IBM ILOG JViews Maps V8.6

Introducing JViews Maps

Copyright notices

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.

IBM ILOG JViews Maps copyright

For further copyright information see ***<installdir>*** **/`license/notices.txt`**

Table of contents

About JViews Maps

Tells you about JViews Maps in general, describes background cartographic concepts and presents the key benefits and some typical uses of JViews Maps.

In this section

Overview
Provides a short description of JViews Maps.

Cartography
Introduces the cartographic concepts that any mapping product must address.

The added value of JViews Maps
Presents the key benefits of JViews Maps.

Typical uses of JViews Maps
Gives examples of use of JViews Maps.

Developing with the SDKs
Explains how you can extend your development of maps with the software development kit.

Overview

This is your starting point for finding out about JViews Maps.

IBM® ILOG® JViews Maps is built on IBM® ILOG® JViews Diagrammer.

JViews Maps comes with a set of easy-to-use GUI-based tools for rapidly developing dynamic map applications and designing symbols to be placed on such background maps.

JViews Maps includes the rule-based presentation layer based on the Styling and Data Mapping (SDM) features of IBM® ILOG® JViews Diagrammer. In purchasing JViews Maps you not only acquire a full range of advanced mapping features, but also the versatile look-and-feel configuration facilities and powerful graphics framework of JViews Diagrammer for easily changing visual representations as internal application data changes.

You should read Before you start to understand the purpose and relationship of each tool.

This section looks at maps used as backgrounds to applications and provides information about the IBM® ILOG® JViews Software Development Kits (SDKs).

Cartography

Introduces the cartographic concepts that any mapping product must address.

In this section

What is a map?
Presents the different categories of map.

Spatial reference systems
Introduces and illustrates what a spatial reference system is.

Coordinate systems
Introduces and illustrates coordinate systems.

Map projections
Presents and illustrates the different map projections.

Map data formats
Describes briefly map data formats and links to more detailed information about map formats.

What is a map?

In general, we all have experience of paper travel maps or electronic maps as found in Google Maps™ or Google Earth™ . Unlike paper maps, electronic maps provide the ability to display the data you need, the way you need it.

There are three main categories of map:

- Vectorial maps are a set of points, lines or areas and can contain additional data such as road names. You can decide on the look and feel of the display.
- Raster maps as satellite views. These provide realistic views of the earth, but are often heavy and can be confusing.
- Elevation maps that provide altitude information for any place on the globe.

All three categories can be mixed together in a single map display.

Spatial reference systems

Different reference systems exist, but georeferencing services encompass the spatial positioning of objects on the surface of the globe and the superimposition of data from different sources. They provide a framework for modeling the earth.

The earth used to be modeled as a sphere, but in reality it has an irregular shape and the modeling framework has to take into account the irregularities.

The earth modeled as an irregular shape

Coordinate systems

Real object coordinates can be expressed in different systems. The main systems are:

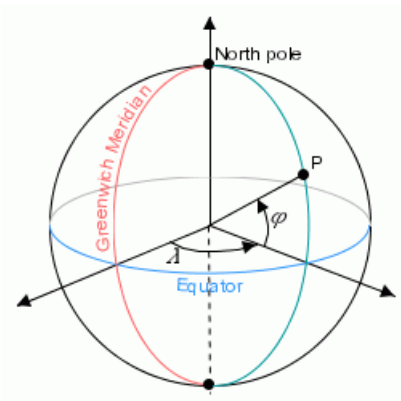
- *Geographic coordinate system*
- *Geocentric coordinate system*
- *Projected coordinate system*

Geographic coordinate system

In a geographic coordinate system, latitude and longitude are used and are represented by two angles from the center of the earth:

- Latitude = south to north +90 or 90N = north pole, -90 or 90S = south pole, and 0 is on the equator.
- Longitude = east to west. 0 is in England.

An example of coordinates using the geographic: longitude and latitude with optional altitude is shown in *Geographic coordinate system*.



Geographic coordinate system

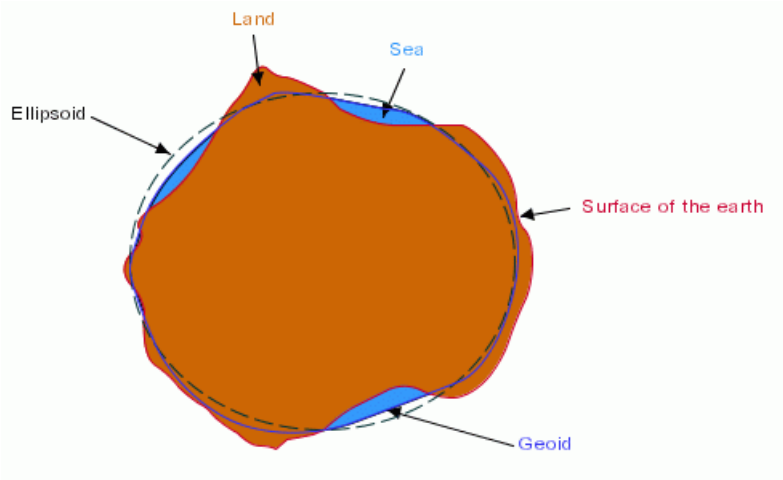
In a geographic coordinate system the angular distance is expressed in degrees and minutes, for example, New York:

- Latitude 40 Degrees 59'N
- Longitude 73 Degrees 39'W

A geographic coordinate system is defined by:

- The unit system of coordinates (angular unit and linear unit).
- The prime meridian:
  - For example, the Greenwich meridian.

