IBM ILOG Views V5.3

Product Overview

June 2009

Copyright notice

Trademarks

IBM, the IBM logo, ibm.com, Websphere, ILOG, the ILOG design, and CPLEX are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Notices

For further information see <*installdir*>/license/notices.txt in the installed product.

Table of Contents

About This Manual

This manual introduces the IBM® ILOG® Views Component Suite.

What You Need to Know

This manual assumes that you are familiar with the Microsoft® Windows® or UNIX® environment in which you are going to use IBM ILOG Views, including its particular windowing system. Since IBM ILOG Views is written for C++ developers, the documentation also assumes that you can write C++ code and that you are familiar with your C++ development environment so as to manipulate files and directories, use a text editor, and compile and run C++ programs.

Notation

Typographic Conventions

The following typographic conventions apply throughout this manual:

- Code extracts and file names are written in `courier` typeface.
- Entries to be made by the user are written in *`courier italics`*.

- Some words in *italics*, when seen for the first time, may be found in the glossary at the end of this manual.

Naming Conventions

Throughout this manual, the following naming conventions apply to the API.

- The names of types, classes, functions, and macros defined in the IBM ILOG Views Foundation library begin with `Ilv`.
- The names of classes as well as global functions are written as concatenated words with each initial letter capitalized.

  class `IlvDrawingView`;

- The names of virtual and regular methods begin with a lowercase letter; the names of static methods start with an uppercase letter. For example:

  `virtual IlvClassInfo*` **`getClassInfo`**`() const;`

  `static IlvClassInfo*` **`ClassInfo*`**`() const;`

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. We are interested in continuing to improve your IBM support experience, and encourage you to provide feedback by clicking the Feedback link in the left navigation bar on any page. 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 by telephone. For individual countries, please visit the Technical Support section of the IBM Directory of worldwide contacts.

Welcome to the IBM ILOG Views Component Suite

Welcome to the IBM® ILOG® Views Component Suite. In this product overview you will find:

- *The IBM ILOG Views Component Suite*
  - *Contents*
  - *Base Products*
  - *Optional Products*
- *Why Choose the IBM ILOG Views Component Suite?*
- *Applications that Benefit from the IBM ILOG Views Component Suite*
- *IBM ILOG Views Component Suite Packages*
  - *Foundation*
  - *Studio*
  - *Gadgets*
  - *Application Framework*
  - *Manager*

- *Grapher*
- *Prototypes*
- *Gantt Chart*
- *Charts*
- *Maps*
- *Data Access*
- *Graph Layout*

◆ *Setting Up IBM ILOG Views*, described in the sections:

- *Disk Space and Memory*
- *Distribution Structure*
- *Setup on UNIX*
- *Setup on Windows*
- *Windows Compiling and Linking Options*
- *Library Dependencies*

◆ *Execution Requirements*

◆ *Compilation*

◆ *Library Build Information (Unix platforms only)*

◆ *Licensing*

◆ *Note for Unix Users*

◆ *Note for Windows Server 2003 / XP / Vista Users*

◆ *Note for Microsoft Visual C++ Users*

◆ *Various Versions*

The IBM ILOG Views Component Suite

The IBM® ILOG® Views Component Suite is composed of several products, and each product contains several technical packages. Depending on your application needs, you will select one of these products and possibly options.

In *Base Products* you will find a brief description of each product. Each description indicates the main services, the technical packages that are included, and the possible options.

	Packages												
Base Product Definition	**Foundation**	**Studio**	**Gadgets**	**Application Framework**	**Manager**	**Grapher**	**Prototypes**		**Gantt**	**Charts**	**Maps**	**Data Access**	**Graph Layout**
Views Controls	X	X	X	X									
Views 2D Graphics Standard	X	X			X								
Views 2D Graphics Professional	X	X			X	X	X						
Views	X	X	X	X	X	X	X		X				
Optional Product Definition													
Views Charts										X			
Views Maps											X		
Views Data Access												X	
Views Graph Layout													X

Figure 1 *The IBM ILOG Views Component Suite Products*

Contents

IBM ILOG Views Studio

A ready-to-use version of IBM® ILOG® Views Studio, delivered as a binary file called `ivfstudio` is provided in the `studio/<system>` directory. IBM ILOG Views Studio is a GUI builder that will help you create and generate IBM ILOG Views applications.

Directories

- `data/ilviews`: contains the data files used by the library (inspector panels, message databases, and so on).
- `data/images`: contains image data files.
- `data/icon`: contains some icons.
- `data/DCW`: contains some DCW-generated files.
- `bin`: contains binary files, along with source code. The `README` file in the bin directory explains how to build these binary files. Two useful programs can be built in `<ILVHOME>/bin/<system>`: `ilv2data`, which can build a "resource file" that can be linked with your application in order to make it environment-independent. splitdbm, which converts pre-3.0 message databases into the new format, including the new language definition and encoding.

- `samples`: contains sample files. You can read the `README` file by clicking here. For each sample, you need to go to the platform directory and run the make utility to build it.
- `tools`: contains more specific solutions to common problems.

IBM ILOG Views Data Access: Known Incompatibilities with Version 5.2

Data Access 5.2 is source-compatible with Data Access 5.0 (with the exception of the Open Ingres database). Binary compatibility, however, is not granted. C++ code must be recompiled and programs must be relinked because the header files have been modified (for example, to support new databases).

Note that source compatibility concerns only the documented API (with the exception of APIs of databases no longer supported). Data Access 5.2 supports Oracle 9i, 10g and 11g (but it does not support Oracle 8.x and Open Ingres).

Base Products

The Controls, 2D Graphics—Standard and Professional, and IBM® ILOG® Views form the four **base products**. All of these products incorporate the Foundation classes and the basic IBM ILOG Views Studio, in addition to adding the expertise of the individual product.

- *IBM ILOG Views Controls*
- *IBM ILOG Views 2D Graphics Standard*
- *IBM ILOG Views 2D Graphics Professional*
- *IBM ILOG Views*

IBM ILOG Views Controls

IBM ILOG Views Controls is designed for developing your application GUI with portable controls. These controls respect Microsoft® Windows® systems and Motif® standards, and facilitate your application localization. Development tools are provided to help you draw your panels and associated scripting or C++ code. Wizards and frameworks are also provided to easily implement a Document/View architecture for your application and generate the corresponding user interface.

Packages Included:

- *Foundation*
- *Studio*
- *Gadgets*

- *Application Framework*

Options:

- *IBM ILOG Views Charts*
- *IBM ILOG Views Data Access*

IBM ILOG Views 2D Graphics Standard

The IBM ILOG Views 2D Graphics Standard product is the base product for 2D Graphics.

IBM ILOG Views 2D Graphics Standard enables development of very efficient 2D vector graphics representations with interactive capabilities to zoom, pan, drag and drop, select, and draw your objects. You can display your objects in several layers, each of which can be visible or not. You can zoom and pan very rapidly, and you can have several simultaneous views on your graphical objects, with different zoom levels.

Packages Included:

- *Foundation*
- *Studio*
- *Manager*

Options:

- *IBM ILOG Views Charts*
- *IBM ILOG Views Maps*

IBM ILOG Views 2D Graphics Professional

IBM ILOG Views 2D Graphics Professional adds grapher management (nodes and links), point-and-click business graphic object creation (Prototypes) to IBM ILOG Views 2D Graphics Standard. It stands as a first-class product as soon as your application needs to represent graphs and networks. Using IBM ILOG Views Studio, you can edit your network, and you can create new types of dynamic graphic objects.

Packages Included:

- *Foundation*
- *Studio*
- *Manager*
- *Grapher*
- *Prototypes*

Options:

- *IBM ILOG Views Charts*
- *IBM ILOG Views Maps*
- *IBM ILOG Views Graph Layout*

IBM ILOG Views

IBM ILOG Views includes all packages for base products. It offers the capabilities of IBM ILOG Views Controls and IBM ILOG Views 2D Professional. In addition, it includes the Gantt Chart package.

Packages Included:

- *Foundation*
- *Studio*
- *Gadgets*
- *Application Framework*
- *Manager*
- *Grapher*
- *Prototypes*
- *Gantt Chart*

Options:

- *IBM ILOG Views Charts*
- *IBM ILOG Views Maps*
- *IBM ILOG Views Data Access*
- *IBM ILOG Views Graph Layout*

Optional Products

The following optional products allow you to add specific functionality to your IBM® ILOG® Views product.

Table 1 *IBM ILOG Views Component Suite Optional Products*

Package	Description
IBM ILOG Views Charts	Compatible with all the base products. It is a very powerful product for data visualization, providing polar and Cartesian charts, fixed or scrolling charts (in any direction), selectable orientation, and a high level of customization. It works with any IBM ILOG Views Component Suite product.
IBM ILOG Views Maps	The mapping package for applications requiring cartographic backgrounds. It provides advanced mapping features such as projection systems and load-on-demand. IBM ILOG Views Maps requires the IBM ILOG Views 2D Standard product.
IBM ILOG Views Data Access	Adds connectivity to data—and especially to relational databases. An extension of the IBM ILOG Views Studio editor allows you to graphically build SQL statements and to bind GUI objects to the results. This product requires the IBM ILOG Views Controls product.
IBM ILOG Views Graph Layout	Adds layout capabilities for displaying hierarchical networks of nodes and links. This product requires the IBM ILOG Views 2D Graphics Professional product.

Why Choose the IBM ILOG Views Component Suite?

IBM® ILOG® Views **is a cross platform C++ library** that brings the power of 2D graphics and the convenience of GUI builders into one array of easy-to-learn products. It handles a wide range of graphic tasks, from simple form-based graphical user interfaces to complex real-time vectorial applications managing hundreds of thousands of objects simultaneously. IBM ILOG Views can accomplish all of this because of its advanced architecture, emphasizing separation of behavioral and graphical aspects along with high-level abstractions for easily managing many objects. This architecture also provides for a common set of services across all interfaces, including PostScript dumping, event recording and playback, and object persistence in files.

IBM ILOG Views ***is powerful.*** IBM ILOG Views starts where Microsoft® Windows® systems and the Open Software Foundation's Motif® leave off. You can create a window or a simple widget with Microsoft Windows and Motif, but you cannot create a graph, draw a map, manage a spreadsheet, or animate thousands of objects. With IBM ILOG Views you can display thousands of animated objects at once.

IBM ILOG Views ***is efficient.*** Low memory requirements and optimal data structures allow for extremely fast execution. Quadtrees manage display updates for hundreds or thousands of objects. Double buffering provides flicker-free animation. Optimal memory allocation and minimal objects with shared components mean that even the biggest applications will not swap.

IBM ILOG Views ***is simple.*** High-level graphic objects mean less tedious coding. Do you need a grapher? IBM ILOG Views 2D Graphics Professional has a grapher class. Do you need bar charts, scatter plots, and gauges? IBM ILOG Views has a dedicated product for charts. Spreadsheets? A snap with IBM ILOG Views' matrix class in the IBM ILOG Views Controls Standard product. These and many other high-level classes give you the capability you need to quickly write your interface.

IBM ILOG Views ***is extensible.*** IBM ILOG Views is written in exemplary C++ style. The class hierarchy and internal protocol are well-documented, which means you can write new classes derived from the documented IBM ILOG Views classes. Code reuse is at the center of object-oriented development, and IBM ILOG Views lets you take full advantage of it.

IBM ILOG Views ***is portable.*** IBM ILOG Views is not specific to any one environment or platform. It has been tested and used in many C++ development environments and can be integrated into numerous graphical environments as well. IBM ILOG Views applications work transparently in Windows, Motif, or Xlib. Its use of graphic primitives lets you write code independently of the underlying window system. A single application, for example, can be integrated into both Windows (PC) and Motif (UNIX). You can change the look-and-feel of the display between these two environments by simply calling an IBM ILOG Views function.

IBM ILOG Views ***is ready to use.*** IBM ILOG Views provides easy-to-use GUI editors and scores of predefined objects including gadgets, graphic objects, behaviors, charts, and gauges, as well as spreadsheet functionality and robust graph, drawing, and formula editors. An IBM ILOG Views package includes a complete class hierarchy of gadgets: interactive objects targeted for specific uses with flexible graphic representations and predefined behavior. Gadgets include scroll bars, menu bars, buttons, and so on. IBM ILOG Views lets you mix basic drawing primitives and interactive gadgets together to produce intuitive, decorated interfaces.

IBM ILOG Views ***is modular.*** IBM ILOG Views is available as several products, each of which is specialized in a particular GUI area: GUI controls, advanced 2D graphics, graph layout, maps, charts, and so on. Each product is provided with its libraries, Studio extensions, samples, and comprehensive documentation. You can assemble just the

IBM ILOG Views capabilities you need, while allowing for future growth with the option of integrating additional IBM ILOG Views products as your application needs warrant.

Applications that Benefit from the IBM ILOG Views Component Suite

All applications that need a graphical user interface (GUI) can benefit from IBM® ILOG® Views' timesaving GUI-building features, including automatic generation of C++ code. IBM ILOG Views is a tool you can use to build any graphical user interface, either a standard GUI or one with advanced two-dimensional graphics. Moreover, IBM ILOG Views code is portable: the interface you create on one platform can be ported to another platform with a simple compilation.

Just some examples of the many IBM ILOG Views uses are:

- *Cartographic Applications*
- *Specialized Editors*
- *Control and Process Applications*
- *Network Monitoring Applications*
- *Financial Applications*
- *Transportation Applications*

Cartographic Applications

IBM® ILOG® Views provides the power necessary in this domain because of its multiple view and layer capabilities, its ability to handle a large amount of data, and the specific mapping features of IBM ILOG Views Maps (projections, load-on-demand, and so on.).

