IBM ILOG Diagram for .NET V2.0

Getting Started

June 2009

Table of Contents

Getting Started

For an accelerated introduction to the IBM® ILOG® Diagram for .NET Framework, follow these steps in sequence:

1. Read the *Overview of IBM ILOG Diagram for .NET* topic.
2. Scan the Tutorials, beginning with *Creating an IBM ILOG Diagram for .NET Windows Forms Application and User Symbol*.
3. Identify the samples that are relevant to your development needs by using the Sample Browser (Start Menu>IBM ILOG>Diagram>Sample Browser) which allows you to quickly scan what the product can offer you. The samples are installed in <*install-dir*>\Samples. Use the samples as a foundation for developing your own IBM ILOG Diagram for .NET applications.
4. As you begin to develop your applications, continue to use the IBM ILOG Diagram for .NET documentation as a primary source of information. The documentation is compiled into an easy-to-use Help file that is displayed in the Microsoft® Document Explorer viewer. The viewer includes features such as an index keyword search, a full-text search, that enable you to customize how the information is displayed.

In This Section

Describes the new features and new documentation for IBM ILOG Diagram for .NET 2.0.

Overview of IBM ILOG Diagram for .NET

Introduces the IBM ILOG Diagram for .NET architecture and its components.

Document Conventions

Shows the typographic conventions for programming elements that are referenced in the documentation.

System Requirements

Lists the minimum and recommended system requirements for client and server applications.

License Management

Provides information about license requirements.

Support

Provides information about product support on the IBM ILOG Diagram for .NET SDK.

What’s New in IBM ILOG Diagram for .NET 2.0

The IBM® ILOG® Diagram for .NET version 2.0 extends version 1.6 with new features, improvements to existing features, and enhancements to the documentation. This section provides information about the key additions and modifications.

In This Section

New IBM ILOG Diagram for .NET Support for Silverlight

IBM® ILOG® Diagram for .NET offers the ability to integrate graph displays inside your Silverlight application. The Silverlight support is very similar to the WPF support already available in version 1.6, and brings a dedicated set of Silverlight elements that ease the creation of graph representations. With these classes, you may create your graph directly from a data source, by specifying the nodes and links that compose your graph in XAML (the XML language of Silverlight) or directly by code. Thanks to the power of Silverlight it is possible to create very appealing graph representations through the styling and templating features of Silverlight combined with the power of the Graph Layout library of IBM ILOG Diagram for .NET. For more details on the Silverlight support, see *IBM ILOG Diagram for .NET and Silverlight*.

Namespace and DLL Name Changes for the WPF Controls

The namespace containing the WPF classes of IBM® ILOG® Diagram for .NET has been changed from **ILOG.Diagrammer.Wpf** to **ILOG.Controls.Diagram**. The DLL containing the WPF classes has been renamed from **ILOG.Diagrammer.Wpf.dll** to **ILOG.Controls.Diagram.dll**.

Dependency Change for the Web Controls

To build AJAX enabled applications, IBM® ILOG® Diagram for .NET 2.0 requires by default the .NET Framework 3.5 SDK. In case you want to target a .NET Framework 2.0 Web application and ASP.NET AJAX 1.0, IBM ILOG Diagram for .NET 2.0 provides the **ILOG.Diagrammer.Web.Ajax10.dll** assembly which has a dependency with ASP.NET AJAX 1.0. Note that in this case, the IBM ILOG Diagram for .NET 2.0 Project Templates cannot be used and the Web controls must be added manually to the Visual Studio Toolbox.

Enhanced Features

IBM® ILOG® Diagram for .NET 2.0 includes the following enhanced features:

Improvements on the ASP.NET Controls

The ILOG.Diagrammer.Web.UI.DiagramHttpHandler class supports a image as attachment mode by means of the **attachmentName** request parameter. When this parameter is specified, the handler generates metadata to instruct the browser to download the image as a file attachment. See ILOG.Diagrammer.Web.UI.DiagramHttpHandler.ProcessRequest method.

Improvements on the Design Time API

The ILOG.Diagrammer.Design.GraphicObjectDesigner.SmartTagMenuTitle property allows you to customize the title displayed by the smart tag menu of a selected **GraphicObject** in the Visual Studio designer.

