram between Windows Forms and Web Forms
- `GoWinPocketDiffs.doc`, a listing of the differences in GoDiagram between Win and Pocket
- `WinEvalLicense.rtf`, evaluation license agreement file
- `WinBinaryLicense.rtf`, binary development license agreement file

#### `.\bin`

- `LicenseManager.exe`, to install developer's licenses for the GoDiagram products

#### `.\Samples`, sample GoDiagram Win applications written in C#, all built with and depending on .NET Framework version 1.0

- `*.exe`, sample applications using GoDiagram Win
- Sample source code, in subdirectories:
  - **BasicApp**, a simple editor using basic nodes
  - **Classier**, GoDiagram class hierarchy browser and property/method viewer
  - **Demo1**, GoDiagram demo application, built by modifying prototype app
  - **FamilyTree**, displays the relationships amongst some of the English Tudors
  - **FlowCharter**, a simple flow chart editor
  - **IconicApp**, a simple editor using nodes displaying icons
  - **InstrumentDemo**, demonstrates various dials and meters from the `Northwoods.Go.Instruments` library
  - **LayoutDemo**, demonstrates some of the features of the automatic layout library, `Northwoods.Go.Layout`
  - **MinimalApp**, a minimal GoDiagram application
  - **MovableLinkApp**, a simple editor that supports disconnected or partly connected links
  - **ObjectBrowser**, GoDiagram part hierarchy browser
  - **OrgCharter**, a simple organization chart editor
  - **Processor**, an editor for a model of a work-process system
  - **ProtoApp**, prototype application
  - **StateCharter**, a simple state diagram editor
  - **SubGraphApp**, an editor that supports three different kinds of subgraphs
  - **TreeApp**, a two-dimensional tree that supports expand/collapse
  - **WebWalker**, shows references between HTML web pages
- `BuildSamples.bat`, a command file to recompile the sample applications

#### `.\SamplesVB`, the same GoDiagram Win samples translated from C# to VB.NET, all built with and depending on .NET Framework version 1.0

- `BuildSamplesVB.bat`, a command file to recompile the sample applications

#### `.\PocketSamples`, sample GoDiagram Pocket applications written in C#, all built with and depending on .NET Framework version 1.1

- `*.exe`, sample applications using GoDiagram Pocket
- Sample source code, in subdirectories:

- **MinimalApp**, a minimal GoDiagram Pocket application
  - **Shaper**, a simple app that lets you create different shapes (including Bezier strokes and polygons), animates **GoPie** shapes, rotates the whole document including rotating strokes and polygons, and includes a sample meter from GoInstruments Pocket
- . \PocketSamplesVB, just the same MinimalApp GoDiagram Pocket sample translated from C# to VB.NET, all built with and depending on .NET Framework version 1.1

## Summary of Significant New Features since 2.2

The complete list of changes and new features is included later in this document.