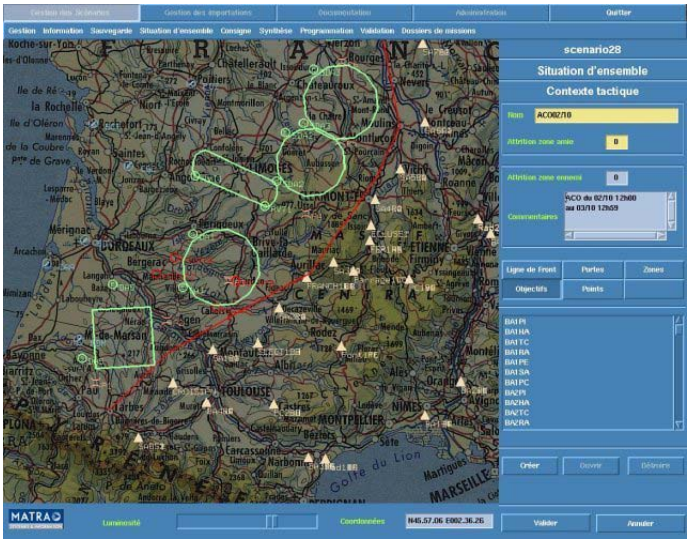


Figure 2 *A Cartographic Application Using IBM ILOG Views Controls, IBM ILOG Views 2D Graphics Professional, and IBM ILOG Views Maps*

Specialized Editors

IBM® ILOG® Views 2D Graphics Standard provides predefined behaviors or “interactors” which, combined with the extensibility of the library, provide for a vast amount of operations easily integrated into an editing application.

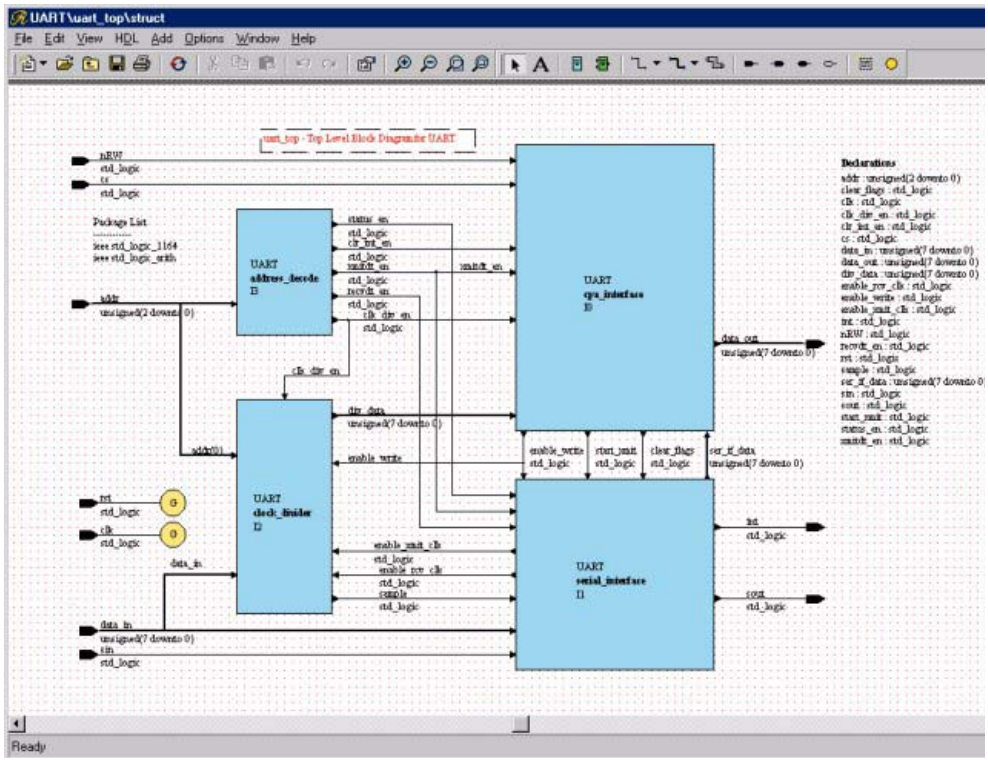


Figure 3 *Diagram Editor*

For editors that need to be portable across platforms, IBM ILOG Views Controls Standard and IBM ILOG Views 2D Graphics Professional combine portable controls, application framework, and editing functionality.

Control and Process Applications

Applications such as system surveillance and electrical network monitoring can benefit from IBM® ILOG® Views linked to real-time application data to animate objects on the screen. IBM ILOG Views 2D Graphics Professional provides Prototypes, business graphic objects that can be designed and animated using a point-and-click editor.

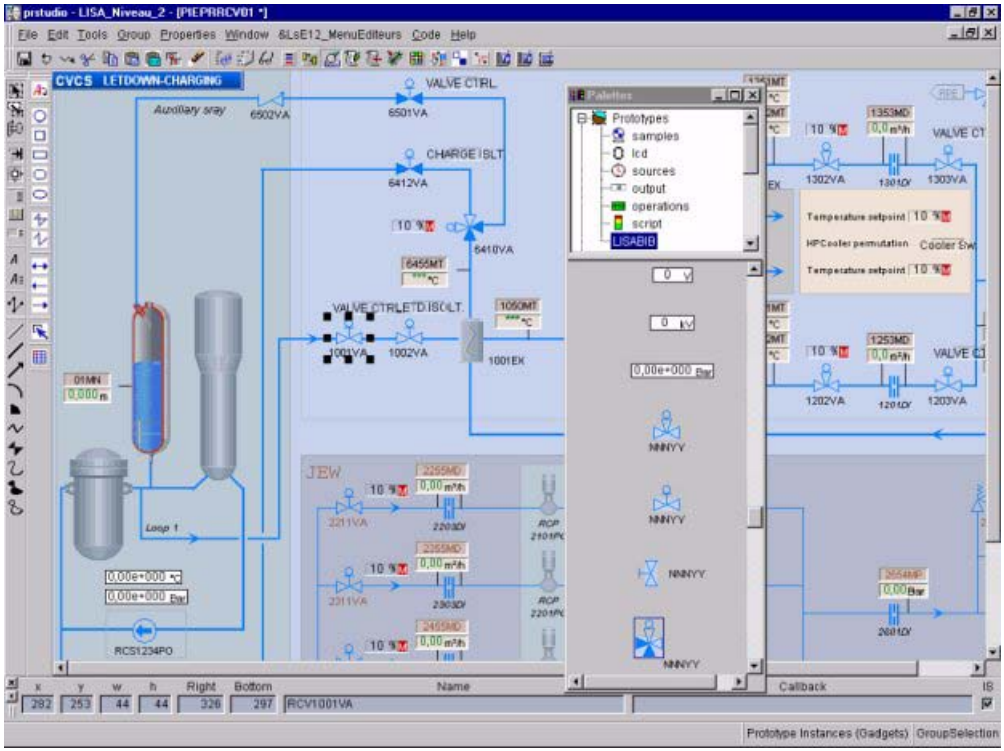


Figure 4 *Graphical Modeling and Simulation System*

Network Monitoring Applications

IBM® ILOG® Views 2D Graphics Professional includes the Grapher, a powerful way to create nodes and links, with each element defined as a separate object that may be of any type: text, a geometric shape, button, complex chart, or one of many other possibilities. The IBM ILOG Views Graph Layout package can perform automatic arrangements of nodes and links, using several predefined layout algorithms.

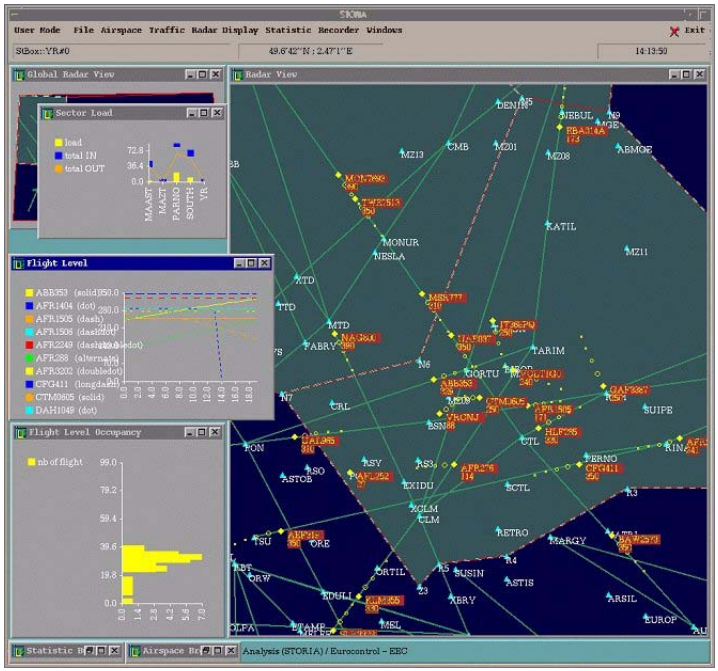


Figure 5 *Air Traffic Management*

IBM ILOG Views Charts provides powerful data visualization features, including all of the most widely used chart types (lines, bars, pies), with polar or Cartesian projections, several scrolling modes, selectable orientation, and many other features.

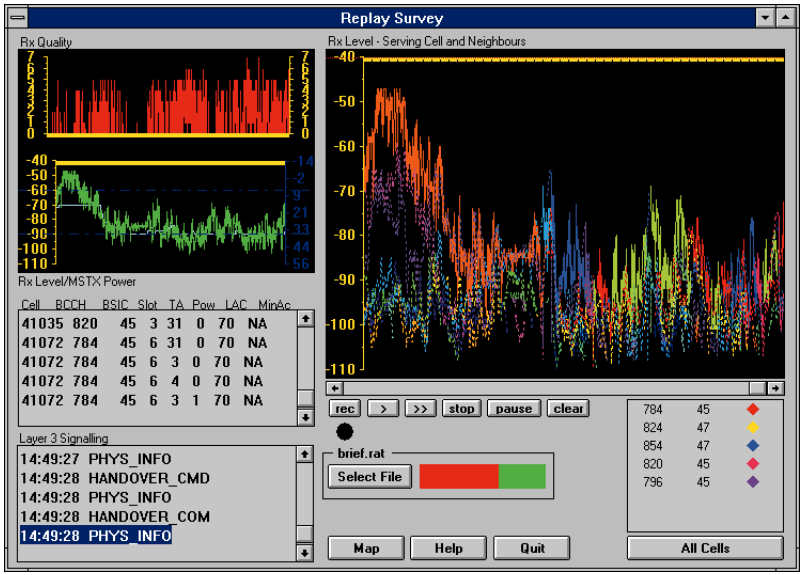


Figure 6 *GSM Network Performance Monitoring*

Financial Applications

IBM® ILOG® Views Controls combined with IBM ILOG Views Charts provides all the necessary tools for financial applications: powerful charts, predefined gauges, and spreadsheets that can display huge amounts of data in real time.

Figure 7 *Trading Application*

Transportation Applications

IBM® ILOG® Views 2D Graphics Standard provides the manager, which can handle the fast updating of graphics in multiple views. This is particularly adapted to air traffic control where the rapid updating of plane position on different maps is essential, and it can be generalized for all applications where moving vehicles must be supervised.

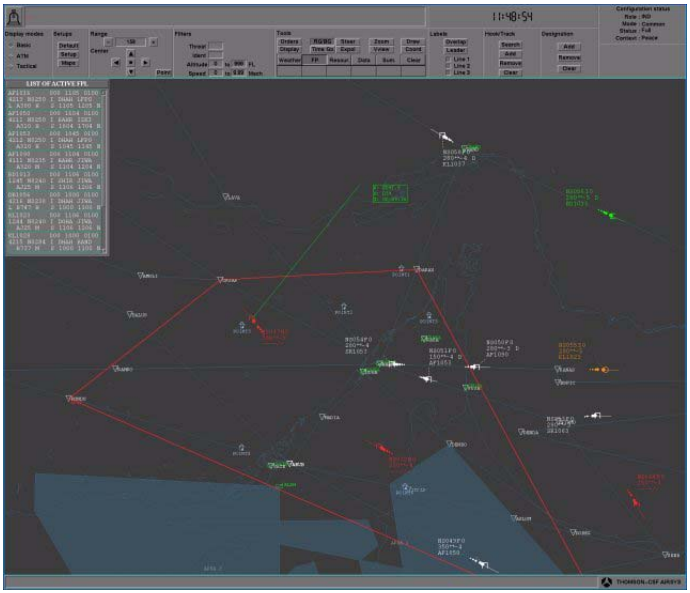


Figure 8 *Air Traffic Control*

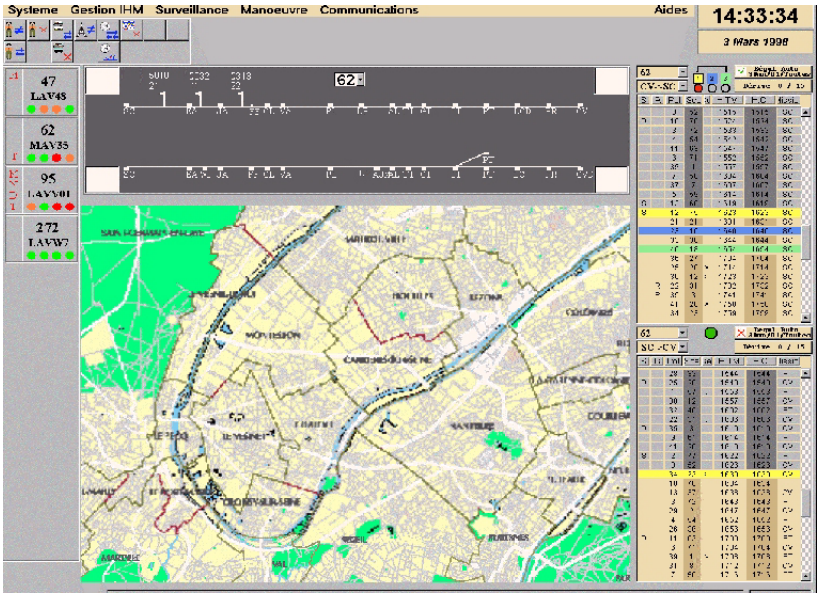


Figure 9 *Fleet Tracking*

IBM ILOG Views Component Suite Packages

For information on a particular IBM® ILOG® Views technical package see:

- *IBM ILOG Views Foundation*
- *IBM ILOG Views Studio*
- *IBM ILOG Views Gadgets*
- *IBM ILOG Views Application Framework*
- *IBM ILOG Views Manager*
- *IBM ILOG Views Grapher*
- *IBM ILOG Views Prototypes*
- *IBM ILOG Views Gantt*
- *IBM ILOG Views Charts*
- *IBM ILOG Views Maps*
- *IBM ILOG Views Data Access*
- *IBM ILOG Views Graph Layout*

IBM ILOG Views Foundation

IBM® ILOG® Views Foundation is a technical package that contains all the base classes for implementing graphic applications. These include display, basic graphic objects, resource management, containers, and basic interactors. It is the base package on top of which any other IBM ILOG Views packages are built.