Improvements on the WPF Support

The WPF Diagram control has new properties that make it easier to populate the diagram. The LinksSource property can be used to specify a second data source containing explicit links between nodes (in addition to the implicit links specified by the SuccessorsBinding, PredecessorsBinding or ParentBinding properties). The StartBinding and EndBinding properties are used to specify the start and end nodes of links contained in the **LinksSource**. The **Nodes** and **Links** collections can be used to add nodes and links programmatically to a **Diagram**.

Improvements on the Editing of GraphicObject Text

Changes have been made to ease customisation of text editing. The following members have been added:

Name	**Type**	**Description**
DiagramView.StartEditText	Method	Starts editing the text of the specified graphic object.
DiagramView.StopEditText	Method	Stops editing the graphic object being edited.
DiagramView.EditedObject	Property	Gets the graphic object whose text is being edited in the **DiagramView**
DiagramView.GraphicObjectTextEditing	Event	Occurs when the editing of the text of a graphic object starts.
DiagramView.GraphicObjectTextEdited	Event	Occurs when the editing of the text of a graphic object ends.
GraphicObject.GetEditTextBounds	Method	Gets the bounds that can be used when editing the text of this graphic object.

In the previous release, the methods related to text editing where located in the SelectInteractor class. These methods are now deprecated.

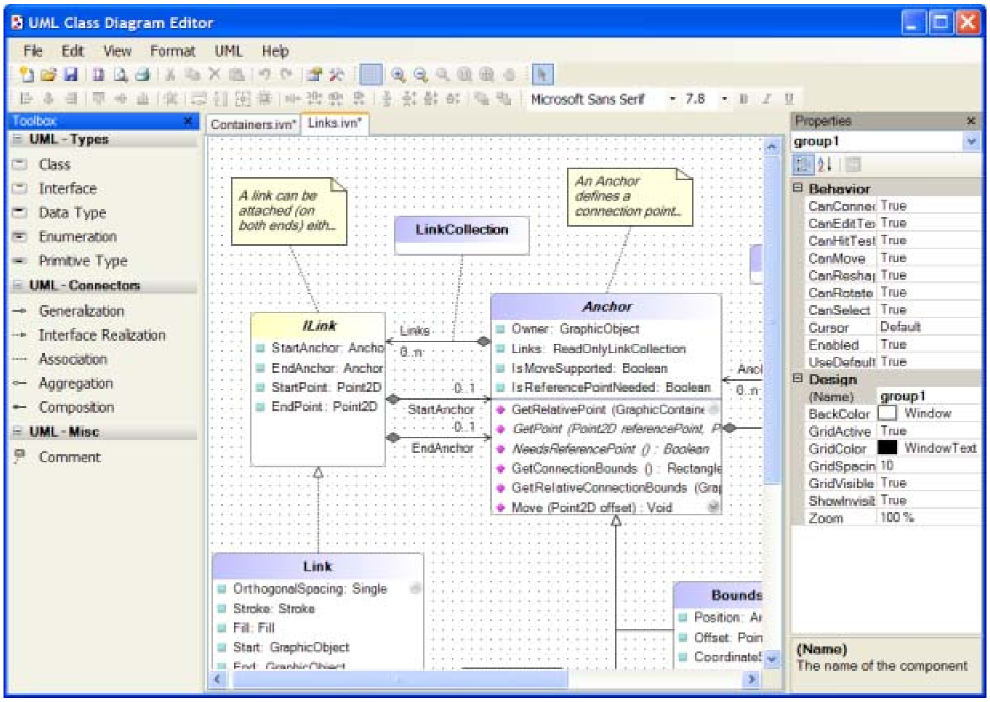
Improvements on the DiagramView API

Name	**Type**	**Description**
DiagramView.CancelInteraction	Method	Cancels any pending intercation.
DiagramView.CreateGhostGraphic	Event	Occurs when the **SelectInteractor** creates a ghost graphic object during a copy or reparent operation.
DiagramView.EnsureVisible(GraphicObject)	Method	Scrolls the view to make the specified graphic object visible.
DiagramView.GetSelectionGraphic	Method	Gets the selection graphic associated with the specified object when it’s selected in the view.
DiagramView.StartAutoScroll	Method	Starts scrolling the view in the specified direction.
GraphicObject.StopAutoScroll	Method	Stops scrolling initiated by calling **StartAutoScroll**.