- added **GoView.BackgroundLayer** property, behind all document layers
- added **GoView** properties: **TopBar**, **RightBar**, **BottomBar**, **LeftBar**, **TopLeftCorner**, **TopRightCorner**, **BottomRightCorner**, **BottomLeftCorner**
- added **GoView.OnObjectEnterLeave** event (with corresponding methods and delegate), **GoView.DoObjectEnterLeave**, and **GoObject.OnEnterLeave** method; to simplify implementing mouse-over behaviors when entering and leaving document objects
- added **GoView.NewLinkPrototype** property, subsuming the **GoView.NewLinkClass** property
- added **GoView.NewGoLink** and **NewGoLabeledLink** properties as type-specific values for **NewLinkPrototype**, to simplify initializing new links without defining a **GoView.LinkCreated** event handler
- added **GoView.CursorName** property, for setting **GoView.Cursor** using a cursor name rather than a WinForms **Cursor**
- added **PickObjectsInRectangle** method in **GoView**, **GoDocument**, and **GoLayer**, now called by **GoView.SelectInRectangle**
- **GoDocument.StartTransaction**, **FinishTransaction**, **AbortTransaction**, **Undo**, and **Redo** now raise **Changed** events with corresponding hints, to make updating UI easier using event handlers instead of overriding methods, and to support saving changes to a database as each undo or redo occurs (if not using batch-style save and load)
- added **GoObject.GetCursorName** method, to specify a cursor for an object, for WebForms and for SVG, not just for WinForms
- added mapping names to **GoGroup** children: **ChildName**, **FindChild**, **AddChildName**, **RemoveChildName**, String indexer, **ChildNames**; this allows you to refer to particular child objects without having to refer to them by their position in the **IList/GoGroup** and without adding fields in a subclass to refer to them.
- added **GoGroup.PickableBackground** property (actually promoted up from **GoSubGraph**), so that a user's mouse event in the whole rectangular area of a group can cause the group to be picked, even if there are no child objects at that mouse point
- many changes to **GoSubGraph**, mostly involving removing the restriction that the **Bounds** of the subgraph be the same as where the border is drawn
- added support for interactive reanchoring of **GoBalloons**
- added **GoToolCreating** tool, to let users create objects by background-dragging their bounds, given a prototype object
- Generated SVG now supports limited interaction: tooltips, cursors, custom panning/zooming, HREF links, selection by clicking or by rubber-band dragging
- **GoXmlWriter** methods have been generalized to either write XML to a **Stream** or produce an **XmlDocument**
- many other improvements to GoXml

- examples of customizing SVG generation in Demo1
- example reading/writing GraphML in OrgCharter
- added several example node classes displaying much textual information in different arrangements--see the **InfoNode\*** examples in Demo1
- added example **CollapsibleListGroup** class in Demo1, that uses **GoCollapsibleHandle** to show one of two **GoListGroups**
- added example **BarNode** class in Demo1, as a wide bar with a single port (a **BarPort**) with links only coming in from the top and leaving out the bottom
- added example **AdjustableConnectionBoxPort** and **AdjustableConnectionLink** classes in Demo1, to demonstrate how a port can dynamically adjust the link point at a port for a link as the user drags the end point of the link
- added example timeline class using a **GraduatedScaleLinear** in InstrumentDemo

## Upgrading from 2.2 to 2.3

You will need to make sure your projects now refer to the new Northwoods.Go.dll version 2.3 assembly, not an older version.

You can use either the 2.3.1.0 version which is compiled for .NET Framework 1.0, or the 2.3.1.1 version which is compiled for .NET Framework 1.1. **If you want to deploy into a reduced-trust environment, such as part of a no-touch deployment application, or called by a DLL that is hosted by Internet Explorer, contact us for special licensing instructions and DLLs that can run with limited permissions.**

You will need to change all of your licenses.licx files so that references to GoDiagram Win components of:

Version=2.2.2.0 (or the version you have been using until now)  
are replaced with:

Version=2.3.1.0 (when targeting .NET Framework version 1.0)

or

Version=2.3.1.1 (when targeting .NET Framework version 1.1)

## Known Bugs and Misfeatures

- There are a number of Microsoft bugs that you may encounter:
  - License keys are only recognized when they are linked into an executable assembly (.EXE), not a library (.DLL); you need to have your executable's project refer to Northwoods.Go and include the desired component references in the executable project's LICENSES.LICX file (you can leave them in your library that uses GoDiagram, but they are ineffective there)
  - SOAP serialization
  - text clipping
  - when setting properties on **Controls** created by **GoControls**, and the real control needs to be re-created
  - stopping Timers in .NET Framework 1.1; we have rewritten **GoView** as a work-around to avoid this problem
  - rendering within a **PrintPreviewDialog**

## Detailed List of Changes since 2.2.2

### GoView

- added **GoView.BackgroundLayer** property, behind all document layers
- added **GoView.RescaleWithCenter** method, to change the scale while keeping the center of the view at the same document point
  