Features

The features of IBM ILOG Views Foundation are described in:

- *Structured 2D Graphics*
- *Graphics Primitives*
- *Other Services*

Structured 2D Graphics

These features provide basic graphic objects.

- **Predefined graphic objects**: The graphic engine builds on IBM ILOG Views primitives to define *graphic objects*. Rectangles, labels, polygons, splines—and many more—represent the basic graphic objects that you can extend.

IBM ILOG Views makes a clear separation between graphic objects and *behaviors*, thus allowing you to apply a particular behavior to an object.

- **Predefined behavior objects**: In IBM ILOG Views, a predefined behavior is called an *interactor*. An interactor can be applied to any graphic object to give it a particular behavior, thus defining its functionality.
- **Containers**: IBM ILOG Views maintains objects in one of two basic types of storage data structures: containers and managers. A view is associated with a set of graphic objects stored in a container or manager.

  A *container* stores a certain number of graphic objects, and it is associated with only one view, which displays the objects stored in the container. Each object can be associated with a specific behavior, and accelerators—which are keyboard events that immediately call a predefined function—can be attached to the container itself.

  Besides the containers, IBM ILOG Views provides a more efficient data structure called a *manager*. Managers are in the *IBM ILOG Views Manager* package (included in IBM ILOG Views 2D Graphics Standard). Managers provide layers, multiple-view, fast redraw, persistence, and editing functionality. Table 2, explains the difference.

Graphics Primitives

These basic Foundation features allow you to easily perform sophisticated drawing and display operations.

- **Windows and views**: In IBM ILOG Views, a *view* is an object to which basic services can be added, and which is associated with the *window* of the underlying display system, such as X Window™ in Unix. All drawing takes place in the view, which displays an image of the objects or a subset of them. This image can be geometrically transformed by moving, zooming, or rotating without affecting the objects themselves.
- **Drawing primitives**: Using a two-dimensional vector graphic engine, IBM ILOG Views provides drawing ports (memory, screen, and dump file) as well as a large set of *drawing primitives* to create basic *geometric forms*. You can draw basic geometric shapes such as arcs, curves, rectangles, labels, and so on. You can draw on the screen, in memory, or generate dump files such as PostScript. You can create black-and-white and color images.
- **Resources**: IBM ILOG Views provides predefined geometric objects, while also allowing you to create graphic images through the use of numerous drawing tools. You have at your disposal a large selection of *resources* including fill and line patterns, colors, and font attributes. You use *palettes* to apply these resources to graphic objects and text.
- **Events**: Manage keyboard, mouse, and timer events.

- **Transformers**: Manipulate transformers, which are objects that compute geometric transformations to draw graphic objects while applying behavior such as movement, rotation, and zooming.

Other Services

- **ActiveX**: ActiveX generation and integration. These services will let you create an ActiveX with a simple button-click operation, and integrate any existing ActiveX in your GUI.
- **Scripting**: IBM ILOG Script, an implementation of the JavaScript™ scripting language for accessing graphic objects.
- **Internationalization**: Internationalized applications, using the POSIX locale model.
- **XML**: The XML parser allows you, for example, to read an XML file to create objects in a container, or to write an XML file from IBM ILOG Views data.

IBM ILOG Views Studio

IBM® ILOG® Views Studio is a powerful editor for creating, editing, and saving IBM ILOG Views objects. With IBM ILOG Views Studio you can design a panel, edit gadgets, draw and import graphic objects, group objects for editing, assign attributes and multilingual labels to objects, and, in general, create your GUI.

This executable is designed to be supplemented by dynamic modules for each specific use (Studio for Gadgets, Studio for Manager, and so on).

Main Studio Features

The main features of IBM ILOG Views Studio include:

- **WYSIWYG editor including drag-and-drop and testing mode.**
- **Drag-and-drop**.
- **Marking menus**: IBM ILOG Views Studio provides a new kind of pop-up menu: the Marking Menus. These are directional pop-up menus that let you choose one or several items by moving the mouse in a certain direction. They are very easy to use, and very efficient when you are used to them.

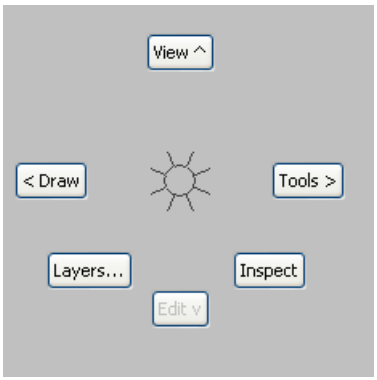


Figure 10 *IBM ILOG Views Studio Marking Menus*

- **MDI (Multiple Dockable Interface), dockable panels and toolbars.**
- **Platform look and feel**: With IBM ILOG Views Studio you can specify the platform look-and-feel with a simple menu selection.

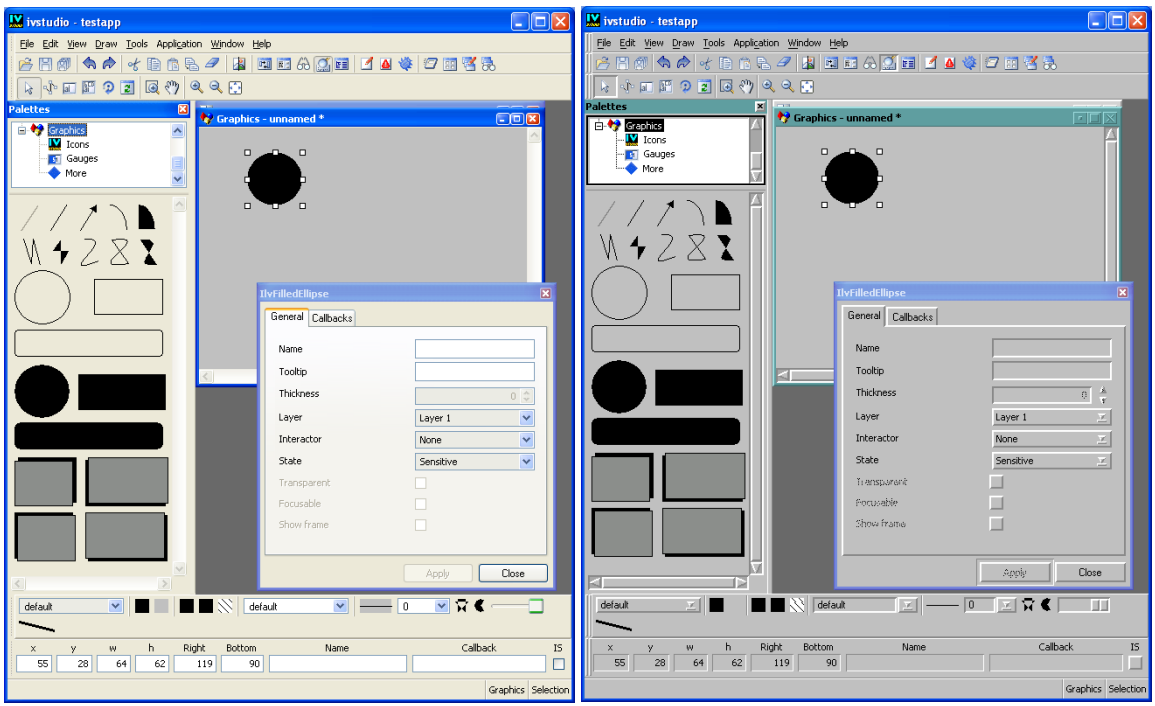


Figure 11 *IBM ILOG Views Studio Look-and-feel on Various Platforms: Windows® XP (left) and Motif® (right)*

- **Access to** IBM ILOG Script**.**
- **C++ code generation** when relevant.
- **Customization files**.
- **Extension by plug-ins**. You can supplement IBM ILOG Views Studio by adding plug-ins. Most IBM ILOG Views packages provide plug-ins that bring new graphic objects and their inspectors, data or code generation, drawing tools, and other extensions.
- **Recompilable**. In addition to plug-ins, IBM ILOG Views Studio provides libraries to link with your code. IBM ILOG Views Studio can then become your own application editor. This functionality requires that you have access to the *Gadgets*, *Manager*, and *Grapher* packages.

IBM ILOG Views Gadgets

With IBM® ILOG® Views Gadgets you have ready access to a large number of interface objects such as menus, buttons, scroll bars, and many other “gadgets” that can be easily viewed and manipulated with IBM ILOG Views Studio to build GUI panels.
IBM ILOG Views Studio for Gadgets allows you to construct your interface from predefined “gadgets” using simple drag-and-drop operations while generating C++ code for you to use to program your application.

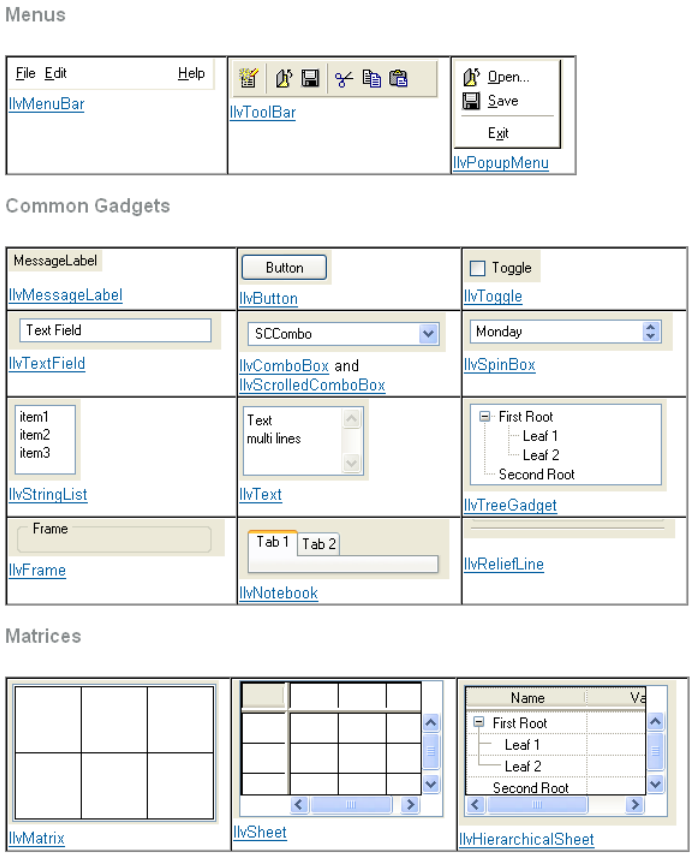


Figure 12 *Some Common Gadgets*

Main Gadgets Features

Here are just some of the Gadgets features.

- Gadgets are compliant with Microsoft® Windows®, systems and Motif® look-and-feel.
- Since IBM ILOG Views implements the gadgets without relying on a particular windowing system, you can use windows-specific features such as dockable toolbars and MDI (Multiple Document Interface) on Linux® and UNIX®.
- The callback executed upon gadget activation can be defined in IBM ILOG Script or C++.
- Gadgets include tables, trees, lists, dialogs, and others.
- C++ code is generated from IBM ILOG Views Studio to create your application GUI.

- Localization support. The gadgets support Japanese, Chinese, and Korean languages using the operating system locale support and the display system input methods, and perform the storage of messages as multibyte strings when necessary. A bidirectional version is also available for supporting languages such as Arabic (restricted to AIX® with Bidi support and Arabic Windows®).

Prerequisite Packages: *Foundation*, *Studio*.

IBM ILOG Views Application Framework

IBM® ILOG® Views Application Framework is a new library that lets you build a complete application very efficiently by using the Document/View architecture. This architecture allows several views to access the same document, and every modification in a view updates all the other views as necessary, while hiding the complexity of message sending between views and documents.