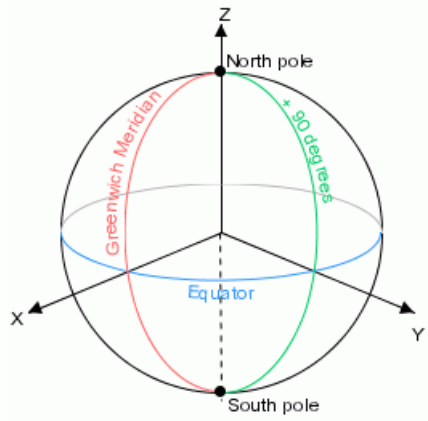
- The horizontal datum:
  - A horizontal datum is used to represent the shape of the earth.
  - The datum is defined by an ellipsoid (the default is a sphere) and a translation of the ellipsoid (position of the ellipsoid relative to the center of the earth), see *Horizontal Datum*:



Horizontal Datum

Geocentric coordinate system

An example of coordinates using the geocentric: three-axis Cartesian system with the center of the earth as the origin is shown in *Geocentric Coordinate System*:



Geocentric Coordinate System

A three-axis Cartesian system is defined by:

- Its origin, the center of the earth.
- Its x-axis, which lies in the plane containing the equator and which is oriented towards the Greenwich meridian.
- Its y-axis, which lies in the plane containing the equator and which is oriented towards the longitude 90 Degrees East of Greenwich.
- Its z-axis, which corresponds to the polar axis and which is oriented northwards.

A geocentric coordinate system is defined by:

- The unit system of coordinates (linear units) on the axes.
- The horizontal datum.

Projected coordinate system

A projected coordinate system is a representation of the earth on a 2-D surface. Units are attached to each axis of a coordinate system. For example, for geographic coordinates:

- x- and y-coordinates expressed in degrees
- z-coordinate expressed in meters

A kernel unit is defined for each type of unit.

- Length in meters
- Angles in radians

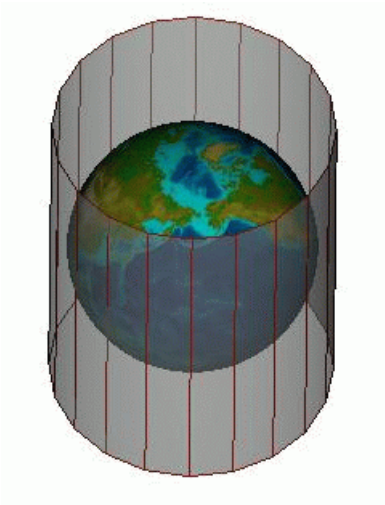
Coordinate systems are used to display maps and a map projection is used to reduce the dimensions to two. A projected coordinate system is defined by:

- The associated geographic coordinate system
- The projection
- The unit system of projected coordinates

Map projections

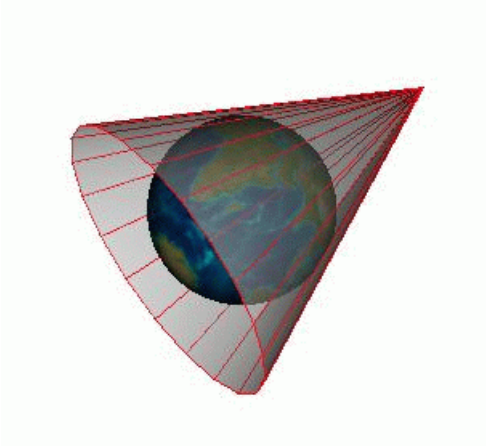
The Earth is a globe and displaying it on a plane requires complex mathematical transformations known as projections. Map projections are attempts to portray the surface of the earth or a portion of the earth on a flat surface. Some distortions of conformality, distance, direction, scale, and area always result from this process. Some projections minimize distortions in some of these properties at the expense of maximizing errors in others. Some projections are attempts to only moderately distort all of these properties. Map projections convert geographic points, represented by a longitude and a latitude, to Cartesian coordinates in a planar coordinate system.

- Cylindrical projections are precise around the equator



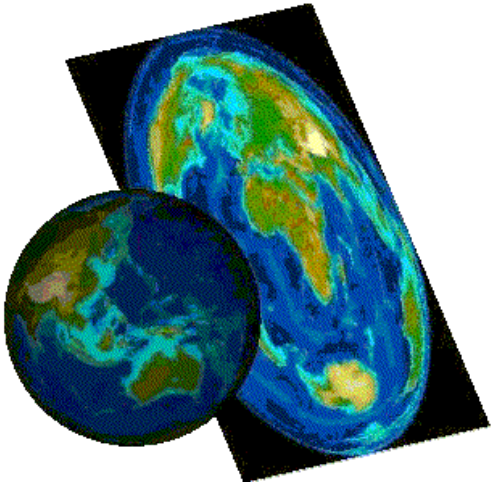
Cylindrical projection

- Conic projections are good for regions other than the equator



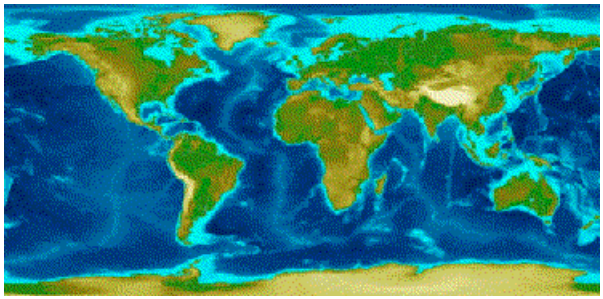
Conic projection

- Azimuthal projections are used for regions around the Poles

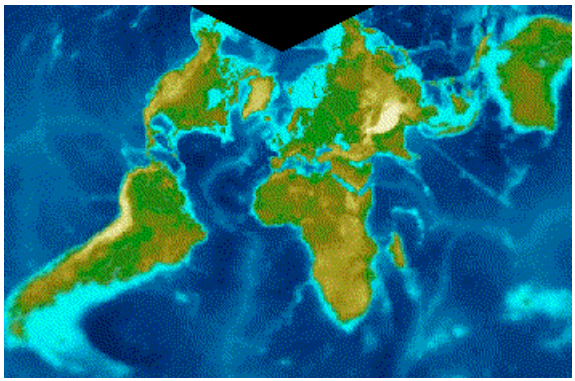


Azimuthal projection

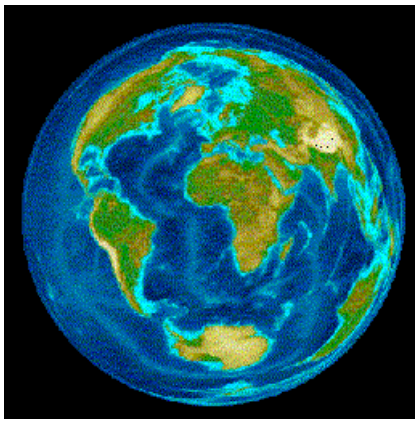
You use a particular projection depending on the area of the globe that interests you. Mercator preserves the heading and is useful for navigation. Lambert (conic) distorts distance. Lambert 1 is used for Northern France and Lambert 2 for Southern France.