- added **GoView** properties: **TopBar**, **RightBar**, **BottomBar**, **LeftBar**, **TopLeftCorner**, **TopRightCorner**, **BottomRightCorner**, **BottomLeftCorner**
- **GoView.CornerControl** is now just implemented as **BottomRightCorner**
- **GoView.VerticalScrollBar** is now just implemented as **RightBar** or **LeftBar**, whichever is a **VScrollBar**
- **GoView.HorizontalScrollBar** is now just implemented as **BottomBar** or **TopBar**, whichever is an **HScrollBar**
  
- added **GoView.ObjectEnterLeave** event, along with **GoObjectEnterLeaveEventHandler** and **GoObjectEnterLeaveEventArgs**, to simplify implementing mouse-enter and mouse-leave event handlers for document objects; this event is only raised for mouse-over events when the **GoToolManager** or **GoToolDragging** tools are running. The event has a reference to the document object that the mouse had been over, to help handle cases with overlapping objects, especially child objects of **GoGroups**.
- added **GoView.DoObjectEnterLeave**, to call **GoView.RaiseObjectEnterLeave**, and to call **GoObject.OnEnterLeave** on the document objects that the mouse had been over and that the mouse is now over
  
- added **GoView.NewLinkPrototype** property, subsuming the **GoView.NewLinkClass** property
- added **GoView.NewGoLink** and **NewGoLabeledLink** read-only properties as type-specific values for **NewLinkPrototype**, to simplify initializing new links without defining a **GoView.LinkCreated** event handler
  
- added **GoView.PrintsViewObjects** property, to easily allow **GoView.PrintView** to print view objects (such as selection handles) as well as document objects [not in Pocket]
- changed **GoLayer.AllowPrint** property to default to true for layers that belong to views
  
- added **GoView.CursorName** property, for setting **GoView.Cursor** using a cursor name rather than a WinForms **Cursor**
  
- added **GoView.ContextClickSingleSelection**, for easier access to the view's **GoToolContext.SingleSelection** property
  
- added **GoView.SelectInRectangleStyle** property, which controls behavior of **GoView.SelectInRectangle**
  
- changed **GoView.MatchesNodeLabel** to take **IGoLabeledPart** instead of **IGoLabeledNode**
  
- added **GoView.FindMouseTool(Type, bool)** to support finding a tool that is an instance

of a subclass of a **Type**

- changed **GoView.GetShadowBrush** and **GetShadowPen** to take the shadowed object as a parameter
- added **GoView.ClipboardCopied** event, raised by **GoView.EditCopy** and **EditCut**, to make updating UI easier using event handlers instead of overriding methods
- added **GoView.GetBitmap** method, to match the same method for WebForms
- fixed setting **GoView.Document** so that the view no longer ignores the next mouse down if it didn't have focus
- fixed external drag-and-drop when **GoView.ExternalDragDropsOnEnter** is true, to occur in a single transaction rather than in two, so that undo/redo behaves more naturally

## GoOverview

- fixed bug when setting **GoOverview.Observed** to remove any **Controls** associated with **GoControls**

## GoPalette

- fixed bug in **GoPalette.LayoutItems** causing positioning to expand when **Orientation.Horizontal**

## GoDocument

- added **GoDocument.LastPartID** property, to allow for proper restoration of document state when loading a saved document.  
*You should change your persistence code to save this property and to restore it before adding objects to the document.*
- improved **GoDocument.EnsureUniquePartID** to make sure **GoDocument.LastPartID** is at least as large as the largest value of **IGoIdentifiablePart.PartID** used in the document
- added **IGoLabeledPart**, as supertype of **IGoLabeledNode**, to provide access to **IGoLabeledPart.Text** string only, so that objects can be found without requiring them to have **GoText Labels**; used by **GoDocument.FindNode**, by **GoView.SelectNextNode** and **MatchesNodeLabel**, and by **GoPalette** sorting
- fixed **GoDocument.FindNode** when searching subgraphs
- added **PickObjectsInRectangle** method in **GoView**, **GoDocument**, and **GoLayer**, now called by **GoView.SelectInRectangle**
- fixed **GoDocument.PickObjects** to return an empty collection instead of null when passed **selectableOnly** but **GoDocument.CanSelectObjects()** is false
- added **GoDocument.Initializing** property, set to true during undo or redo calls to **GoDocument.ChangeValue**
- **GoDocument.StartTransaction**, **FinishTransaction**, **AbortTransaction**, **Undo**, and **Redo** now raise **Changed** events with corresponding hints, to make updating UI easier using event handlers instead of overriding methods, and to support saving changes to a

database as each undo or redo occurs (if not using batch-style save and load)