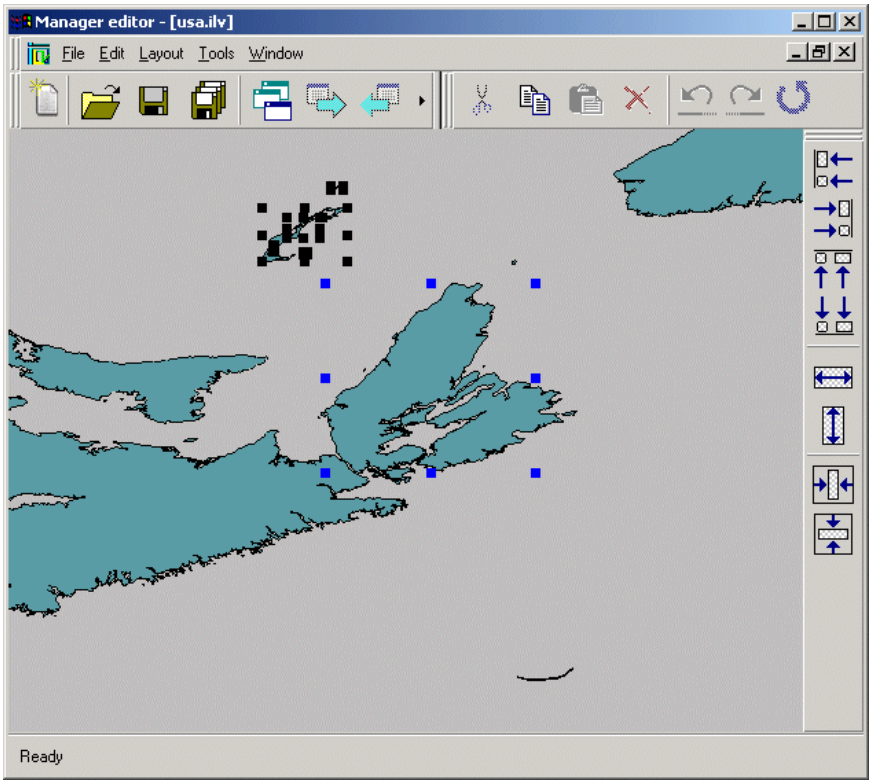


Figure 13 *Standard Application*

In addition to the Document/View architecture, Application Framework makes the development of user interfaces easy because it handles all the tedious coding tasks necessary

for MDI and dockable windows implementation. It also offers features such as drag-and-drop of a file in Microsoft® Windows® and recent-file-list management. Moreover, Application Framework lets you modify the configuration of your application's user interface, such as the menu and toolbars, as well as their contents and positions. This configuration is saved on a user basis, and then restored when the application is launched again.

It provides a tool called the *Application Framework Editor* (shown in Figure 14), which allows you to edit the application graphically: all menus, toolbars, actions, dynamic menus, and so on, are specified using this tool.

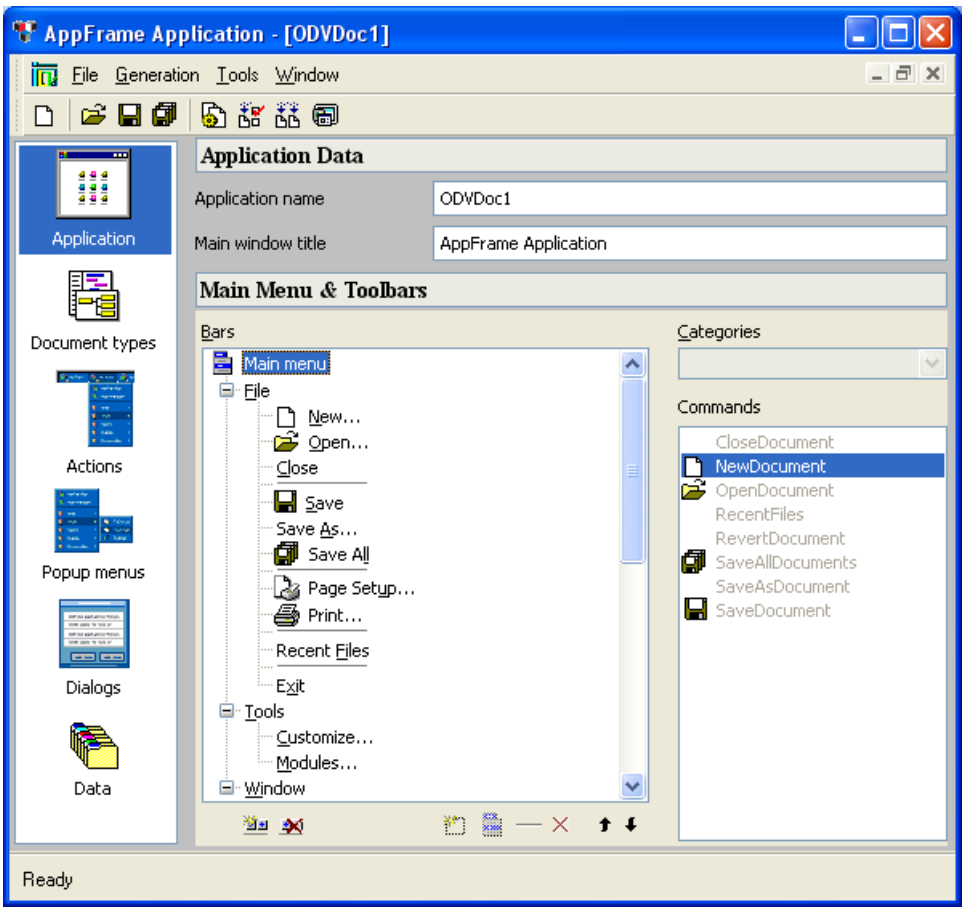


Figure 14 *The Application Framework Editor*

Main Application Framework Features

- Offers a Document/View implementation of the Model-View-Controller (MVC) architecture. This provides an easily accessible environment for MFC programmers and those familiar with up-to-date design patterns and the Java world.
- Supports the editing of several documents at the same time.
- Encapsulates dockable bar mechanisms, and stores (and restores) all dockable bar positions.
- Lets the user reorganize the contents of the menu and the toolbars at run time. All changes are saved and restored.
- Maintains the Most Recently Used file lists.
- Has an extended options language that lets users add their own application options.
- Changes at run time are saved in a user home file, using the same format.
- Provides automatic Undo/Redo/Repeat for Application Framework commands.
- Actions from toolbars or menus are directly sent to the active view or document. No callback is needed.
- An interface mechanism is implemented. It establishes the correspondence between a user name and a C++ method. Using these maps, user-defined C++ methods can be automatically exported for introspection, scripting, or COM.

Prerequisite Packages: *Foundation, Studio, Gadgets*.

IBM ILOG Views Manager

Higher-level functionality, allied with *managers*, goes beyond the operations possible with containers to actively govern the interaction and display of objects in a variety of contexts.

Managers add performance, functionality, and global control to a set of graphic objects. For example, using a manager, you can attach objects to different views or zoom in and out on a region of your display.

For further information see:

- *Basic Manager Features*
- *Manager Service Features*
- *Comparing Managers and Containers*

Basic Manager Features

With managers you can:

- Manage the relationship between sets of objects and sets of views.

- Manage layering of objects.

  Through managers, sets of objects can be displayed in separate views. Managers control layering, so that objects can be assigned to particular layers for display and behavior purposes. Each layer may be set visible or invisible in each view controlled by the manager.

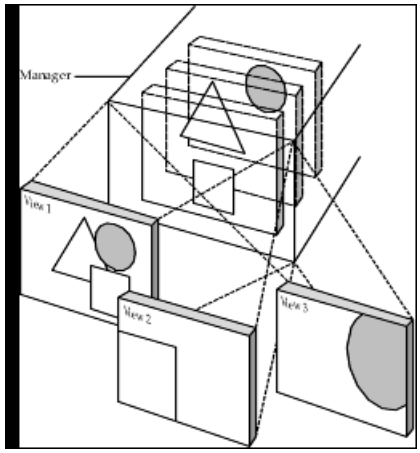


Figure 15 *Managers Allow Different Views to be Displayed on the Screen.*

- Manage different views of a set of objects.
- Attach a particular behavior to a view.

When using managers, interactors go beyond object-level behavior by providing a behavioral context for entire views that is independent of the objects. By activating a view interactor, you enter into an editing mode where you can select objects, draw view objects, drag-and-drop, and so forth.

Manager Service Features

Managers also provide other services, such as:

- Saving to files
- Keeping a command log for undoing and redoing events with minimum coding
- Using specialized data structures such as quadtrees to improve performance
- Advanced refresh functionality

Comparing Managers and Containers

The following table summarizes the differences between containers and managers:

Table 2 *Comparative Features of Managers and Containers*

Features	Managers	Containers
Multiviews	●	
Layering	●	
Saving	●	
Object Interactors	●	●
View Interactors	●	
Accelerators	●	●
Transformers (Zoom/Move/Rotate)	●	●
Double Buffering	●	●
Regional Buffering	●	●
Many objects	●	[1]
Gadget-oriented	[1]	●
Undo	●	

[1] Possible, but not the best choice.

Prerequisite Packages: *Foundation*, *Studio*.

IBM ILOG Views Grapher

The IBM® ILOG® Views Grapher is dedicated to the graphic representation of hierarchical and interconnected information. The Grapher is a customizable network editor, displaying nodes and links using any graphic object. Built on top of the IBM ILOG Views Manager package, the Grapher goes further, providing predefined links and allowing you to create new kinds of links graphically, using the IBM ILOG Views Studio Grapher extension.

Main Grapher Features

Some of the features of the IBM ILOG Views Grapher include:

- **Link management upon manipulation of nodes.**

- **Visual definition of links.**
- **Visual definition of connection points**: Visual definition of connection points on a node or on a link.

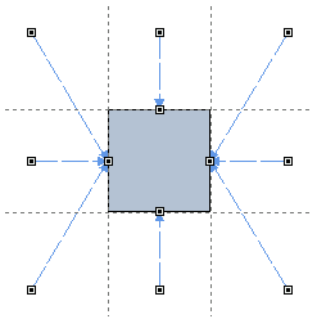


Figure 16 *Endpoint Location When No Connection Pin is Defined*

- **Rich selection of link types**: Customized links, link labels, straight, splined, arc, and free-form polylines; single (unidirectional), double (bidirectional), with automatic or fixed orientations.
- **Grapher pin editing**: When you define a grapher pin, it remains consistent even if the object is zoomed/unzoomed or rotated (the pin points are referenced to the center of gravity of the object).

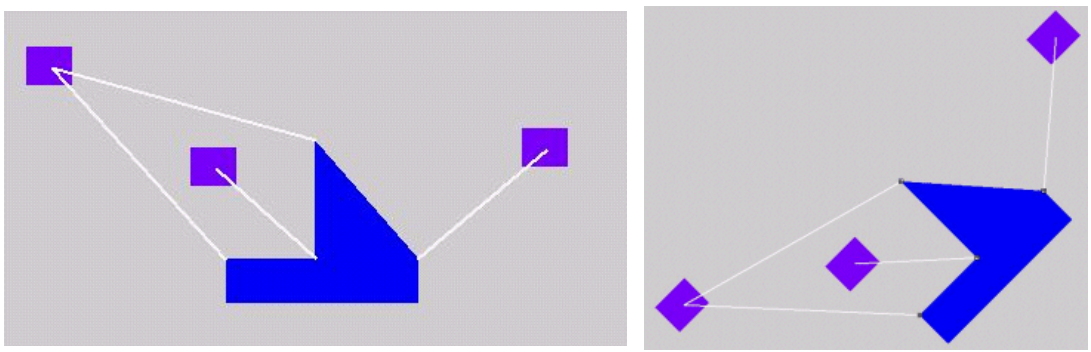


Figure 17 *The links are attached to the objects with grapher pins, and they remain consistent even if the objects are rotated*

Prerequisite Packages: *Foundation*, *Studio*, *Manager*.

IBM ILOG Views Prototypes

Application developers often need to define custom graphic objects to represent domain-specific application objects with which the user is able to interact. The IBM® ILOG® Views Prototypes package lets you create such custom, domain-specific graphic objects. These *business graphic objects* are created interactively using the Prototypes extension for IBM ILOG Views Studio. Prototypes let you build elaborate objects that interact with one another and attach complex behavior to these objects using a scripting language.

In creating a prototype with IBM ILOG Views Studio, you define the domain values and the look of your object, and attach behaviors. You can then create instances of your prototypes and use them in managers or containers just as you would do with basic IBM ILOG Views graphic objects. You can link application objects to their corresponding prototype. The display, synchronization, and user interaction is handled by the Prototypes package. You can edit and modify a prototype at any time: its instances will be automatically updated.

Creating a powerful, direct-manipulation interface for domain-specific objects becomes as easy as creating a forms-based interface for the same objects, but the interface is much more appealing and explicit to the user.

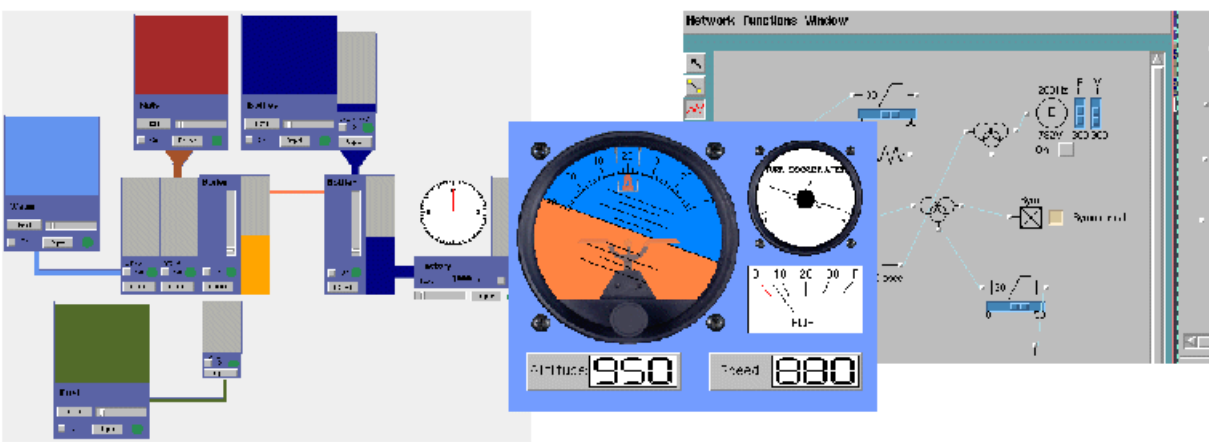


Figure 18 *Examples of Application Panels Built with Prototypes*

Prerequisite Packages: *Foundation, Studio, Manager, Grapher*.