Equidistant cylindrical projection



Lambert conformal conic projection



Lambert azimuthal equal area projection

The ideal characteristics for a projection would be:

- Meridians and parallels crossing at right angles
- Converging meridians

- Equidistant parallels
- The shortest distance between two points is represented by a straight line
- Surface areas are represented by a constant ratio
- Surface contours have an accurate representation
- The whole projection is based on a uniform scale

The perfect projection does not exist, so you have to choose the best suited to your needs, depending on the main properties of your application and the geographic areas used in your application.

Map data formats

There are many types of map data formats that can be read into a map. Many of these are governed by cartographic Standards.

JViews Maps handles a large number of these formats. See *Data formats*.

JViews Maps also allows you to import (read from) and export (write to...) specific raster, vector, and database formats.

The added value of JViews Maps

Presents the key benefits of JViews Maps.

In this section

Overview
Gives an overview of the benefits of JViews Maps.

Display of complex maps
Explains why JViews Maps can display complex maps.

Performance
Explains why JViews Maps brings performance.

Component libraries
Describes the advantages of component libraries.

Map Builder
Describes the advantages of the Map Builder.

Map preparation
Describes the map data formats that are handled by JViews Maps.

Advanced map animation
Presents more advanced features for map animation.

Overview

IBM® ILOG® JViews Maps is the solution for delivering asset-management map displays. Its completely open API is designed to hide the complexity of building such displays by handling all the lower-level mapping engine functionality. You can concentrate on the data you want to display, rather than map formats, projections, spatial reference systems, huge data sets, and so on.

JViews Maps provides a unique combination of advanced technology support and in particular:

- High performance background maps for zooming, panning, rotating, and so on ...
- Advanced overlays as symbols that can, for example, change state according to specific sensors.

Using the JViews Maps advanced technology you can mix rich background maps with interactive foreground objects. A mixture of the two is required by most if not all supervision applications needed to monitor assets within their earthly context (town map, country map, or building floor plans). Refer to Before you start to understand the purpose and relationship of background maps and the overlay parts of JViews Maps.

In general, Geographical Information Systems (GIS) focus on map creation, such as placing news roads or buildings on a map, and map distribution across servers and networks. JViews Maps effectively complements GIS by reusing GIS data and by allowing for fast and efficient applications that permit quick background map development.

Display of complex maps

JViews Maps can display complex maps with numerous overlaid entities due to:

- Advanced rendering techniques
- Smart data loading capabilities
- Multithreaded components

There is a full spectrum of map display capabilities, including the most popular data format readers and writers, geodesic computations such as projections, ellipsoids, and datum, and hundreds of predefined coordinate systems. JViews Maps lets you load maps quickly and create and animate multilayered and multiscale maps with user-defined visual representations. This is achieved by styling properties such as colors or transparency.

Multithreaded components mean that lengthy operations happen in the background so that the application is never frozen waiting for a computation to end. Users can continue working on others tasks and see the results when they are ready. It also greatly helps to improve the perceived performance.

Performance

One of the most striking features is the accelerated performance that leverages load-on-demand and pixel-on-demand for handling very large data sets while minimizing memory footprint. Map labels contribute to easily readable displays. JViews Maps has efficient redraw options and the standard set of interactions such as panning, zooming, selecting, and annotating.

Component libraries

JViews Maps contains component libraries and not ready made tools with a defined feature set. This means that users can easily:

- Modify or extend any existing features and behavior to fit their needs.
- Design the application to fit their architecture and their specific needs.

This flexibility is important for freedom of choice in new application development and also allows the enrichment and extension of legacy applications.

Map Builder

The Map Builder is built from the libraries and is delivered in the source code. Although the Map Builder is a tool for defining the background maps of an application, it also serves three other functions important for getting started with the product quickly:

- **Map preparation tool**: Using the Map Builder you can prepare a map that mixes different data sources (satellite, intelligence, transportation networks, …) and specify what you want to see, when you want to see it, and how you want to see it. This means that you always get a meaningful display, for example, you do not need to see streets at worldwide level.
- **SDK exploration**: The Map Builder offers an easy way to accelerate your evaluation of the wide variety of features offered by the SDK.
- **Jump starting development**: The Map Builder is delivered as source code and built using IBM® ILOG® JViews Maps SDKs. If you want, you can reuse code fragments in your final application.

With the Map Builder, you have access to multiple coordinates and measuring unit systems; contextual grids in Longitude and Latitude or the Military Grid Reference System (MGRS) with minimized memory and CPU footprint.

Map preparation

JViews Maps offers the fusion of vector, raster, and terrain elevation data. The data is reprocessed on-the-fly allowing you to mix and match map data whatever its storage parameters are.

You can display contextual information by separating different contexts onto different layers, since different information is visible at different zoom levels if you use the zoom-sensitive look and feel. JViews Maps provides multilayered and scaled maps. You can define your own theme for a map by configuring individual layers differently.

Data formats

JViews Maps allows you to import many different map formats.

Vector formats:

- ESRI Shape (see Shapefile format)
- TIGER/Line
- DXF (see DXF format) files (`.dxf`)
- KML and KML Zipped (KMZ) formats (`.kml`, `.kmz`) (see KML/ KMZ)
- MapInfo MIF/MID (see MIF file)
- SVG files (.svg)

Raster formats:

- GeoTIFF (see GeoTIFF format)
- GIF
- JPG
- PNG
- TIF/TIFF files
- Images from a Web Map Server (see WMS standard)

Databases:

- Oracle® (see Oracle Spatial)

Terrain elevation:

- NIMA DTED® 0, 1, and 2 (see DTED format)
- GTOPO30 DEM

The products offer facilities for exporting various vector, raster, and database formats:

- Vector: ESRI Shape

- Raster: GeoTIFF
- Database: Oracle®

GIS transformations

Dynamic geodesic management is provided through:

- 1944 predefined coordinate systems (projection, ellipsoid, and datum parameters)
- 27 predefined projections plus user-defined
- 58 ellipsoid forms plus user-defined
- Molodenski DATUM
- Well Known Text (WKT) parsing

Contextual grids are supplied as:

- Longitude and latitude
- Military Grid Reference System (MGRS) (rather like the Universal Transverse Mercator (UTM) system)

JViews Maps offers multiple coordinate systems and the management of multiple units of measurement.