Overview of IBM ILOG Diagram for .NET

IBM® ILOG® Diagram for .NET is a structured 2D graphics package for creating highly custom, visually rich graphical user interfaces. It complements the components provided by Windows® Forms and Web Forms, allowing .NET GUI programmers to develop far more intuitive displays of information. Examples of such types of displays include schematics, workflow and process flow diagrams, command and control displays, network management displays. IBM ILOG Diagram for .NET is ideal for the rapid development of these custom GUIs.

The following illustration shows a UML editor written with IBM ILOG Diagram for .NET.

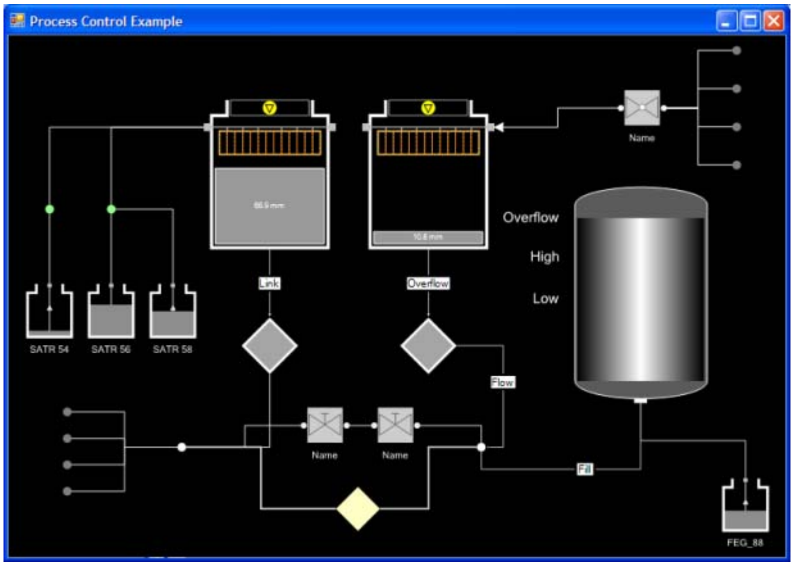


group1

Behavior	
CanConnec	True
CanEditTe	True
CanHitTest	True
CanMove	True
CanReshap	True
CanRotate	True
CanSelect	True
Cursor	Default
Enabled	True
UseDefault	True
Design	
(Name)	group1
BackColor	⬜ Window
GridActive	True
GridColor	⬛ WindowText
GridSpacin	10
GridVisible	True
ShowInvisit	True
Zoom	100 %

(Name)
The name of the component

The following illustration shows a process control application.



The next sections introduce the core types and infrastructure of IBM ILOG Diagram for .NET.

In This Section

IBM ILOG Diagram for .NET Core Types and Infrastructure

Explains the IBM ILOG Diagram for .NET core types.

IBM ILOG Diagram for .NET Diagram Designer

Introduces the IBM ILOG Diagram for .NET Designer.

IBM ILOG Diagram Editor Example

Introduces the IBM ILOG Diagram Editor Example.

Related Sections

Programming with ILOG Diagram for .NET

Describes the essential programming information you need to build applications with IBM ILOG Diagram for .NET.

Tutorials

Provides step-by-step introductions to the fundamental areas of programming for the IBM ILOG Diagram for .NET.

IBM ILOG Diagram for .NET Core Types and Infrastructure

IBM® ILOG® Diagram for .NET is built upon a core set of types and infrastructure that are important to understand. A high percentage of classes in IBM ILOG Diagram for .NET inherit from the following "building block" classes.

Graphic Objects

In IBM ILOG Diagram for .NET a diagram is composed by assembling graphic objects; a graphic object is a subclass of the GraphicObject class. IBM ILOG Diagram for .NET provides many graphic objects that cover a large range of graphical representation. You will find basic shapes such as rectangles, circles, ellipses, polylines, polygons and more complex graphic objects such as panels, controls, gauges and more. For a complete list of predefined graphic objects, see *Using Predefined Graphic Objects*.

Graphic Container

In IBM ILOG Diagram for .NET you assemble graphic objects by adding them to a graphic container. GraphicContainer represents the abstract base class for all graphic containers. Graphic containers are themselves graphic objects, thus you can create a diagram assembled as a complex tree of graphic objects and containers. For details on graphic containers, see *Understanding Graphic Containers*.