IBM ILOG Views Gantt

The Gantt Chart package is included in the IBM® ILOG® Views base product. It manages the display and editing of scheduling data. Scheduling is the process of assigning resources to activities in time.

The Gantt, or time line, chart typically is a horizontal bar chart that shows the time relationships between tasks in a project.

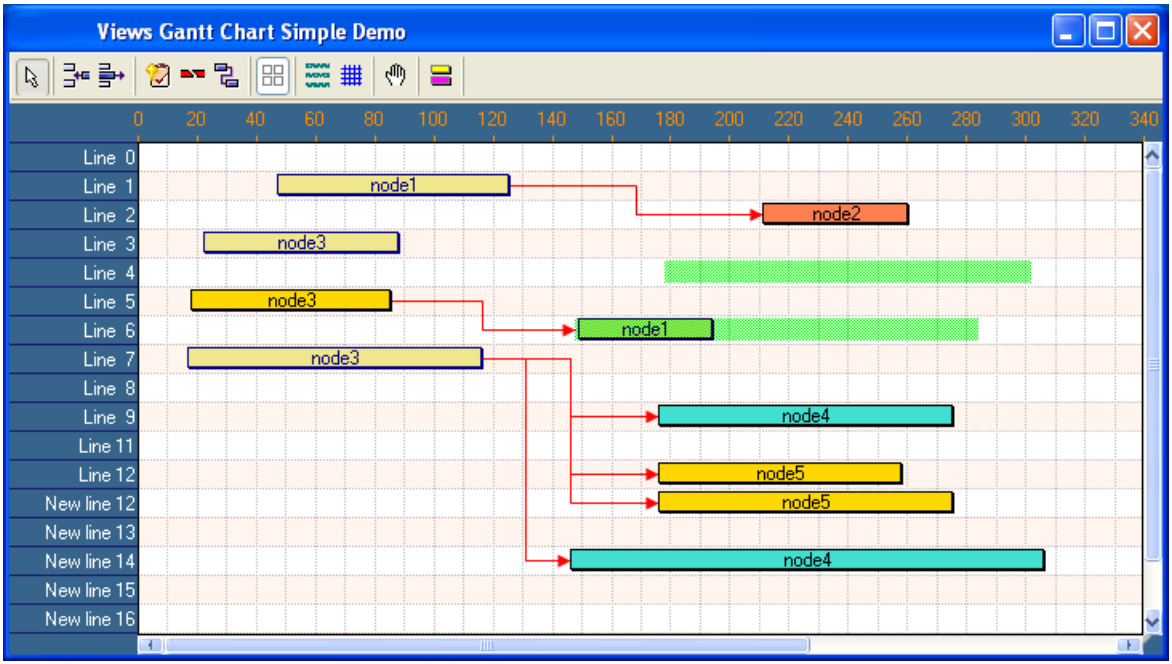


Figure 19 *Example of a Gantt Chart*

With IBM ILOG Views Gantt charts, you can create, edit, and manage for very large sets of data:

- Resources and tasks, the general inputs of a scheduling problem.
- Activities and subactivities, which have start time, end time, and resource capacities.
- Precedence constraints, showing task dependencies. These are defined by Gantt links which have a starting activity, ending activity, and a delay.

Here is an example of a calendar application:

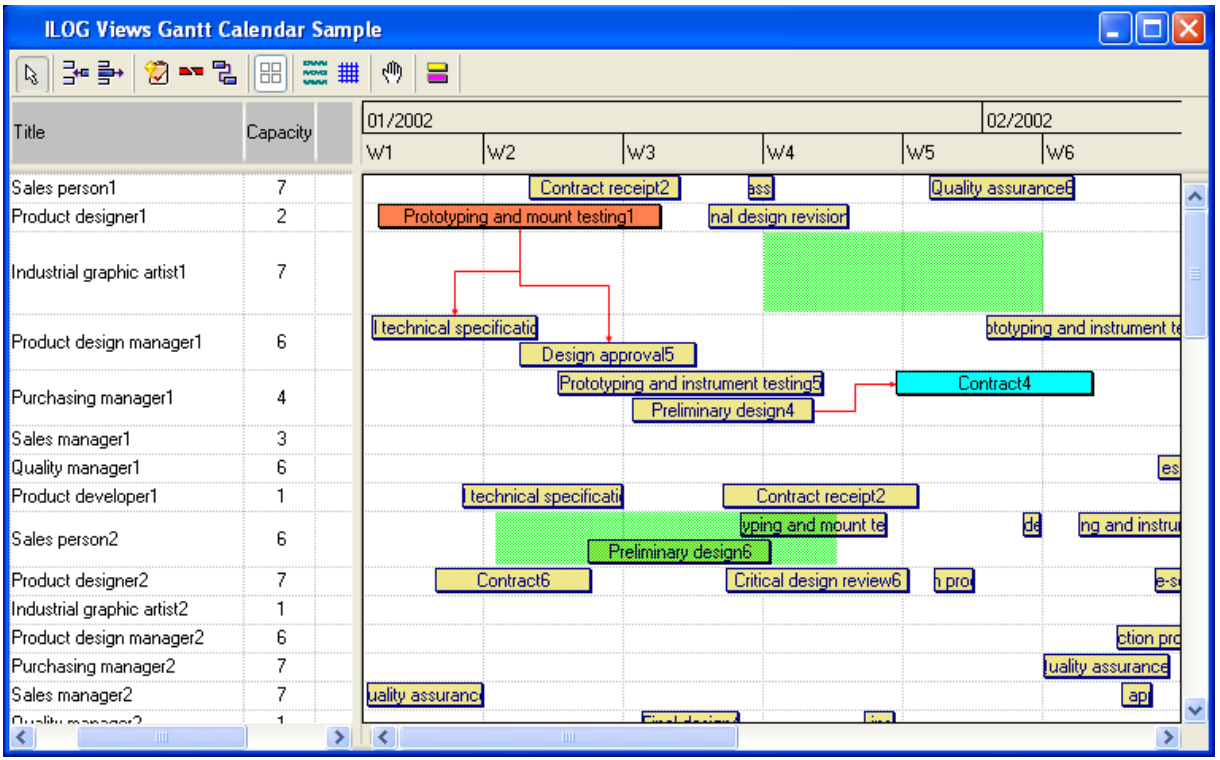


Figure 20 *A Gantt Chart Calendar Application*

For complete details see the IBM ILOG Views *Gantt User's Manual*.

Included in: IBM ILOG Views.

IBM ILOG Views Charts

The IBM® ILOG® Views Charts package is a powerful package for data visualization. This package allows you to display both predefined data or real-time data in highly customizable Cartesian and polar charts.

Main Charts Features

The Charts package includes these features:

- **Numerous chart representations**: Available chart representations are: scatter, polyline, polygon, marked polyline, bar, 3D bar, step, stair, high-low representation, pie, side by side bars, stacked bars, stacked 3D bars, and stacked polygons.

- **Cartesian or polar representation**: These allow you to display data in a standard way or in a circular way, respectively. You can jump easily from one representation to another.
- **Data-aware charts**: Perfectly adapted for real-time monitoring.
- **Automatic management of scrolling**: With several modes (stop, shift, cyclic).
- **Predefined interactors**: For zooming in, selecting data points, and other actions.
- **Logarithmic scales for each axis**.
- **Multiple ordinate scales**: You can add as many ordinate scales as you want (you can have only one abscissa axis).
- **Selectable scales orientation**: In Cartesian charts, the abscissa and ordinate scales can be oriented either horizontally or vertically. In polar charts, the circular abscissa scale can be oriented clockwise or counterclockwise, and the radial ordinate scales can be oriented at any angle.
- **Charts legend and grid management**.
- **Script integration for function definition**: You can use IBM ILOG Script, for example, to define a math function displayed on a chart.
- **Full integration in** IBM ILOG Views **Studio**: This allows you to create and customize charts without having to code.

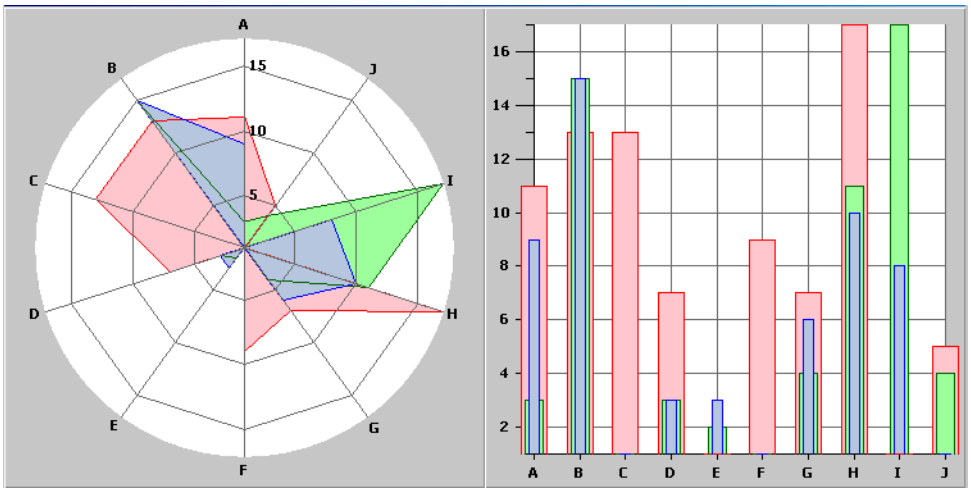


Figure 21 *Examples: A Polar Chart and a Cartesian Chart*

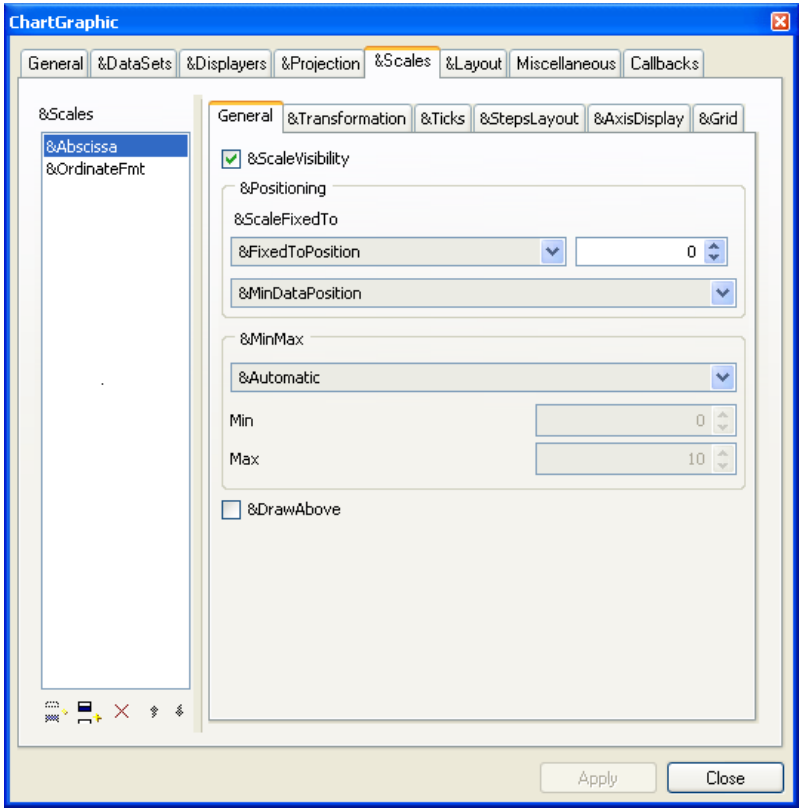


Figure 22 *Studio Chart Inspector Lets You Change Everything On a Chart*

Prerequisite Packages: *Foundation*, *Studio*.

IBM ILOG Views Maps

IBM ILOG Views Maps is the mapping package that supports multiple data formats, map projections, and advanced cartographic features for applications that require spatial data visualization and management.

Features of IBM ILOG Views Maps include:

- **Readers**: Oracle 8i, Shape, DTED, and CADRG use the reader framework architecture described in the next bullet; the **Shape** reader manages the attributes, the **CADRG** reader manages the Load-on-Demand, and the **DTED** reader manages the Load-on-Demand and the color model.

- **Reader Framework**: The IBM ILOG Views Maps architecture makes it easy to develop a new reader. It includes the concept of *map features* (mapping objects that do not have any graphic representation, compliant with OpenGIS simple features) and the concept of *renderers* (object that binds a map feature to a Views graphic object). Some predefined renderers are provided.
- **Projections**: A wide variety of map projection schemes (among the cylindrical, conic, and azimuthal families) to meet your cartographic requirements for most accurate rendering and distortion compensation. Geodetic datum allows a more precise positioning.
- **Load-on-Demand**: This feature lets you manage huge maps by loading the data only when they have to be displayed, with a cache management.
- **Predefined Controls**: Specific gadgets dedicated to map-based applications are also available. This includes a specific toolbar for map control (zoom/unzoom, panning tool, selection tool), compass, scale, and so on. They require the IBM ILOG Views Gadgets package.
- **Map Builder**: This application lets you build maps interactively by choosing data files, layer visibility filters, tiling for load-on-demand, and other features. The saved map can then be used in your application and all the load-on-demand and layer visibility will be handled automatically.