Advanced map animation

The use of a background map and symbols that can be moved in real time across the map, and which can be panned, zoomed and so on, provide advanced map animation features.

There are graphics editing facilities provided by the *Symbol Editor for JViews Diagrammer* to help you create your own custom symbols, see Using the Symbol Editor.

Optimized performance

Modern applications often require large map data sets as satellite and *Unmanned Aerial Vehicles* (UAV) views or exhaustive transportation networks to realistically depict situations.

JViews Maps is optimized to efficiently handle such large data sets to provide video-like performance while minimizing memory requirements.

For example, by leveraging load-on-demand and dynamic data subsampling, the memory footprint is minimized, even though the data contains millions of objects. Display performance is based on disk and RAM caching as well as advanced quadtree rendering and automated clipping and tiling. This performance not only provides a better experience for the user, but also saves processing time for other application CPU intensive tasks.

You will be impressed by the reactivity of the interactive zooming and panning, the storage and navigation facilities for Areas of Interest, and the map rotation capability for GPS-like application development.

Symbology

JViews Mapsleverages the rule-based presentation layer of JViews Diagrammer to update the appearance and position of assets when underlying application data, such as status or positions, evolve. You can create your own symbology and manipulate vector, raster, and complex objects. Symbols support hierarchical logical grouping. See the JViews Diagrammer document Using the Symbol Editor for details of how to create your own symbols.

Typical uses of JViews Maps

Gives examples of use of JViews Maps.

In this section

Overview
Describes the application fields of JViews Maps.

General use
Gives general tips for using JViews Maps.

Typical actions
Describes typical actions when creating maps.

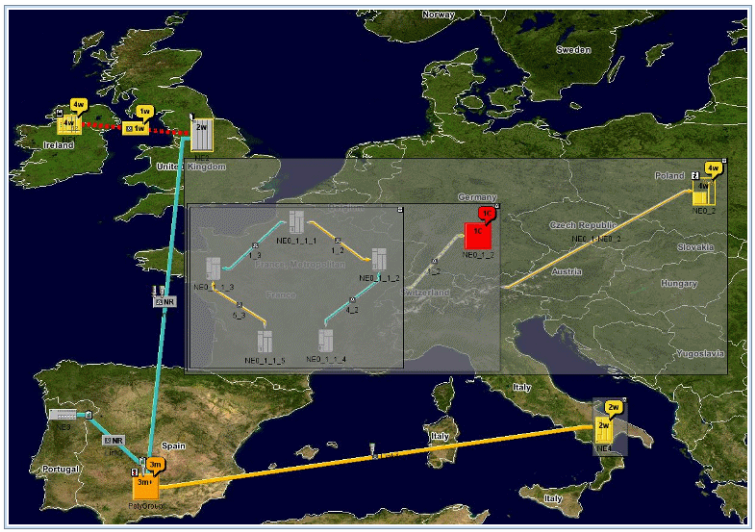
Overview

You can use JViews Maps for:

- Network management systems

  The network elements are georeferenced and displayed on digital maps. JViews Maps manages the type of map to be displayed and displays meaningful data at each zoom level. Overviews display high level networks (summaries) with continent or country maps underneath. Network operators can drill down into more detailed views: they zoom in on a specific region or even on a city, where they can see the network details. In these cases, JViews Maps loads more precise maps that are more convenient for city-level views.

  The following figure shows an example of part of a network.

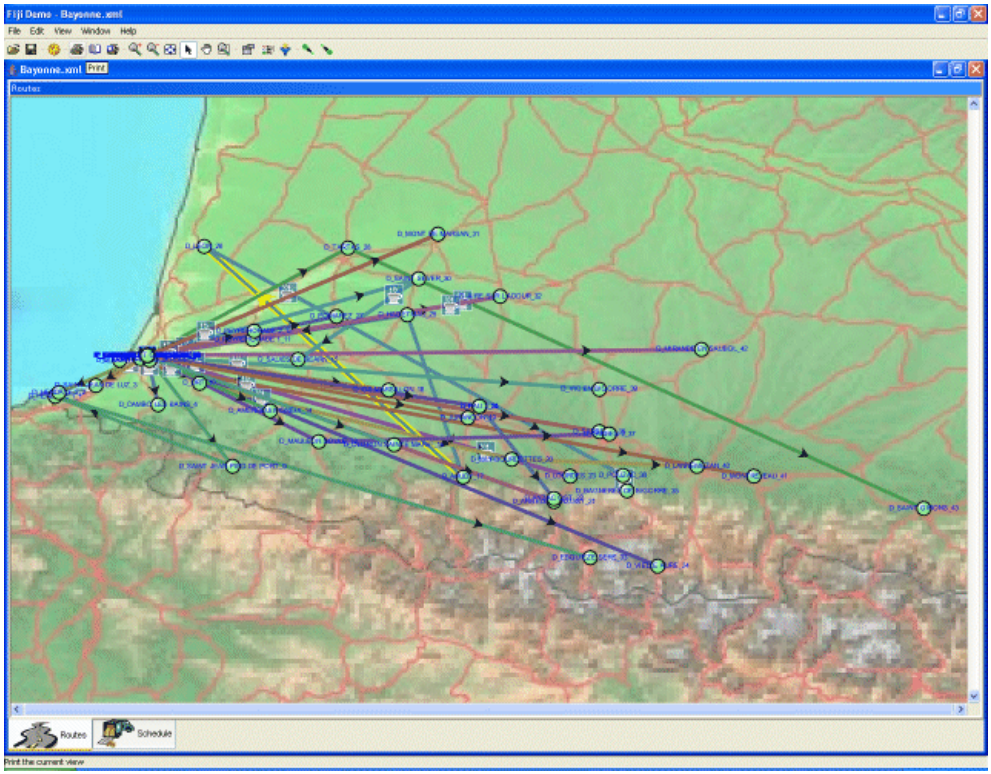


Network management system example

- Supply Chain Management, Transportation

  Company assets such as trucks, warehouses, and customer sites are displayed on maps to make the supply chain more readily appreciated. Warehouses can be displayed as symbols upon maps and routes can be displayed as links or as precise road-routes.