## GoUndoManager

- added **GoUndoManager.CommitCompoundEdit** method, to permit optimizing the edits that are saved, and to support saving changes to a database as each GoDiagram transaction finishes (if not using batch-style save and load)
- changed **GoUndoManager.EndTransaction** by adding "transaction name" parameter, so that both the transaction name and the presentation name can be passed as values to the **FinishedTransaction Changed** event
- changed some **GoUndoManager** properties to no longer be virtual

## GoObject

- **GoObject.Initializing** property now set to true during undo or redo calls to **GoObject.ChangeValue**
- added **GoObject.GetCursorName** method, to specify a cursor for an object, for WebForms and for SVG, not just for WinForms
- added **GoObject.OnEnterLeave** method, called by **GoView.DoObjectEnterLeave** to handling mouse-overs that enter and leave objects
- changed **GoObject.ParentNode** of ports that are immediate child objects of **GoSubGraphs** to return the subgraph rather than themselves
- improved **GoObject** observer mechanism to also call **OnObservedChanged** when an observed object is added or removed from a layer
- added **GoObject.Copy** method, for convenience in copying a single object without calling **GoDocument.AddCopy**
- added **GoCopyDictionary.FinishDelayedCopies** and **CopyComplete** methods, for convenience when trying to copy objects without calling **GoDocument.CopyFromCollection**
- fixed **GoObject.AddSelectionHandles** not to create resize handles when **GoView.CanResizeObjects()** or **CanReshapeObjects()** is false

## GoGroup

- changed **GoGroup.ContainsPoint** and **GetNearestIntersectionPoint** to check child.**Visible** property instead of calling child.**CanView()** predicate, so some operations work properly on non-**Visible** groups
- changed **GoListGroup** to check **Visible** property of child items instead of calling **CanView()**
- added **GoGroup.PickableBackground** property (actually promoted up from **GoSubGraph**), so that a user's mouse event in the whole rectangular area of a group can cause the group to be picked, even if there are no child objects at that mouse point
- added mapping names to **GoGroup** children: **FindName**, **FindChild**, **AddChildName**, **RemoveChildName**, String indexer, **ChildNames**; this allows you to refer to particular

child objects without having to refer to them by their position in the **IList/GoGroup** and without adding fields in a subclass to refer to them.

- fixed **GoGroup.InsertAfter**(null, child) for the next-to-last child in group

## GoSubGraph

The major change is that the **GoSubGraph.Bounds** need not be the same as the background/border area. In overrides of **GoSubGraph.Layout...** methods, replace calls to **ComputeBounds()** with calls to **ComputeBorder()**. Also, if you have defined an override of **ComputeBounds**, you may be able to delete that definition.

- added methods: **ComputeBorder**, **ComputeInsideMargins**
- renamed **GoSubGraph.ComputeBoundsSkip** method to **ComputeInsideMarginsSkip** method, so that overrides are forced to be renamed and remind you to replace calls to **ComputeBounds()** with calls to **ComputeBorder()**
- changed **GoSubGraph.Layout...** methods to ignore whether child object is **Visible**
- added **GoSubGraph.LayoutCollapsedObject** method; functionality was split out from **LayoutLabel** so that it could be called even if no **Label** is present
- changed collapsed subgraph size to consider each child's **SelectionObject** size, not whole **Size**
- changed collapsed subgraph size to ignore invisible child objects if there is a **CollapsedObject**
- fixed **GoSubGraph.SaveChildBounds** to depend on **ComputeReferencePoint**, instead of assuming the top-left corner is the reference point
- improved **GoSubGraph.CollapseChild** to try to center each child's **SelectionObject** instead of whole child
- improved positioning of subgraph after collapse-expand when there are no child nodes
- removed obsolete **GoSubGraph.Margin** property (replaced by **GoSubGraph.TopLeftMargin** and **.BottomRightMargin**)
  