Panels

Graphic objects can be assembled in a particular type of container called panels. A panel is a graphic container that lays out the graphic objects it contains automatically. For example, panels allow you to arrange graphic objects horizontally or vertically through the StackPanel or as a grid with the GridPanel. The base class for panels is the Panel class. To learn more about panels see *Panels*.

User Symbols

The UserSymbol class represents the base class for creating user defined graphic objects. A **UserSymbol** gives you the ability to create graphic objects that can be used in multiple places within an application or organization. You can define the graphical representation of the symbol by combining graphic objects or other **UserSymbol** in a subclass of **UserSymbol**. You can also include all the code needed for defining the logic and the user inputs necessary for the symbol. The easiest way to create a new **UserSymbol** is to use the Designer provided with IBM ILOG Diagram for .NET. To learn more about creating user symbols see *Creating Diagrams and User Symbols Using Visual Studio*.

Creating Graph with Nodes and Links

IBM ILOG Diagram for .NET allows you to create representation consisting of shapes connected by lines to other shapes. Such displays often show the objects in the system (the nodes) and their interconnections (the links) as a diagram, also referred to by the mathematical term graph. The primary purpose of a graph is to explain the relationships between objects in a system. Fundamentally, graphs represent an abstraction of some underlying system. They are typically found in utilities, supply chain management, communications networks, business processes, Web services choreographies, production flows, UML diagrams, organizational charts, and many more. In
IBM ILOG Diagram for .NET, any graphic object can be a node of a graph, the connections (or links) between nodes are implemented by the Link class. In conjunction with the **Link** class, the Anchor class allows you to specify how a link connects to a node. See *Introducing Link and Anchor Classes* for details.

Laying Out Graphs Automatically

IBM ILOG Diagram for .NET provides many built-in layout algorithms that allow you to automatically layout graphs made of nodes and links. The GraphLayout class is the base class for lays out. You will find classes to create automatically tree or hierarchical representations. To learn about graph layout algorithms see *Graph Layout Algorithms*.

Animation

IBM ILOG Diagram for .NET provides a built-in animation framework that eases the creation of animations in a diagram. Animations can be used to improve the user experience, by providing transitions between states. For example, when an object needs to disappear, instead of simply hiding it, one can make it fade out by changing its opacity.

It is composed of an animation engine, and a set of predefined animations. The animation engine is represented by the Animator class. It allows you to control and monitor animations at the application level. An Animation is a subclass of the Animation class. To learn how to use animations see *Animating Graphic Objects*.

Displaying a Diagram in Windows Form and Web Form Applications

A diagram created with IBM ILOG Diagram for .NET can be used in a Windows® Forms or Web Form application. The DiagramView class located in ILOG.Diagrammer.Windows.Forms namespace represents the Windows Forms control for displaying a diagram, while the DiagramView and AjaxDiagramView classes located in the ILOG.Diagrammer.Web.UI namespace represent the Web Form controls.

To learn how to display a diagram in a Windows Forms application see *Displaying Diagrams in a Windows Forms Application*.