  The following figure shows an example of part of a network.



Supply Chain Management transportation example

General use

JViews Maps can manage accurate map data and projections for asset mapping applications. Such applications need to combine map data coming from various sources to display the operations.

JViews Maps has the flexibility and high performance necessary for fast response time. Typically, you would want to read in the map data, create a map theme and then place and animate symbols on the background map. You can use your own predefined theme and apply it to different sets of data.

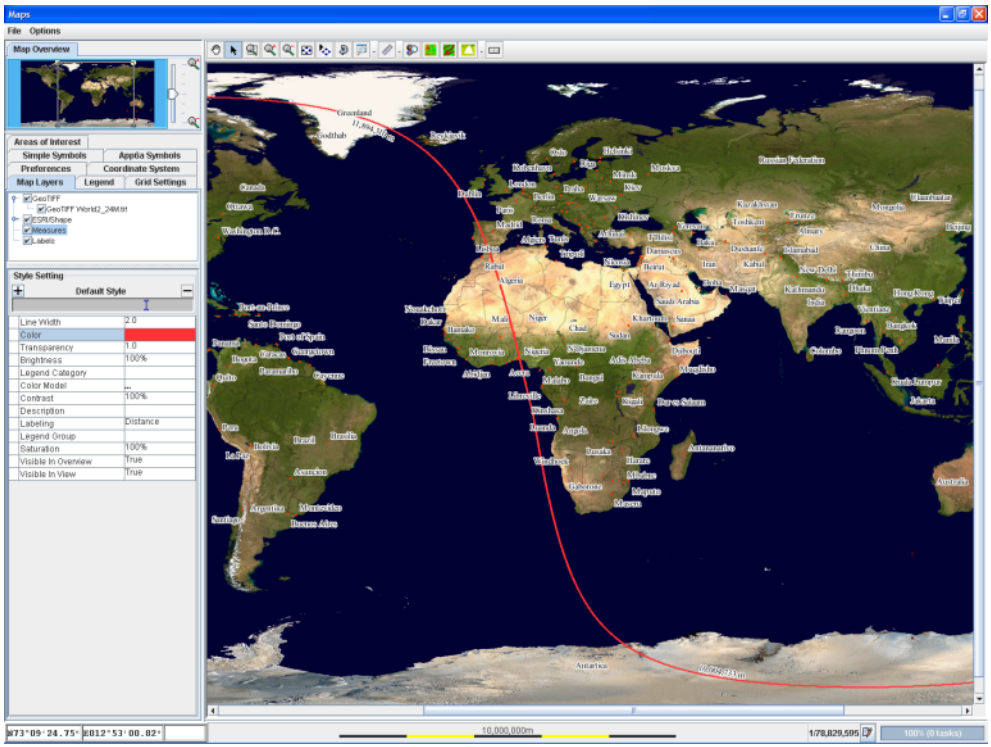
In the Map Builder, you can read the data and apply the theme in an easy-to-use point-and-click editor. You can also save your map application, making the configuration settings persistent. Data and theme can be saved separately.

Typical actions

Reading the Data

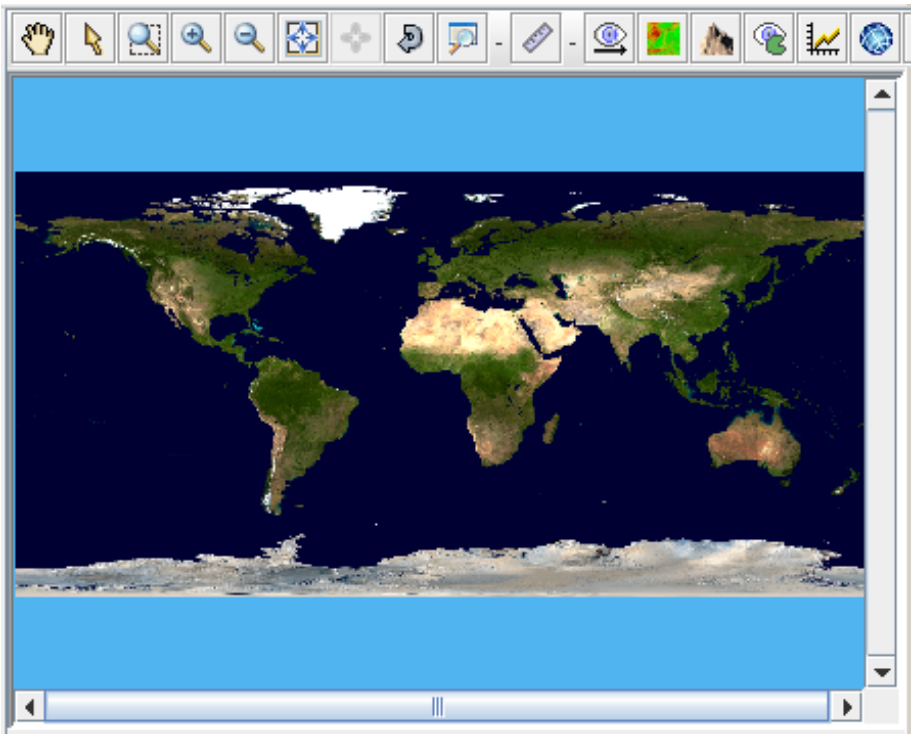
You import different data sources into a map to provide different views of the data contained in the map. Each data source is associated with a different map layer.

For example, you might start with a shape file, to provide an uncomplicated view of the world as a backdrop.