- fixed bug when collapsing/expanding: update **Printable** property of children as well as the **Visible** property
- fixed bug in **GoSubGraph.CopyChildren** that didn't allow **SavedPaths** to hold **GoLabeledLinks**
- fixed undo/redo of subgraphs not maintaining **SavedBounds** and **SavedPaths** information
  
- added **GoSubGraph.PaintsDecoration** predicate to decide whether **Paint** should call **PaintDecoration**
  
- added **IGoCollapsible**, implemented by **GoSubGraph**
- added **GoCollapsibleHandle**, which **GoSubGraphHandle** now inherits from, to make it easy to define groups or nodes that support collapse/expand in programmer defined manners; now used by TreeApp sample application

## GoShapes, GoText, GoImage, GoControl et al.

- added static/shared method **GoShape.DrawPath**
  
- added **GoStrokeArrowheadStyle.Slash** and **.BackSlash** as new **GoStroke** arrowhead

styles

- fixed **GoStroke.GetSegmentNearPoint** not to assume a minimum line width of 1.0f
- added **GoPolygon.GetSegmentNearPoint**, when you want to see if a point is near the edge of a polygon or poly-Bezier shape, and don't want to check for point containment by using **ContainsPoint**
  
- replaced **GoHandle.Cursor** property with **GoHandle.CursorName** property
- removed **GoHandle.GetCursorForHandle** method
- defined **GoHandle.GetCursorName** method instead of overriding **GoObject.OnMouseOver**
  
- fixed sizing text when printing text objects with **GoText.AutoResizes** false
  
- added **GoImage.ThrowsExceptions** property for controlling whether exceptions are ignored, to help debug failures loading and painting images
- added exception handler when painting image, to catch GDI+ errors when drawing image
  
- changed **GoControl** not to call **GoControl.CreateControl** when shown in a **GoOverview** [not in Pocket]
- **GoControl.Paint** in a **GoOverview** now just paints a light gray rectangle [not in Pocket]

## Nodes, Ports, and Links

- changed many properties on nodes to be virtual; a few that should not have been virtual are now not virtual
  
- improved **GoNode.MoveChildren** and **GoNode.RescaleChildren** to first move/rescale any child objects that are links (**IGoLink**), before moving/rescaling other child objects; this handles movement/rescaling where you define your own nodes that contain links, without using **GoSubGraph**
  
- improved **GoPort.GetLinkPointForPoint** to work with **.PortObject** when node contains the **PortObject** but is not yet part of a document
  
- added **GoBoxNode.Port** setter
  
- added **GoLabeledLink.CalculateStroke** method, for convenience; this just calls **GoLabeledLink.RealLink.CalculateStroke()**.
- improved **GoLink.AddOrthoPoints** to avoid thin long loop for some cases where the ports are near to each other
  
- added **GoBalloon.Reanchorable** and **UnanchoredOffset** properties, **ComputeAnchorPoint** and **PickNewAnchor** methods, to support interactive reanchoring of **GoBalloons**
- added overrides of **GoBalloon.AddSelectionHandles** and **DoResize** to support dragging new resize handle with **AnchorHandle ID**
  
- improved **GoGeneralNode.LayoutChildren** not to stretch the **Icon** if the icon's **AutoRescales** property is false