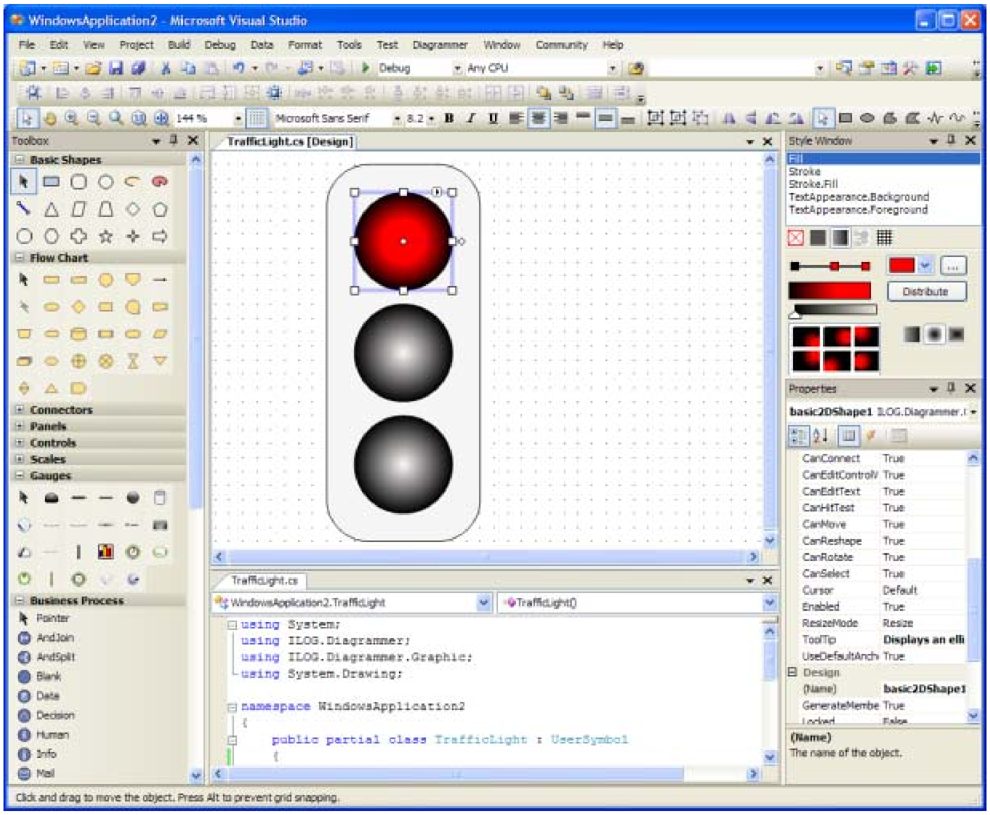
For a Web Forms application see *Displaying Diagrams in an ASP.NET Application.*

IBM ILOG Diagram for .NET Diagram Designer

IBM® ILOG® Diagram for .NET offers developers a powerful foundation for quickly creating advanced graphical user interfaces. In addition to a fully programmable software development kit (SDK), IBM ILOG Diagram for .NET contains a Diagram Designer to automate the production of applications without coding. The Diagram Designer allows you to create complex diagrams and new graphical symbols directly inside Visual Studio.NET.

The typical process for building a diagram with IBM ILOG Diagram for .NET consists first in using the Diagram Designer for creating a library of intelligent graphic symbols that you can reuse through your application or organization. The Diagram Designer is a point-and-click graphical editor that allows user interface developers to quickly create the look and feel of a symbol. Since the Diagram Designer is integrated in Visual Studio.NET you can take advantage of the power of Visual Studio.NET and IBM ILOG Diagram for .NET library to develop and debug complex behaviors for your graphic symbol. Once a library of graphic symbols is created, the symbols can be reused inside the designer to create a diagram or another symbol or can be directly used in a diagram created by code.

The following illustration shows the creation of a traffic light symbol with the Diagram Designer:


WindowsApplication2 - Microsoft Visual Studio
File Edit View Project Build Debug Data Format Tools Test Diagrammer Window Community Help
Debug
Any CPU
144 %
Microsoft Sans Serif
8.2
Toolbox
Basic Shapes
Flow Chart
Connectors
Panels
Controls
Scales
Gauges
Business Process
Pointer
AndJoin
AndSplit
Blank
Data
Decision
Human
Info
Mail
TrafficLight.cs [Design]
TrafficLight.cs
WindowsApplication2.TrafficLight
TrafficLight()
Style Window
Fill
Stroke
Stroke.Fill
TextAppearance.Background
TextAppearance.Foreground
Distribute
Properties
basic2DShape1 ILOG.Diagrammer.
Click and drag to move the object. Press Alt to prevent grid snapping.

```
using System;
using ILOG.Diagrammer;
using ILOG.Diagrammer.Graphic;
using System.Drawing;

namespace WindowsApplication2
{
    public partial class TrafficLight : UserSymbol
    {
```