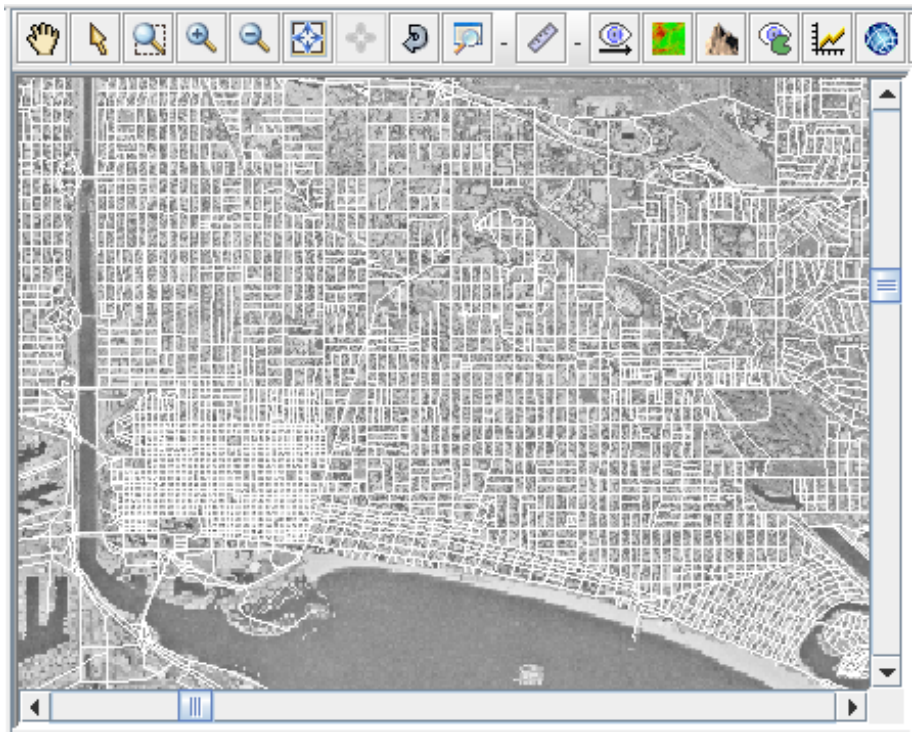
World map in an application

Then you could refine the esthetic representation of the map by importing a GeoTIFF format map of the world.



World map in GeoTIFF format

Street level views can be added from GeoTIFF format and ESRI/Shapefile format data sources.



Street data in Los Angeles from an ESRI/Shape data source

An example of the series of typical data source types outlined here can be found in An example map in *Using the Map Builder* .

That particular example is based on North America and Los Angeles in particular, but you could equally well import your own data sources for other parts of the world.

JViews Maps is supplied with the Map Data disk, which includes general map formats that can be distributed free.

Themes and styling

For each map layer you can decide its look and feel in terms of its color, thickness transparency, and so on, and you can define settings for specific zoom levels so that the look and feel changes when users zoom in and out. The look and feel of each layer and its zoom triggers represent a map theme. This is used to visualize only meaningful information for each context.

Placing symbols

Symbols can be created using the Symbol Editor and added to maps using the Designer for JViews Diagrammer.

You can then load the Designer project file that contains the background map developed with the Map Builder and the Symbols added with the Designer, and load it into your application. For more information about symbols, see the JViews Diagrammer document Using the Symbol Editor.

The Tools for the job

IBM® ILOG® provides a sample application, the Map Builder , that includes the data reading and writing facilities and the layer styling facilities for building theme in a point-and-click editor. The sample code is accessible and you can easily customize it.

The product also includes the Designer for JViews Diagrammer. The Designer is a point-and-click editor used for writing style rules that control the styling of nodes on the map.

You can switch easily between these tools to hone your application. Typically, you use the Map Builder to speed up application development time by reading in the data sources in different formats and then creating the map theme by styling the layers. Then you would switch to the Designer to write style rules to change the appearance of nodes depending on certain conditions and to design symbols to represent nodes on the map.

If you need more than these easy-to-use GUIs have to offer, there is a full-featured SDK for refining and customizing your application. In addition to the general mapping API of JViews Maps, you have access to the Styling and Data Mapping (SDM) package of the JViews Diagrammer API for the styling of nodes to place on a background map, and the whole of the JViews Framework API with its powerful graphics framework. You have full control over what you do.

Key Strengths of JViews Maps

In the application, the up-to-date data can be displayed against the background map with the appropriate symbols correctly positioned.

The key strengths of JViews Maps make it ideal for asset-mapping applications.

- Components, such as beans, readers, map displays, and map views, are easily customizable to adapt to your needs. Your existing system architecture can be maintained.
- Performance allows for almost instantaneous zooming in and out and the system can manage huge amounts of data.
- Animated symbols can show the movement of resources on the map.

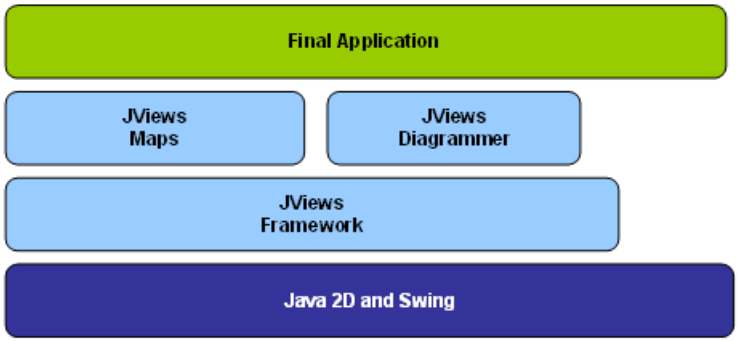
Since the map data and the map theme can be saved separately, the map theme can be used to style maps for different regions so that they have the same color scheme and visual properties. The final user can then reuse this theme and apply it to the appropriate data source at run time.

Map applications built with JViews Maps can be an essential part of a decision support center whether on or offline.

Developing with the SDKs

You may have specific requirements in your application that cannot be satisfied by the pre-defined behavior of the Map Builder or the Designer alone. For example, if your application requires specialized interactions or if you need to implement the data model interface to connect to your application data.

The SDK offers you a comprehensive API to use or extend the Java™ classes involved in the creation of your map-based application. You can basically access all the entities involved in your application, enrich them with new behavior, or modify predefined behavior. See the Programmer's documentation for more detailed information.



SDK layers in JViews Maps

All JViews libraries are built on top of the Java 2D™ and Swing libraries, with no platform-specific code. This means that applications developed with JViews Maps run on any platform that supports Java™ .