## Tools

- added **IGoActionObject.OnActionCancelled**, now called from **GoToolAction.DoCancelMouse**
- changed **GoToolAction** to call **IGoActionObject.OnAction** even if **PickActionObject** returns null or a different object (but **GoButton** continues original behavior of not invoking **Action** events if the mouse up is not over the button)
- added static/shared **GoTool.DragSize** property, to indicate how far a mouse-down-move-up needs to go to be considered a drag and not a click
- added **GoToolCreating** tool, to let users create objects by background-dragging their bounds, given a prototype object
- moved instance of **GoToolLinkingNew** tool from **GoView.MouseDownTools** list to **MouseMoveTools** list; to allow selection of ports, at the expense of requiring users to drag the mouse in order to start drawing a new link
- changed **GoToolLinking.StartRelink** second argument from **IGoPort** to boolean to indicate direction, to fix relinking a link that is not connected at either end to a port
- improved **GoToolLinking.PickPort** to ignore any port that is part of the link being relinked
- removed obsolete property **GoToolLinking.Orthogonal**
- added **GoToolManager.DoMouseOver** and **GoToolDragging.DoMouseOver**, to call **GoView.DoObjectEnterLeave** when current document object at the mouse point changes
- added **GoToolRubberBanding.Active** property
- improved **GoToolResizing** to call **GoToolResizing.DoResizing** with state of **GoInputState.Start** when tool is **Started**
- added **GoInputEventArgs.InputState** property, potentially usable by the **GoToolResizing** and **GoToolDragging** tools

## GoLayout

- fixed bug that reduced the effectiveness of **GoLayoutLayeredDigraph** reducing crossings of links coming out of a node with multiple ports
- improved routing of links by **GoLayoutLayeredDigraph** when link is **Orthogonal** or when link **Style** is **GoStrokeStyle.Bezier**.

## GoInstruments

We have added GoInstruments on the PocketPC platform  
(`Northwoods.GoPocket.Instruments.dll`).

- changed **Indicator.OnActionAdjusted** to set **Value** while **SkipsUndoManager** is true, instead of within a transaction
- changed **Indicator.OnAction** to set **Value** within a transaction
- added **Indicator.OnActionCancelled** to restore original **Value**

- added virtual method **Indicator.GetValueForPoint**, for easier overriding in subclasses
- changed **IndicatorBar.GetPhasePath** to be public and to work independently of the current indicator **Value**
- improved **IndicatorBar** phase-related methods and properties to avoid null reference exceptions when used before initialized

## GoXml

**GoXmlWriter** methods have been generalized to either write XML to a **Stream** or produce an **XmlDocument**. Don't use the **XmlTextWriter** unless you also modify the **XmlDocument/XmlElement** if **XmlTextWriter** is null. Use the new **WriteAttrVal**, **WriteStartElement**, **WriteEndElement**, **WriteTextBody** methods instead, because these methods work with both **XmlTextWriter** and **XmlDocument**.

- **GoXmlReader** is no longer abstract and now has public constructor
- changed **GoXmlReader.UseDOM** initial value from true to false
- added **GoXmlReader.RootObject** property, to provide an **IList** or **IGoCollection** for **GoXmlReader.Consume** to **Add** objects to
- renamed **GoXmlReader.CurrentNode** property to **ReaderNode**
- renamed **GoXmlReader.ConstructObject** method to **ConsumeObject**
- added **GoXmlReader.InvokeConsumeChild** and **InvokeConsumeObjectFinish** methods
- added **GoXmlReader.ReadTextBody** and **ReadAttrVal** methods--these work with either **XmlTextReader** (stream) or with **XmlDocument** (DOM)
- added **GoXmlReader.ObjectStack** property, to hold the stack of **Objects** that have been **Allocated** during the tree walk
- reimplemented **GoXmlReader.ParentObject** as read-only property that refers to the top of the **ObjectStack**
- added **GoXmlReader.GrandParentObject** property, that refers to the penultimate object on the **ObjectStack**
  