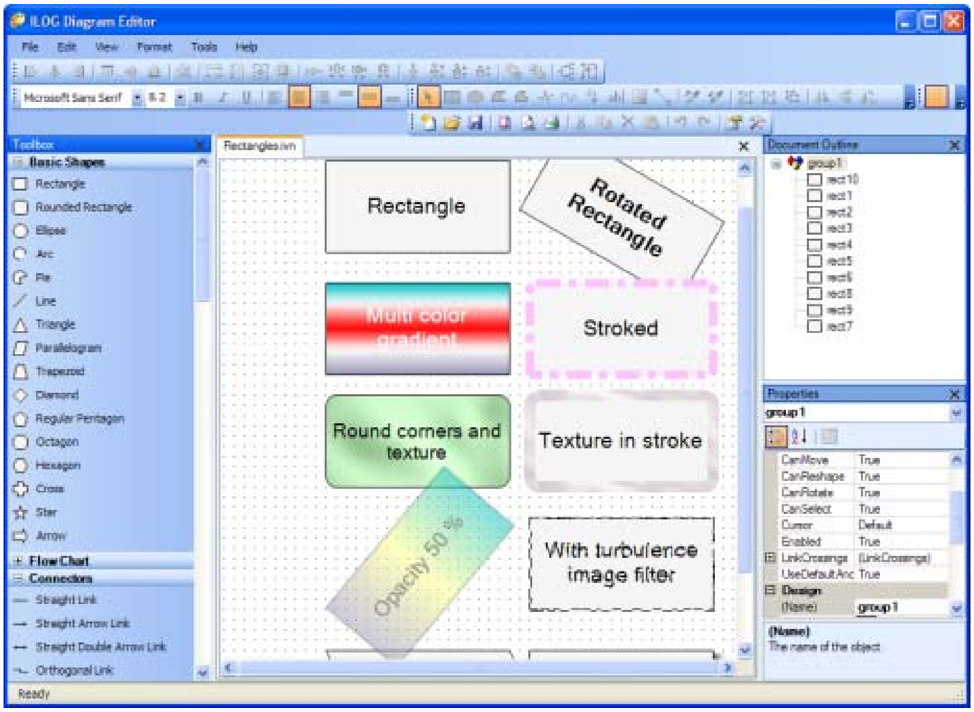
CanConnect	True
CanEditControlV	True
CanEditText	True
CanHitTest	True
CanMove	True
CanReshape	True
CanRotate	True
CanSelect	True
Cursor	Default
Enabled	True
ResizeMode	Resize
ToolTip	**Displays an elli**
UseDefaultAnch	True
Design	
(Name)	**basic2DShape1**
GenerateMembe	True
Locked	False

(Name)
The name of the object.

To learn more on the Diagram Designer see *Building Diagrams and User Symbols Inside Visual Studio*.

IBM ILOG Diagram Editor Example

IBM® ILOG® Diagram for .NET is provided with the IBM ILOG Diagram Editor application which is a powerful foundation that allows developers to quickly create advanced diagram editors. The IBM ILOG Diagram Editor is provided with its source code and contains many common editor features such as editing area, toolbox with symbols, overview, document outline, as shown in the following illustration:



The IBM ILOG Diagram Editor is located under the <install-dir>\Samples\Applications\DiagramEditor directory.

To build a diagram editor with IBM ILOG Diagram for .NET you can use the Diagram Designer (see *IBM ILOG Diagram for .NET Diagram Designer*) to create a library of intelligent graphic symbols that you can reuse in your editor.

Once the library of graphic symbols is created, the symbols can be reused directly inside the IBM ILOG Diagram Editor by importing them in the editor toolbox.

If the editor user interface needs to be modified, you need to create an extension of the IBM ILOG Diagram Editor. Two examples of the IBM ILOG Diagram Editor extension are provided:

- The UML Class Diagram Editor, located under the <install-dir>\Samples\Applications\UMLClassDiagram directory.
- The BPMN Editor, located under the <install-dir>\Samples\Applications\BPMNEditor directory.

Document Conventions

The following table shows the typographic conventions used in the IBM® ILOG® Diagram for .NET documentation.

Convention	**Description**	**Example**
Monospace	Indicates code lines embedded in text and code examples.	`Public Class`
italic	Indicates the placeholders that represent the information supplied by the implementation or the user.	*context* parameter
Bold	Indicates most predefined programming elements, including namespaces, classes, delegates, objects, interfaces, methods, functions, macros, structures, constructors, properties, events, enumerations, fields, operators, statements, directives, data types, keywords, exceptions, non-HTML attributes, and configuration tags, as well as registry keys, subkeys, and values. Also indicates the following HTML elements: attributes, directives, keywords, values, and headers.	**Path** class **Resolve** method

Convention	Description	Example
Capital letters	Indicates the names of keys and key sequences.	ENTER CTRL+R
Plus sign	Indicates a combination of keys. For example, ALT+F1 means to hold down the ALT key while pressing the F1 key.	ALT+F1

System Requirements

To ensure adequate performance, the IBM® ILOG® Diagram for .NET has the following minimum and recommended system requirements.