On top of this low-level layer is the JViews Framework layer, which includes among others efficient data structures, prebuilt user interaction services, and a printing facility.

Above JViews Framework is IBM® ILOG® JViews Maps, which provides a wide range of map manipulation and display services. It is built on the data structures and I/O facilities of JViews Framework.

At the same level, there is JViews Diagrammer, which provides a wide variety of displays consisting of custom graphic objects that are data-aware. This means that the graphic objects in the display can change their appearance as the underlying data model changes. For example, a graphic object that represents a vehicle can have its color change if its status field changes.

The JViews Diagrammer SDK is needed when you display interactive objects on top of your maps (the symbols), whereas the IBM® ILOG® JViews Maps SDK handles the background map manipulations.

IBM® ILOG® JViews Diagrammer uses a Model-View-Controller (MVC) architecture that will be very familiar to Java™ programmers used to the Swing structure. Its purpose is to separate the data model from the views and connect the two with a rule-based style manager that controls the look of the objects based on the data values.

JViews Diagrammer calls this mechanism Styling and Data Mapping (SDM). It is used by JViews Maps to store and manipulate any interactive object that appears on top of a map.

Thus, JViews Maps uses two distinct data structures:

- One for storing the map data that is relatively static (such as definitions for roads and boundaries).
- SDM for storing information about the custom objects of an application.

Basic concepts

Introduces you to the important concepts and features of JViews Maps

In this section

Usage and design concepts
This section gives a brief explanation of what IBM® ILOG® JViews Maps is designed to do from the perspective of a user, the design concepts that support the approach, and how to get started with your project.

Map preparation
Describes the steps involved in the preparation of a map.

Map animation
Describes the ways in which you can animate a map.

Usage and design concepts

This section gives a brief explanation of what IBM® ILOG® JViews Maps is designed to do from the perspective of a user, the design concepts that support the approach, and how to get started with your project.

In this section

Usage concepts
Describes the usage of JViews Maps.

Design concepts
Lists the advantages of JViews Maps in terms of design.

Development approach
Presents the two possible ways to approach the development of maps.

Usage concepts

IBM® ILOG® JViews Maps is designed to help you to create map displays or build any application that uses maps.

It was designed to be used as follows:

1. Import the map data and design the look and feel. IBM® ILOG® JViews Maps can read map data from multiple map file formats and storage locations and merge it into a single view. You can then transform the map into a new projection, add styling, and create a customized theme to reuse with other maps.
2. Animate your map. You can zoom, pan and load new map data with minimum delay. You can also add and move symbols that have been added to your map using Designer for JViews Diagrammer. IBM® ILOG® JViews Maps can move large datasets in seconds and, because of its multithreaded design, it keeps running while performing multiple memory and CPU-intensive tasks.

Design concepts

The design of JViews Maps delivers:

- Productivity, through high-level components and numerous off-the-shelf features that enable you to build advanced prototypes in just days or weeks.
- Flexibility, from the use of advanced design patterns and a Model-View-Controller (MVC) architecture. JViews Maps is a pure object-oriented toolset built for Java™ developers.
- Ease of integration: JViews Maps is based on standards such as Java2 and XML, and on cartographic Standard map formats. It is a development toolkit built with an open architecture that facilitates integration with third-party solutions and allows you to create new map-enabled applications easily or enrich your legacy systems.
- Optimized performance and a minimized memory footprint: JViews Maps can display large data sets in seconds without interrupting the system.

Development approach

The way you use JViews Maps depends on your requirements. If the map area is decided at run time, you can use the API to read and style map data.

- For importing map data during application execution, use the SDK components and user-interface elements:
  - Beans defining numerous GUI features.
  - Map Builder, a fully functional map preparation application, supplied with source code for you to customize and use in your own applications.
  - Numerous demos, once again supplied as source code, and focusing on discrete features.
  - An open API allowing full customization.
- When a map area is known in advance, use the Map Builder to prepare it beforehand. JViews Maps provides a prepared map as output containing either the map description only or all of the map data packed in a single file. The prepared map can be called in the final application with just a single line of code.

Map preparation

Describes the steps involved in the preparation of a map.

In this section

Overview
Summarizes the procedure for preparing a map.

Importing map data
Explains how to prepare a map by importing data.

Merging
Explains the possibilities of merging different map data formats.

Contextual information display
Explains how to customize the display of map information.

Coordinate system
Describes the coordinate systems and their possible transformations.

Grids, units, and measures
Describes the grid systems, units, and measures supported by JViews Maps.

Export
Lists the export options of a map.

Overview

To prepare a map you import and merge maps and map data into a single view. You can decide what to make visible, when and how, style the data into a theme, transform the map according to the required projection, and apply unit and measure preferences. You can also export the map, if required.

Importing map data

You create new maps by exploiting the data contained in your existing electronic maps. Also, you can update your new map by importing additional map data at anytime during map preparation or at run time and without interrupting the system.

Map data model properties

JViews Maps integrates the various map data sources using a set of properties that define the way your map will display and react to run-time changes.

Properties

By default, JViews Maps embeds the following properties into a working map:

- The coordinate system, including ellipsoid and datum, to use to display the map.
- Ground and altitude measuring units such as meters or feet, but also the preferred way to display coordinates, such as Degrees Minutes Seconds (DMS).
- A data source list, specifying where map data comes from.
- A layer tree, arranging the diverse data sources and settings into layers.
- A styling theme, defining the colors, thickness, visibility, and so on, of each layer, depending on the scale factor.
- A list of areas of interest.

Extending and changing properties

The map properties can be extended or changed at any time during map preparation or final application execution. For example, you can implement applications to allow for a change of map projection or data source at run time.

About readers

Readers are provided that read map data in its native format, that is, without any need for preprocessing.