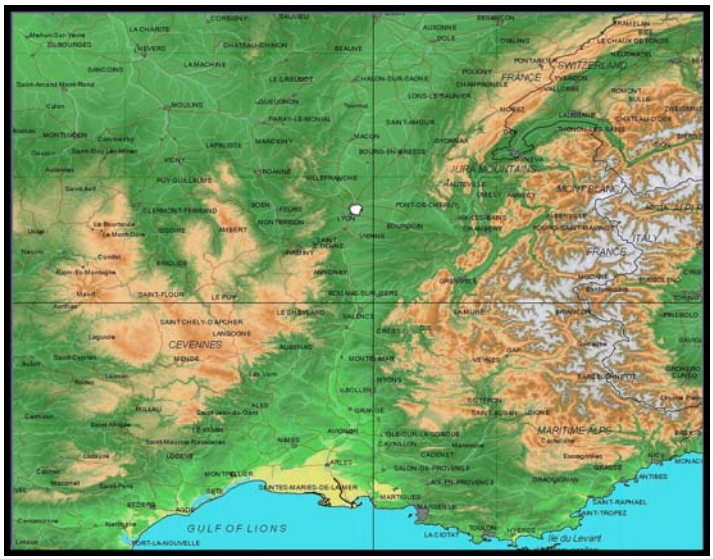


Figure 23 *Sample of Map Projection*

Prerequisite Packages: *Foundation*, *Studio*, *Manager*.

IBM ILOG Views Data Access

The IBM ILOG Views Data Access package (formerly InForm) is a visual environment for developing graphic-intensive applications. Data Access enhances the functionality of graphic objects created with IBM ILOG Views by linking them to data sources. The Data Access package provides SQL data sources and enables the creation of new data sources to connect your information stream data-aware graphic objects.

For further information see:

- *Main Data Access Features*
- *Data Access Extension for IBM ILOG Views Studio*

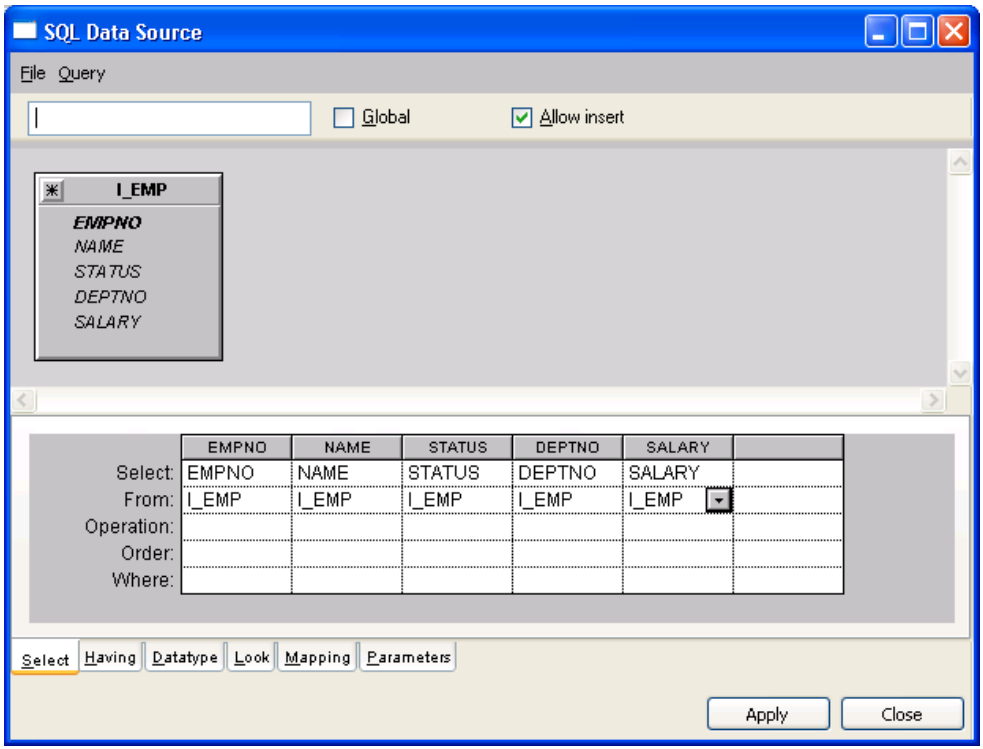


Figure 24 *Sample: SQL Data Source*

Main Data Access Features

- With Data Access you can rapidly create intuitive, user-friendly interfaces for client/server applications. These applications generally consist of forms, which contain a set of fields (text fields, check boxes, and so on). The values shown in these fields are the result of a mapping between the fields and the data from an external data system.

- The mapping between the fields in the graphical user interface and the external data system is bidirectional: data can be retrieved from the database and displayed in the fields, and it can be modified by the user and updated in the external data source for long term storage.
- Building applications is easy: simply drag GUI objects from Studio's palette and drop them into new panels. Data access is managed by data source objects: once connected to a database, tables and columns are displayed to build views. Simple point-and-click actions define classic SQL operations, all without writing a single line of code.

Data Access Extension for IBM ILOG Views Studio

Data Access provides a plug-in to IBM ILOG Views Studio. Studio is used both for building application interface panels and visually accessing the database. There is also a special interface that allows you to set up the connection with the external data source in a simple graphical way.

Prerequisite Packages: *Foundation*, *Studio*, *Gadgets*.

IBM ILOG Views Graph Layout

The IBM ILOG Views Graph Layout package provides automatic network layout for applications that need to display graphics composed of nodes and links. Any graphic object can be defined to behave like a node and be connected to other nodes via links, which themselves can have many different forms. The layouts are available directly in IBM ILOG Views Studio for easy use, and a complete API is also provided.

Graph Layout includes the following layout algorithms:

- Tree
- Hierarchical
- Orthogonal Links
- Random
- Bus

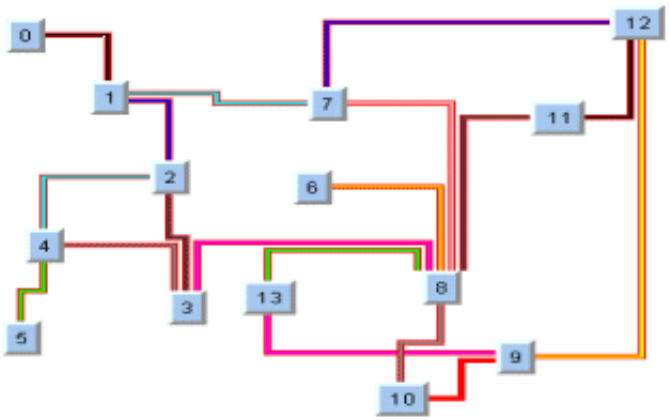


Figure 25 *Example of Orthogonal Link Layout*

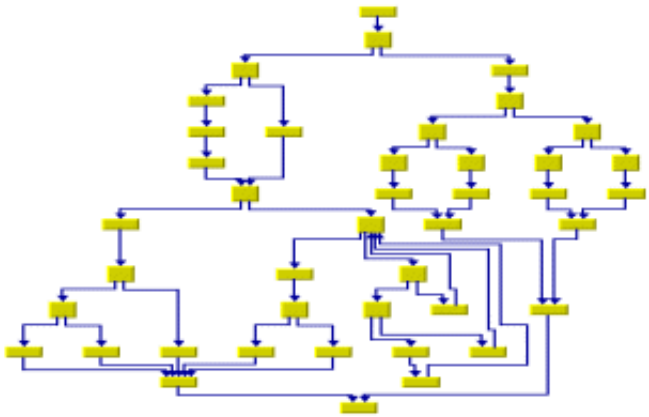


Figure 26 *Example Of Hierarchical Layout*

Prerequisite Packages: *Foundation, Studio, Manager, Grapher*.

Setting Up IBM ILOG Views

Before running IBM® ILOG® Views, you must install it in the appropriate directory and configure the IBM ILOG License Manager (ILM) provided with the product as explained in its documentation.

- See IBM ILOG Views *Disk Space and Memory* requirements.

- The directories created upon installation are listed in the *Distribution Structure*.

System Environments

Before using IBM ILOG Views, you must perform several setup steps, depending on your platform. Since the sections are independent, you can go directly to the one that applies to you:

- *Setup on UNIX*
- *Setup on Windows*

Microsoft® Windows® systems users will also want to refer to the additional setup information in:

- *Windows Compiling and Linking Options*

For the latest list of platforms on which you can use IBM ILOG Views, see the table in the `README` file delivered with the product.

Disk Space and Memory

Disk space and memory requirements for running IBM ILOG Views and IBM ILOG Views applications are as follows:

- **Disk space** The requirements depend on the components that are installed. For a full installation of one single platform with all options, you need about 450 MB on Microsoft® Windows® systems and about 350 MB on UNIX®.

  The disk space required for a complete application for your client is typically about 10 MB.

- **Memory** An IBM ILOG Views application usually requires less than 16 MB to run.

  Your system should have enough memory to support the operating system, the IBM ILOG Views applications and their data, as well as the compiler, if you plan to use it. It should also have some extra memory for executing IBM ILOG Views.

> ***Note:*** *IBM ILOG Views libraries are compiled to be optimized on x486 Intel processors. They work only with Ix486 or higher processors.*

Distribution Structure

When IBM ILOG Views is installed on your machine, several directories are created, some of them accompanied with a dedicated `README` file that you are advised to read. The following main directories are created:

- **`bin`** and its subdirectories, provide sources and data for certain delivered editors. The content depends on what packages are installed.

- **data** and its subdirectories provide panel description files (suffixed .ilv) and icons, used by the delivered IBM ILOG Views samples and editors. You should avoid modifying them.
- **doc** and its subdirectories provide the IBM ILOG Views getting started sources.
- **include** and its subdirectories provide all IBM ILOG Views class header files.
- **lib** and its subdirectories contain the IBM ILOG Views libraries.
- **samples** and its subdirectories provide sample coding to let you see particular aspects of the classes provided by IBM ILOG Views.
- **src** contains some IBM ILOG Views source files, some of which are analyzed in this manual.
- **studio** contains the necessary files for building and launching the IBM ILOG Views editor.
- **tools** and its subdirectories provide source files for tools that are extensions of IBM ILOG Views classes but not included in the library (see README for more details). Most of these tools can be executed on all platforms; hbm is for Microsoft® Windows® systems only.

In every main directory hierarchy, you find <system> subdirectories for your specific target systems and executable files. This structure guarantees the whole organization is shared and allows you to install different compiler systems, even different operating systems, on the same machine.

Setup on UNIX

This procedure explains how to configure IBM ILOG Views on a UNIX® station. You must have on hand the following installation information:

- **Installation Directory**: IBM ILOG Views components are normally installed in the /usr/ilog/ directory, including /usr/ilog/views*XX* (where *XX* stands for the version number). If you install IBM ILOG Views in another directory, replace subsequent references to the default directory with your actual installation directory.
- **System Name**: The name of your <system> directory can be found in the README file delivered with the product. For example, x86_RHEL4_3.4 or hp64_11_3.73. In the following instructions, replace <system> with your actual system name.

To configure IBM ILOG Views on UNIX, do the following:

1. Set the ILVHOME variable.

   The shell variable ILVHOME must be set to the IBM ILOG Views installation directory.

- If you use sh or one of its derivatives, such as ksh or bash, type:

```
$ ILVHOME=/usr/ilog/viewsXX
$ export ILVHOME
```

- If you use `csh` or `tcsh`, type:

```
% setenv ILVHOME /usr/ilog/viewsXX
```

2. Set up the shared library path variable.

   On most operating systems, the directories containing dynamic libraries must be added to an environment variable: `SHLIB_PATH` on HP-UX, `LIBPATH` on AIX®, `LD_LIBRARY_PATH` on the others.

- If using `sh` or one of its derivatives, type:

```
$ LD_LIBRARY_PATH=$ILVHOME/lib/<system>:$LD_LIBRARY_PATH
$ export LD_LIBRARY_PATH
```

- If using `csh` or `tcsh`, and the variable `LD_LIBRARY_PATH` is not already defined, type:

```
% setenv LD_LIBRARY_PATH $ILVHOME/lib/<system>
```

- If using `csh` or `tcsh`, and the variable `LD_LIBRARY_PATH` is already defined, type:

```
% setenv LD_LIBRARY_PATH $ILVHOME/lib/<system>:$LD_LIBRARY_PATH
```

This can be automated by putting the above commands in your shell startup file (either `.profile` if using `sh`, or `.login` if using `csh`).