.NET Framework Requirements

IBM ILOG Diagram for .NET 2.0 Windows Forms and Web Forms components require the .NET Framework, version 2.0.

To build AJAX enabled applications, IBM ILOG Diagram for .NET 2.0 requires by default the .NET Framework 3.5 SDK. In case you want to target a .NET Framework 2.0 Web application and ASP.NET AJAX 1.0, IBM ILOG Diagram for .NET 2.0 provides the **ILOG.Diagrammer.Web.Ajax10.dll** assembly which has a dependency with ASP.NET AJAX 1.0. Note that in this case, the IBM ILOG Diagram for .NET 2.0 Project Templates cannot be used and the Web controls must be added manually to the Visual Studio Toolbox.

To build WPF applications with IBM ILOG Diagram for .NET, you need to have the .NET Framework v3.0 or more installed.

To build Silverlight applications with IBM ILOG Diagram for .NET, you need to have the Silverlight v2.0 installed.

Please refer to the Microsoft® documentation for operating systems and hardware requirements for each Microsoft® .NET framework version.

Visual Studio Requirements

IBM ILOG Diagram for .NET 2.0 integrates into Visual Studio 2008.

Developing IBM ILOG Diagram for .NET applications without having Visual Studio 2008 installed is possible, but you'll not take advantage of the Visual Studio 2008 integration features: documentation, Visual Studio Designer.

To build Silverlight applications inside Visual Studio 2008, you will need to install the Microsoft® Silverlight tools for Visual Studio 2008.

License Management

In order to compile or run applications using the IBM® ILOG® Diagram for .NET Framework, you need a valid license.

You might have received a license file whose content looks like:

```
LICENSE MyCompany
NODE     Diagrammer.NET  1.600 NEVER BSHS80YYVHG c85437cb
RUNTIME  Diagrammer.NET  1.600 NEVER FWEFW5345KK
RTNODE   Diagrammer.NET  1.600 NEVER FWEFW5345KK c85437cb, opt : WebApplicaion
```

Your license must be installed before you can start developing applications using IBM ILOG Diagram for .NET. You can install a license for the product in two different ways:

◆ The license can be specified during the installation process of IBM ILOG Diagram for .NET. Before the installation completes, a dialog asks you to register the license. If you accept, the **IBM ILOG License Manager** tool is launched.

◆ The license can be installed after the installation process by using the **IBM LOG License Manager**.

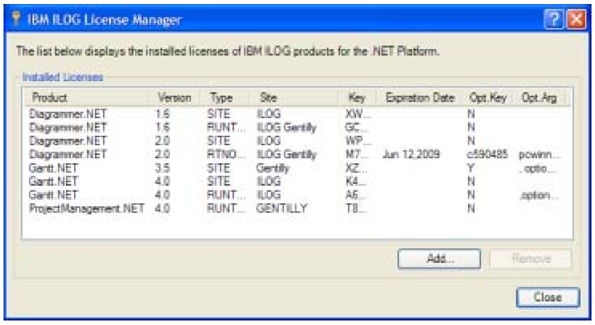
The **IBM ILOG License Manager** tool is available from the Windows® **Start Menu** in **All Programs>IBM ILOG>IBM ILOG Diagram for .NET>License Manager.**

You can use the **IBM ILOG License Manager** tool to:

◆ List installed licenses.

- Add new licenses.
- Remove existing licenses.

The following picture shows the **IBM ILOG License Manager**:



The list below displays the installed licenses of IBM ILOG products for the .NET Platform.

Installed Licenses

Product	Version	Type	Site	Key	Expiration Date	Opt.Key	Opt.Arg
Diagrammer.NET	1.6	SITE	ILOG	XW...		N	
Diagrammer.NET	1.6	RUNT...	ILOG Gentilly	GC...		N	
Diagrammer.NET	2.0	SITE	ILOG	WP...		N	
Diagrammer.NET	2.0	RTNO...	ILOG Gentilly	M7...	Jun 12 2009	c590485	pcwinn...
Gantt.NET	3.5	SITE	Gentilly	XZ...		Y	..optio...
Gantt.NET	4.0	SITE	ILOG	K4...		N	
Gantt.NET	4.0	RUNT...	ILOG	A6...		N	.option...
ProjectManagement.NET	4.0	RUNT...	GENTILLY	T8...		N	

When compiling an application using IBM ILOG Diagram for .NET under Visual Studio 2008, the .NET license compiler will embed your license inside the final application executable file.