- **GoXmlWriter** is no longer abstract and now has public constructor
- added **GoXmlWriter.Generate()** method overload to produce **XmlDocument** (DOM) instead of writing to a **Stream**
- added **GoXmlWriter.XmlDocument** public read-only property and protected **SetXmlDocument** method
- added **GoXmlWriter.WriterElement** property, the current **XmlElement** being constructed/written
- renamed **GoXmlWriter.GenerateRenderings** method to **GenerateObjects**
- added **GoXmlWriter.NodesGeneratedFirst** property
- renamed **GoXmlWriter.Define** method to **DefineObject**
- renamed **GoXmlWriter.Render** method to **GenerateObject**
- added **GoXmlWriter.InvokeGenerateElementFinish** method
- added **GoXmlWriter** method: **DefineAndGenerateSharedObject**
- added **GoXmlWriter** methods: **WriteStartElement**, **WriteEndElement**, **WriteTextBody**, **WriteAttrVal**, **SetWriterElementAttribute**
- added **GoXmlWriter** methods: **GetNamespaceUri** and **SetNamespaceUri**
  
- added **GoXmlReaderWriterBase** class, inherited by both **GoXmlReader** and **GoXmlWriter**, to share implementation of transformer management code, and to permit

having a single static method that registers all transformers for both reader and writer

- added **IGoXmlTransformer.ConsumeChild**, **ConsumeObjectFinish**, and **GenerateElementFinish** methods
- added **GoXmlTransformer.ConsumeChild**, **ConsumeObjectFinish**, and **GenerateElementFinish** methods
- added **GoXmlTransformer.IdAttributeUsedForSharedObjects** property
- added **GoXmlTransformer.BodyConsumesChildElements** property
- removed all the **GoXmlTransformer.BaseMMM** methods -- in your method overrides you should just call *base.MMM* instead
- renamed **GoXmlTransformer.CurrentNode** property to **ReaderNode**
- added **GoXmlTransformer.WriterElement** property
- added **GoXmlTransformer** convenience methods: **WriteStartElement**, **WriteEndElement**, **WriteTextBody**, **ReadTextBody**
- replaced **GoXmlTransformer.StringRef** method with **WriteAttrRef** method
- replaced **GoXmlTransformer.StringVal** overloaded methods with **WriteAttrVal** overloaded methods
- added **GoXmlTransformer** methods for converting from a string to various types
- added **GoXmlTransformer.IsAttrPresent** method
- changed **GoXmlTransformer.TypeAttr** to call **Type.GetType** with case insensitive search
- changed **GoXmlTransformer.Allocate** to return an instance of the **GoXmlTransformer.TransformerType**, if it is a class

## GoSvg

Generated SVG now supports limited interaction: tooltips, cursors, custom panning/zooming, HREF links, selection by clicking or by rubber-band dragging.

- fixed SVG generation for collapsed **GoSubGraph**
- fixed SVG generation for **GoStrokes** with non-polygon **Style** arrowheads
- added **GoSvgWriter** constructors that produce **XmlDocument** instead of writing to a **Stream**
- removed **GoSvgWriter.FixedSizeDisplay** property; generated SVG assumes this old property is true
- added **GoSvgWriter.Scripting** and **ScriptFile** properties
- added **GoSvgWriter.PanAndZoomControls**, **ToolTips**, and **SupportsSelection** properties
- added **GoSvgWriter.ObjectsLimitedToDocExtent** property
- added **GoSvgWriter.GetSelectionId** and **GetHref** methods
- added **GoSvgWriter.RenderControlsAndForms**, **RenderPanAndZoomControls**, **RenderToolTip**, and **RenderRubberBand** methods
- added **GoSvgWriter.GenerateScript**, **GenerateBackgroundDecoration**, and **GenerateSelectionHandles** methods

GoSvg is still not yet implemented for the PocketPC; if this functionality is required for your application, please contact us.

## **Support**

Northwoods Software provides e-mail support during the 30-day evaluation period and for 30 days after purchase. If you purchase the optional support subscription, you receive e-mail support for a year after purchase plus all new versions that are released during that period.

For general sales and licensing questions, send e-mail to [GoSales@nwoods.com](mailto:GoSales@nwoods.com).