Setup on Windows

This procedure explains how to configure IBM ILOG Views on Microsoft® Windows® systems. You must have on hand the following installation information:

- **Installation Directory**: IBM ILOG Views components are normally installed in the `C:\ILOG\` directory, including `C:\ILOG\VIEWS`*XX* (where *XX* stands for the version number). If you install IBM ILOG Views in another directory, replace subsequent references to the default directory with your actual installation directory.
- **System Name**: The name of your `<SYSTEM>` directory can be found in the `README` file delivered with the product. This is the name of the port that you will use. For example, if you are using Visual Studio 2008 on an x86 processor machine running Windows XP or Vista, your system name is `x86_.net2008_9.0`.
- **Subsystem Name**: The IBM ILOG Views libraries are provided in several versions for Windows. This is due to the different compiler options that were used when compiling the IBM ILOG Views libraries, such as whether these libraries are static (`.lib`) or dynamic (`.dll`). These different library types are contained in subdirectories of `LIB\<SYSTEM>` and are referred to by the name `<SUBSYSTEM>`. Each of the available versions (`x86_.net2003_7.1, x86_.net2005_8.0` and `x86_.net2008_9.0`) requires specific compiler flags. For more information on these compiler flags, see *Compiling and Linking with Microsoft Visual C++ .NET 2003, 2005 or 2008*.

In the instructions that follow, replace the strings <SYSTEM> and <SUBSYSTEM> in commands and directory names with your actual system and subsystem names.

To configure IBM ILOG Views on Windows, do the following:

1. Set the ILVHOME variable to the IBM ILOG Views installation directory.

- For Windows XP or Vista:

  Open the Configuration Panel *Performance and Maintenance* (or *System and Maintenance* on Vista)>*System*>*Advanced* tab, then click *Environment variable*. This displays a dialog box where you can define new variables. Use this dialog box to define the ILVHOME variable as:

  ```
  C:\ILOG\VIEWSXX
  ```

  Do not close the dialog box as you will need to use it again.

  Alternatively, the following lines are automatically added to the VIEWS.INI file:

  ```
  [Ilog Views]
  home=C:\ILOG\VIEWSXX
  ```

  The file is located in the directory where Windows NT, 2000, or Me is installed.

2. Set the PATH variable.

   The directory containing IBM ILOG Views DLLs must be added to the PATH variable. To do this, either copy the DLLs into a directory of the path, or:

- For Windows XP or Vista, use the System dialog box to define the PATH variable as:

  ```
  C:\ILOG\VIEWSXX\LIB\<system>\<subsystem>;%PATH%
  ```

  Click Apply and then OK to validate your changes.

3. Restart your computer to make sure your changes become effective.

Execution Requirements

You need to set the environment variable ILVHOME to the directory where the product has been installed to ensure that every provided binary works properly. Usually, you will set:

(csh/tcsh):

```
% setenv ILVHOME /usr/ilog/viewsXX
```

(sh/bash):

```
$ ILVHOME=/usr/ilog/viewsXX
$ export ILVHOME
```

(Windows Server 2003/XP/Vista):

```
C:\> set ILVHOME=C:\ILOG\VIEWSXX
```

Please check with your system administrator the exact location where IBM® ILOG® Views is installed.

Compilation

IBM ILOG Views is delivered with a set of libraries that are stored in a directory that depends on the type of machine you are using, the operating system it runs on and the compiler you plan to use.

This directory can be one of:

System name	Hardware	Operating System	Compiler
alpha_5.1_6.5	HP/Compaq (DEC) Alpha	Tru64 UNIX (OSF1) V5.1 or higher	C++ 6.5 or higher
hp32_11_3.73	HP 9700	HP-UX 11.11 or higher	HP ANSI C++ A.03.73 or higher using -mt -AA options
hp64_11_3.73	HP 9700	HP-UX 11.11 or higher	HP ANSI C++ A.03.73 or higher in 64bit mode (+DA2.0W) using -mt -AA options
ia64_hpux11_6.17	IA-64	HP-UX 11i v3	HP C++/ANSI A.06.17 in 64 bit mode
ia64-32_hpux11_6.17	IA-64	HP-UX 11i v3	HP C++/ANSI A.06.17
x86_RHEL4.0_3.4	x86	RedHat Enterprise Linux 4.0 (linux 2.4, glibc 2.3)	gcc3.4
x86-64_RHEL4.0_3.4	x86-64	RedHat Enterprise Linux 4.0 (linux 2.4, glibc 2.3)	gcc3.4
x86_sles10.0_4.1	x86	SUSE Linux Enterprise Server 10 (linux 2.6, glibc 2.4)	gcc4.1
x64_solaris10_11	x86-64	Solaris 10	Sun C++ 5.8 (aka Sun One Studio 11) in 64 bit mode
x86_solaris10_11	x86	Solaris 10	Sun C++ 5.8 (aka Sun One Studio 11)

System name	Hardware	Operating System	Compiler
x86_.net2003_7.1	x86	Windows Server 2003 / XP / Vista	Microsoft Visual C++ .NET 2003 (7.1)
x86_.net2005_8.0	x86	Windows Server 2003 / XP / Vista	Microsoft Visual C++ .NET 2005 (8.0)
x86_.net2008_9.0	x86	Windows 2000/Server 2003/XP/Vista	Microsoft Visual C++ .NET 2008 (9.0)
x64_.net2008_9.0	x64	Windows (64) Server 2003 / XP / Vista	Microsoft Visual C++ .NET 2008 (9.0 - 64 bits)
rs6000_5.1_6.0	RS6000	AIX 5.1 or higher	Visual Age 6.0 using standard C++ streams (-DIL_STD)
power32_aix5.2_7.0	PowerPC	AIX 5.2 or higher	Visual Age 7.0 using standard C++ streams (-DIL_STD)
ultrasparc32_8_6.2	Sun Ultra Sparc	Solaris 2.8 or higher	Sun C++ 5.3 (Forte 6.2) or higher
ultrasparc64_8_6.2	Sun Ultra Sparc	Solaris 2.8 or higher	Sun C++ 5.3 (Forte 6.2) or higher in 64bit mode (-xtarget=ultra -xarch=v9)
ultrasparc32_10_11	Sun Ultra Sparc	Solaris 10	Sun C++ 5.8 (aka Sun One Studio 11)
ultrasparc64_10_11	Sun Ultra Sparc	Solaris 10	Sun C++ 5.8 (aka Sun One Studio 11) in 64bit mode (-xtarget=ultra -xarch=v9)

It will be referred to as `<system>` in the following text. You may need to change the provided makefiles to match your own installation requirements:

- `VIEWSDIR` must be set to the directory where you unpacked the product (usually `/usr/ilog/viewsXX` on Unix or `c:\ilog\viewsxx` on Windows, where `XX` indicates the current version of ILOG Views);
- `XINC` and `XLIBS` should be set to the directory where your `X11` include and lib directories are located (only on Unix platforms);
- `MINC` and `MLIBS` should be set to the directory where your Motif include and lib directories are located (only on Unix platforms).

Windows Compiling and Linking Options

To compile the examples supplied with the libraries, you can use:

- Visual Studio

  A number of solution (`.sln`) and project (`.vcproj`) files for Visual Studio are provided for you to compile.

- The command line tool.

  A number of makefiles that can be used with the command `nmake` (Microsoft Visual C++) are provided.

A default definition file, `VIEWS.DEF`, is supplied in each `<SUBSYSTEM>` directory.

> ***Reminder:*** *The* `<SUBSYSTEM>` *directories are located in the directory specific to your system (*`x86_.net2003_7.1`, `x86_.net2005_8.0` *and* `x86_.net2008_9.0`*). For the value of* `<SUBSYSTEM>`*, see the section corresponding to the compiler you use.*

The packing alignment value for the structure and union members used to build the libraries is the default value of the compiler as described in *Compiling and Linking with Microsoft Visual C++ .NET 2003, 2005 or 2008*.

Compiling and Linking with Microsoft Visual C++ .NET 2003, 2005 or 2008

This section discusses the requirements for compiling your source files with Microsoft Visual C++ 2003 or higher.

Most of the binaries rely on the data files provided in the IBM ILOG Views library, and you will need to copy these files as well as the `.EXE` file when installing a program on another platform. You may also place these files in the resources bound to the executable.

The packing alignment for structure and union members is 8 bytes.

The libraries are standard C++ libraries and are provided in the following formats:

- DLL_MDA: dynamic multi-threaded runtime libraries with subsystem DLL_MDA
- STAT_MDA: static multi-threaded runtime libraries with subsystem STAT_MDA
- STAT_MTA: static multi-threaded runtime libraries with subsystem STAT_MTA
- STAT_STA: static single-threaded runtime libraries with subsystem STAT_STA (Microsoft Visual C++ .NET 2003 only)

Here are the compilation flags to use:

Table 3 *Compilation Flags*

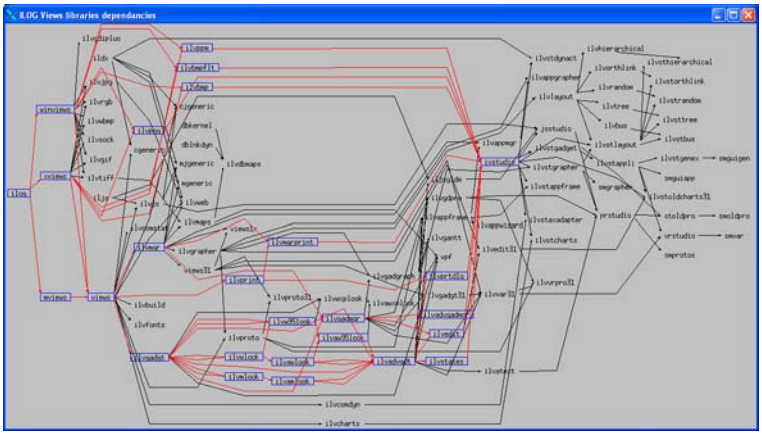
	Visual Studio .NET 2003	Visual Studio .NET 2005 and 2008
DLL_MDA	/MD /GX /GR /DILVSTD /DILVDLL	/MD /EHsc /GR /D_CRT_SECURE_NO_DEPRECATE / DILVSTD /DILVDLL
STAT_MDA	/MD /GX /GR /DILVSTD	/MD /EHsc /GR /D_CRT_SECURE_NO_DEPRECATE / DILVSTD
STAT_MTA	/MT /GX /GR /DILVSTD	/MT /EHsc /GR /D_CRT_SECURE_NO_DEPRECATE / DILVSTD
STAT_STA	/ML /GX /GR /DILVSTD	[not available]

Note: *When compiling files in debug mode, you must use the nondebug version of the runtime libraries and also delete the macro* `_DEBUG`*. (This is because IBM ILOG Views uses only nondebug runtime libraries.)*

Library Dependencies

Libraries that are included in your applications often depend on other libraries, which must also be included in your declarations.

The `libdeps` sample, found in `<installdir>\samples\layout\libdeps\index.html`, shows these dependencies graphically, as well as providing an example of how the Graph Layout package can be used to simplify the interpretation of linked data.



Warning: *This sample can only be executed if you have a license for the Graph Layout package.*

The following table is also provided as a quick way for you to identify the dependencies of the library you want to link into your application. (This table shows only the first level of dependency. Make sure that the libraries on which your library depends are not themselves dependent on other librairies.)

Table 4 *Library Dependencies*

Library	Dependence
<*displaylib*>[a]	ilog
views	<*displaylib*>
cgeneric	ildx views
cjgeneric	ilvjs ildx views
mgeneric	ildx ilvmgr
mjgeneric	ilvjs ildx ilvmgr
iljsgide	ilvadvgdt iljs
ilvadvgdt	ilvgadgt
ilvadvgdt	ilvgadgt ilvamlook ilvawlook ilvaw95look
ilvadvgadmgr	ilvadvgdt ilvgadmgr
ilvamlook	ilvadvgdt ilvmlook

Table 4 *Library Dependencies (Continued)*

Library	Dependence
ilvappframe	ilvadvgdt ilvgadmgr ilvgadgraph
ilvappgrapher	ilvappmgr ilvgrapher
ilvappmgr	ilvappframe ilvmgr
ilvappwizard	ilvappframe
ilvatext	ilvadvgdt
ilvaw95look	ilvawlook ilvw95look
ilvawlook	ilvadvgdt ilvwlook
ilvawxplook	ilvaw95look ilvwxplook
ilvbmp	xviews
ilvbmp	winviews
ilvbmpflt	xviews
ilvbmpflt	winviews
ilvbuild	views
ilvcomstat	views
ilvcomdyn	views
ilvedit	ilvadvgdt ilvgadmgr
ilvfonts	views
ilvgadgt	views
ilvgadgt	views ilvmlook ilvwlook ilvw95look
ilvgadgraph	ilvgrapher ilvgadmgr
ilvgadmgr	ilvgadgt ilvmgr
ilvgantt	ilvgadgraph ilvadvgdt
ilvgdiplus	winviews
ilvgdpro	ilvproto ilvgadgraph
ilvgif	xviews

Table 4 *Library Dependencies (Continued)*

Library	Dependence
ilvgif	winviews
ilvgrapher	ilvmgr
ilvjpg	xviews
ilvjpg	winviews
ilvjs	iljs views
ilvmgr	views
ilvmgrprint	ilvprint ilvmgr
ilvmlook	ilvgadgt
ilvpng	xviews
ilvpng	winviews
ilvppm	xviews
ilvppm	winviews
ilvprint	views
ilvproto	views ilvgrapher
ilvprtdlg	ilvprint ilvadvgdt
ilvrgb	xviews
ilvrgb	winviews
ilvstappframe	ilvappframe ivstudio
ilvstappli	ilvstgadget
ilvstates	ilvadvgdt
ilvstaxadapter	ivstudio ilvcomdyn
ilvstbus	ilvstlayout ilvbus
ilvstcharts	ivstudio ilvcharts
ilvstdynact	ildx ivstudio
ilvstgadget	ivstudio