Since the license is embedded in the final application, there is no need to install any license on the target machine when deploying a compiled application. For the same reason, you need to recompile your application if your license has changed. For example, if you have started using IBM ILOG Diagram for .NET using an Evaluation license (the second line of your license starting with **EVAL**), the evaluation license is also embedded in the compiled application. Thus, the compiled application will stop working when the evaluation period is over. To move from an evaluation license to a normal license you need to install the new license using the **IBM ILOG License Manager** tool. Then you need to rebuild the application so that the new license will be embedded in the final application. Note that if you are using Visual Studio 2008, you must use the Rebuild command to get the application resources recompiled.

The process of compiling the license inside the final application is automatically done by Visual Studio 2008: when a licensed component is dropped onto a design surface, a license file is automatically created and added to the project. This license file contains the name of the licensed components used by the project. When building the project, Visual Studio 2008 will compile the license file and put the compiled license into the application resources.

This means that:

- If you are not using Visual Studio 2008 to compile your application, you will have to compile a license by yourself using the .NET license compiler (`lc.exe`). Then, you will have to embed the compiled license into the application resources.

- If you are using Visual Studio 2008 but you did not use the toolbox to drag and drop licensed components, you should create a `license.licx` file into your project. This file will be automatically compiled into the resources when building the project. Each line of the file should represent the fully qualified name of a licensed component.

Note: To deploy an IBM ILOG Diagram for .NET Web application, you will need a runtime license with the **WebApplication** option.

Support

IBM Software Support Handbook

This guide contains important information on the procedures and practices followed in the service and support of your IBM products. It does not replace the contractual terms and conditions under which you acquired specific IBM Products or Services. Please review it carefully. You may want to bookmark the site so you can refer back as required to the latest information. The "IBM Software Support Handbook" can be found on the web at

http://www14.software.ibm.com/webapp/set2/sas/f/handbook/home.html

Accessing Software Support

When calling or submitting a problem to IBM Software Support about a particular service request, please have the following information ready:

IBM Customer Number

The machine type/model/serial number (for Subscription and Support calls)

Company name

Contact name

Preferred means of contact (voice or email)

Telephone number where you can be reached if request is voice

Related product and version information

Related operating system and database information

Detailed description of the issue

Severity of the issue in relationship to the impact of it affecting your business needs

Contact by Web

Open service requests is a tool to help clients find the right place to open any problem, hardware or software, in any country where IBM does business. This is the starting place when it is not evident where to go to open a service request.

Service Request (SR) tool offers Passport Advantage clients for distributed platforms online problem management to open, edit and track open and closed PMRs by customer number. Timesaving options: create new PMRs with prefilled demographic fields; describe problems yourself and choose severity; submit PMRs directly to correct support queue; attach troubleshooting files directly to PMR; receive alerts when IBM updates PMR; view reports on open and closed PMRs.

You can find information about assistance for SR at http://www.ibm.com/software/support/help-contactus.html.

System Service Request (SSR) tool is similar to Electronic Service request in providing online problem management capability for clients with support offerings in place on System i, System p, System z, TotalStorage products, Linux, Windows, Dynix/PTX, Retail, OS/2, Isogon, Candle on OS/390 and Consul z/OS legacy products.

IBMLink - SoftwareXcel support contracts offer clients on the System z platform the IBMLink online problem management tool to open problem records and ask usage questions on System z software products. You can open, track, update, and close a defect or problem record; order corrective/preventive/toleration maintenance; search for known problems or technical support information: track applicable problem reports: receive alerts on high impact problems and fixes in error; and view planning information for new releases and preventive maintenance.

Contact by Phone

If you have an active service contract maintenance agreement with IBM , or are covered by Program Services, you may contact customer support teams via telephone. For individual countries, please visit the Technical Support section of the IBM Directory of worldwide contacts via http://www.ibm.com/planetwide/.

Index