Table 4 *Library Dependencies (Continued)*

Library	Dependence
ilvstgenex	ilvstappli
ilvstgrapher	ivstudio ilvgadgraph
ilvsthierarchical	ilvstlayout ilvhierarchical
ilvstlayout	ilvlayout ivstudio ilvstgrapher
ilvstorthlink	ilvstlayout ilvorthlink
ilvstrandom	ilvstlayout ilvrandom
ilvsttree	ilvstlayout ilvtree
ilvtiff	xviews
ilvtiff	winviews
ilvw95look	ilvgadgt ilvwlook
ilvwbmp	xviews
ilvwbmp	winviews
ilvwlook	ilvgadgt
ilvwxplook	ilvgadgt ilvw95look
ivstudio	ilvedit ilvstates ilvadvgadmgr ilvpng ilvbmp ilvppm ilvbmpflt
jsstudio	ivstudio ilvjs iljsgide
mviews	ilog
prstudio	ilvgdpro jsstudio ilvatext ilvstgrapher
smgrapher	ilvstgrapher
smguiapp	ilvstappli
smguigen	ilvstgenex
smoldpro	stoldpro
smprotos	prstudio
smvar	vrstudio
stoldpro	prstudio ilvstgadget

Table 4 *Library Dependencies (Continued)*

Library	Dependence
views	winviews
views	xviews
vrstudio	ilvstappli ilvvar31
winviews	ilog
xviews	ilog
ilvcharts	views
ilvbus	ilvlayout
ilvhierarchical	ilvlayout
ilvlayout	ilvgrapher
ilvorthlink	ilvlayout
ilvrandom	ilvlayout
ilvtree	ilvlayout
iljs	*(none)*
views31	ilvmgr
ilvgadgt31	ilvgadmgr ilvadvgdt views31
ilvedit31	ilvedit views31
ilvproto31	ilvproto views31
ilvvar31	ilvgadmgr ilvgrapher ilvadvgdt ilvgadgt31
ilvvrpro31	ilvgdpro ilvproto ilvvar31
ilvstoldcharts31	ilvstappli ivstudio ilvedit31
views1x	ilvmgr views31
ilvmaps	ilvmgr ilvtiff
ilvdbmaps	dblnkdyn dbkernel ilvmaps
views	mviews

[a] *<displaylib>* corresponds to the library ensuring the interface with the windowing system: `winviews` on Windows platforms, `xviews` on UNIX/X11, and `mviews` on UNIX/X11/Motif

Motif and Shared Libraries

The use of libmviews (Motif based) is deprecated in shared library format.

Since version 4.0, all shared libraries provided by IBM® ILOG® Views are built using libxviews and are incompatible with libmviews. libmviews is only provided as a static library and can only be used with the static version of other IBM® ILOG® Views libraries.

Library Build Information (Unix platforms only)

If a problem occurs while running your applications, you can retrieve the information on the platform that was used to build the library by using the shell script called ilvversion.sh, located in the IBM ILOG Views root directory.

Go into the specific static subplatform subdirectory of `lib` and type:

```
../../../ilvversion.sh
```

Two lines that have the following form will be printed out (these lines may be swapped):

IlvVersion: X.X

```
IlvBuild: OS - platform - compiler - display revision
```

The shell script locates the strings '`IlvBuild`' and '`IlvVersion`' in the library file `libxviews.a` and prints out the result.

If you have installed multiple platforms in the directory ILVHOME, you can check each of the platforms:

```
$ cd $ILVHOME
$ ./ilvversion.sh platform
```

where platform is the system identifier of the IBM ILOG Views library (that is, the `<system>`, one of `hp32_11_3.73`, power32_aix5.2_7.0, and so on).

(but it does not support Oracle 8.x and Open Ingres).

Licensing

If you have a file called `$ILVHOME/access.ilm` on Unix or `%ILVHOME%\access.ilm` on Windows, then this is an evaluation copy of IBM® ILOG® Views.

You must set the environment variable `ILOG_LICENSE_FILE` to this path name to run the IBM ILOG Views applications:

(csh/tcsh):

```
% setenv ILOG_LICENSE_FILE $ILVHOME/access.ilm
```

(sh/bash):

```
$ ILOG_LICENSE_FILE=$ILVHOME/access.ilm
$ export ILOG_LICENSE_FILE
```

(Windows Server 2003/XP/Vista):

```
C:\> set ILOG_LICENSE_FILE=%ILVHOME%\access.ilm
```

Note for Unix Users

On Unix® platforms, when you plan to link your application with IBM® ILOG® Views, you have to decide whether or not it will include pure Motif® code. If this is the case, then you need to link with `libmviews`.

If you need your application to be a pure Xlib application, then you need to link with `libxviews`.

Never link with both, and always link with one of these two. The library "views" is likely to be necessary for every application.

Note for Windows Server 2003 / XP / Vista Users

The libraries are provided in several versions on the Windows® versions that are provided in subdirectories of `lib\<system>`. This is due to different versions of the run-time system libraries and to the DLL. Each version needs specific compiler flags as described below:

- `x86_.net2003_7.1`:
  - `stat_sta`: Static library in single thread with static run-time library using new IOStreams. Flags: `/GX /GR /ML /DIL_STD` (Single Threaded).
  - `stat_mta`: Static library in multithread with static run-time library using new IOStreams. Flags: `/GX /GR /MT /DIL_STD` (Multithreaded).
  - `stat_mda`: Static library in multithread with dynamic run-time library using new IOStreams. Flags: `/GX /GR /MD /DIL_STD` (Multithreaded DLL).
  - `dll_mda`: Dynamic library in multithread with dynamic run-time library using new IOStreams. Flags: `/GX /GR /MD /DIL_STD /DILVDLL` (Multithreaded DLL).
- `x86_.net2005_8.0, x86_.net2008_9.0 and x64_.net2008_9.0:`

- `stat_mta`: Static library in multithread with static run-time library using new IOStreams. Flags: `/EHsc /GR /MT /DIL_STD` (Multithreaded).
- `stat_mda`: Static library in multithread with dynamic run-time library using new IOStreams. Flags: `/EHsc /GR /MD /DIL_STD` (Multithreaded DLL).
- `dll_mda`: Dynamic library in multithread with dynamic run-time library using new IOStreams. Flags: `/EHsc /GR /MD /DIL_STD /DILVDLL` (Multithreaded DLL).

Notes:

1. *You must also use the flag* `/DILJSTDH` *if you want to use IBM LOG Script in these modes.*
2. *You must also link with the system libraries* `wsock32.lib` *and* `imm32.lib`.

Note for Microsoft Visual C++ Users

For all your projects, make sure that `wsock32.lib` and `imm32.lib` are specified in the link command line.

To debug your program more easily, add the following lines to the section `[AutoExpand]` of the file

x86_.net2003_7.1, x86_.net2005_8.0, x86_.net2008_9.0 or x64_.net2008_9.0:
`%INSTALLDIR%\Common7\Packages\Debugger\autoexp.dat`

```
; from Ilog Views
IlvRect =x = <_orig._x,d> y = <_orig._y,d> width = <_w,u> height = <_h,u>
IlvPoint =x = <_x,d> y = <_y,d>
IlvTransformer =x11 = <_x11,g> x12 = <_x12,g> x21 = <_x21,g> x22 = <_x22,g> x0
= <_x0,g> y0 = <_y0,g>
```

These lines let you display the values of the objects that these classes define (instead of their address) in the tooltip that appears when the mouse is positioned over a variable.

You may get a series of link errors when using the Visual C++ 6.0 integrated development environments to build your project in Debug mode.

In Debug mode, to be able to link your application with Visual C++ and the libraries provided in IBM® ILOG® Views, follow these instructions:

`x86_.net2003_7.1, x86_.net2005_8.0, x86_.net2008_9.0 or x64_.net2008_9.0:`

- in Microsoft Visual Studio.NET, select your projects in the "Solution Explorer" window,
- right-click on the selection and choose the "Properties" item,

- in the "Property Pages" window, select "Configuration Properties" -> "C/C++" -> "Preprocessor",
- in the "Preprocessor Definitions" field, remove "_DEBUG",
- recompile and relink the application.

Various Versions

The libraries are stored in subdirectories of the directory `lib/<system>` on UNIX® and `lib\<system>` on Windows® (Server 2003/XP/Vista).

There are several library files (static and shared equivalent):

Unix platforms:

- `libxviews`: Pure Xlib code of IBM® ILOG® Views.
- `libmviews`: Motif-dependent code of IBM ILOG Views.

Windows Server 2003/XP/Vista platforms:

- `winviews.lib`: Windows-dependent code of IBM ILOG Views.

For all platforms (the name should be prefixed with `lib` on UNIX platforms, and have the `.lib` extension on Windows platforms):

- Foundation Package
  - `ilog`: ILOG utility classes (arrays, lists, and so on).
  - `views`: Core library. Includes Standard 2D graphics.
  - `ilvgadgt`: Core Gadgets library.

    This library is provided with the Foundation Package, but can be used and deployed only if you have the "IBM ILOG Views" or "IBM ILOG Views Controls" license. This library contains the following classes: `IlvMessageLabel`, `IlvFrame`, `IlvButton`, `IlvToggle`, `IlvColoredToggle`, `IlvComboBox`, `IlvTextField`, `IlvNumberField`, `IlvPasswordField`, `IlvPopupMenu`, `IlvToolBar`, `IlvMenuBar`, `IlvScrollBar`, `IlvSlider`, `IlvGadgetItem`, `IlvMenuItem`, `IlvGadgetContainer`, `IlvScrolledView`, `IlvDialog`, `IlvIMessageDialog`, `IlvIInformationDialog`, `IlvIErrorDialog`, `IlvIWarner`, `IlvIQuestionDialog`, `IlvGadgetContainerRectangle`, `IlvSCGadgetContainerRectangle`, `IlvSCViewRectangle`.
  - `ilvmlook`: Motif® Look and Feel library.

- `ilvwlook`: Windows 3.11 Look and Feel library.
- `ilvw95look`: Windows 95 Look and Feel library.
- `ilvwxplook`: Windows XP Look and Feel library (Windows only).
- `ilvprint`: Printing support library.
- `ilvbmpflt`: Bitmap Filters library.
- `ilvbmp`: BMP bitmap streamer.
- `ilvwbmp`: WBMP bitmap streamer.
- `ilvpng`: PNG bitmap streamer.
- `ilvjpg`: JPG bitmap streamer.
- `ilvppm`: PBM-PPM bitmap streamer.
- `ilvtiff`: TIFF bitmap streamer.
- `ilvrgb`: SGI RGB bitmap streamer.
- `iljs`: IBM ILOG Script library.
- `ilvjs`: IIBM ILOG Views Script implementation.
- `iljsgide`: IBM ILOG Script debugger interface.
- `ilvbuild`: Resource-handling classes for binaries.

◆ Gadgets Package (Controls)

- `ilvadvgdt`: Advanced Gadgets library (IlvMatrix, IlvTreeGadget, ...).
- `ilvadvgadmgr`: Manager classes using advanced gadgets.
- `ilvatext`: IlvAnnotext and related classes.
- `ilvedit`: Inspector classes (Color chooser, Font chooser, ...).
- `ilvstates`: States library.
- `ilvamlook`: Advanced Motif Look and Feel library.
- `ilvawlook`: Advanced Windows 3.11 Look and Feel library.
- `ilvaw95look`: Advanced Windows 95 Look and Feel library.
- `ilvawxplook`: Advanced Windows XP Look and Feel library (Windows only).
- `ilvprtdlg`: Printing dialogs library.

◆ Application Framework Package (Controls)

- `ilvappframe`: Application Framework library.

- `ilvappmgr`: Application Framework classes using the Manager package (`IlvDvManagerDocument` and related classes).
- `ilvappgrapher`: Application Framework classes using the Grapher package (`IlvDvGrapherDocument` and related classes).
- `ilvappwizard`: Application Framework library for wizard.

◆ Manager Package (2D Standard)

- `ilvmgr`: Manager library (`IlvManager` and related classes).
- `ilvgadmgr`: Manager classes using gadgets (`IlvGadgetManager` and related classes).
- `ilvmgrprint`: Manager Printing support library.

◆ Data Access Add-On

- `dataccess`: Core Data Access libraries (was '`inform`' in ILOG InForm 3.0).
- `dbaccess`: Relational databases libraries (was '`dbinform`' in ILOG InForm 3.0).
- `dbchart`: Charts-based Data Access classes library.
- `dbgadget`: Gadgets-based Data Access classes library.
- `dbgantt`: Gantt Chart-based Data Access classes library.
- `dbgraphe`: Grapher-based Data Access classes library.
- `dbsqlgad`: Relational Databases Gadgets classes library.

◆ Grapher Package (Advanced 2D)

- `ilvgrapher`: Grapher library (`IlvGrapher` and related classes).
- `ilvgadgraph`: Grapher classes using gadgets (`IlvSCGrapherRectangle`).

◆ Prototypes Package (Advanced 2D)

- `ilvproto`: Prototypes Base library.
- `ilvgdpro`: Prototype classes using gadgets.

◆ Charts Add-On

- `ilvcharts`: Charts library (`IlvChartGraphic` and related classes).

◆ Graph Layout Add-On

- `ilvlayout`: Core Graph Layout library.
- `ilvbus`: Bus layout library.
- `ilvhierarchical`: Hierarchical layout library.

- `ilvorthlink`: Orthogonal Link layout.
- `ilvrandom`: Random layout library.
- `ilvtree`: Tree layout library.

◆ Gantt Add-On

- `ilvgantt`: Gantt Chart library (`IlvGanttChart` and related classes).

◆ Maps Add-On

- `ilvmaps`: Core Maps library.
- `ilvdbmaps`: Database-based maps library.

Index

M

N

P

S

U

W