Formats supported

Readers can read a wide range of file formats containing the map and map data such as geometry and topology. *Supported File Formats* shows the supported file formats:

Supported File Formats

Types	Georeferenced	Non-Georeferenced
Vector	ESRI Shapefile formatfiles (`.shp`) TIGER/Line (`.RTx`) MapInfo (`.mif`, `.mid`)	DXF format files (`.dxf`)
Raster	GeoTIFF format(`.tif`) Images from a Web Map Server.	GIF, JPG, PNG, TIF
Vector & Raster	Regular IBM® ILOG® JViews Maps files (`.ivl`) KML/ KMZ files (`.kml`, `.kmz`) SVG files	
Terrain Elevation	NIMA DTED format 0/1/2 GTOPO30 DEM	
Databases	Oracle SpatialOracle® Spatial	

Additionally, you can use the open APIs to import other map formats.

Merging

JViews Maps can merge vector, raster, and elevation data from multiple map file formats and storage locations. You can prepare a map using georeferenced data (that is, systems defining projection, ellipsoid, and/or datum information), and non-georeferenced map assets. For example, you can create a map with a UTM projection, import data from a CADRG map (with an Arc projection), a shape file based on a Lambert projection, and overlay these with a JPEG image. JViews Maps performs a reverse projection on the data sources into a common system such as Longitude/Latitude. The support offered is as follows:

- All georeferencing systems are supported. (The list of more than 1900 reference systems in WKT format is easily accessible).
- For non-georeferenced map assets such as GIF and JPEG files, JViews Maps provides components and user interfaces that use calibration and georeferencing mechanisms.
- Images using unknown projections can be calibrated using a polynomial interpolation.

Contextual information display

Imported maps contain a vast amount of data that needs to be organized and presented in a way that is easily understood and suitable for its intended purpose. For example, you may want to create a map in which you can zoom smoothly from a small scale view to large scale view while displaying only information relevant to each particular zoom level. To do this, data sets need to be associated, displayed, changed, hidden, and so on. Additionally, you will want to use styling properties, such as color and transparency, to make the information easy to understand. JViews Maps uses layers and styling to achieve these goals.

Layers

JViews Maps defines a recursive hierarchy for map data. For example, you can define:

- One folder or layer for all roads with subfolders or distinct layers for motorways, major roads, streets.
- One folder or layer for public transport with subfolders or distinct layers for airports, railway stations, bus stations, and so on.

JViews Maps allows you to zoom from worldwide views down to a street level view using different data sets, but presenting a smooth transition to the end user.

Importing data into layers

Data is imported to populate layers as follows:

- Automatically:
  - When you import a map or map data, it is automatically assigned to a layer.
  - When you load multilayered maps such as a TIGER/Line vector map, one layer is created for each original feature set. This allows the data to be more easily understood and styled when it is loaded.
- Manually through the GUI, such as when you create an orthodromy measure.
- Programatically or manually. You can define the map-layer structure to fit your needs and map the different data sources into them. This approach is part of the map theme definition in which you state:
  - *What* map data is to be displayed.
  - *When* the data should be displayed, for example, you can display motorways on country scale views and streets at the city levels.
  - *How* the data is displayed, for example, the color, transparency, and line thickness.

Styling

Styling enables you to change colors, transparency, and so on to enable the data to be better understood. You can define styles for all objects in a particular layer or layer tree.

Coordinate system

JViews Maps can transform coordinates delivered by one geodetic system to those based on another. For example, you can transform coordinates from a satellite-based system to those based on the national geodetic system of a particular country. Additionally you can set the ellipsoid and the datum you require.

Projections

JViews Maps supports the following projections:

Azimuthal Equidistant	Cassini
Albers Equal Area	Lambert Equal Area
Lambert Conformal Conic	French Lambert
Cylindrical equal Area	Eckert 4
Eckert 6	Equidistant Cylindrical
Geographic	Gnomonic
Lambert Azimuthal Equal Area	Mercator
Miller Cylindrical	Mollweide
Oblique Mercator	Orthographic
Polyconic	Robinson
Sinusoidal	Stereographic
Polar Stereographic	Universal Polar Stereographic
Transverse Mercator	Universal Transverse Mercator
Wagner 4	User defined

The projection system is flexible and extensible. For example, you can develop your own specific projection algorithms and integrate them with the product.

Ellipsoid

Each projection system supports configurable ellipsoids:

- CPM (The ellipsoid specified by the French Weights and Measures Commission ("Commission des Poids et Mesures" in 1799)

- SGS85 (The ellipsoid used in the Soviet Geodetic System 85)
- SPHERE (A spherical ellipsoid representing the earth)
- WGS1960, WGS1966, WGS1972, WGS1984
- One of the 60 other predefined ellipsoids
- User defined

You can define your own ellipsoids, entering the major axis and inverse flattening values.

Datum

JViews Maps comes with more than 200 predefined datum.

Transformation

For accuracy up to a meter, IBM® ILOG® JViews Maps provides the following DATUM transformation method:

- Molodensky (three parameters)

Custom Transformation

You can develop your own DATUM algorithm. APIs allow access to the data model and return computational results for display.

Grids, units, and measures

JViews Maps allows you to overlay grid systems and set your preferred units and measures.

Grid systems

Predefined grids are provided for:

- Longitude and latitude
- The UTM/MGRS reference system

Adaptable grids

The grids are called adaptable grids because they adapt to the zoom level to ensure that there is always something meaningful on the screen.

There is no impact on memory and little on performance. JViews Maps uses optimized parameters including, for example, delayed drawings. In this case, a simplified grid is displayed during zooming and panning, with only the 20 lines needed drawn on the view. When the view stops moving, it displays the more refined grid and labels.

Units

JViews Maps supports different units, including:

Kilometer	Meter
Decimeter	Centimeter
Millimeter	International Nautical Mile
International Inch	International Foot
International Yard	International Statute Mile
International Fathom	International Chain
International Link	U.S. Surveyor's Inch
U.S. Surveyor's Foot	U.S. Surveyor's Yard
U.S. Surveyor's Chain	U.S. Surveyor's Statute Mile
Indian Yard	Indian Foot
Indian Chain	

Extending Units

You can extend the supported units for linear and angular measurements.

Different measuring units can be used for distances and altitudes.

Working units can de changed dynamically, with conversion computed by a set of provided components.

Measuring

JViews Maps provides off-the-shelf components and end-user interactors for orthodromy measurements of distance and bearing.

Distances can be measured in the supported units and changed dynamically.

Export

Once prepared, a map can be exported as:

- A map theme with pointers to the original data sets. This avoids data duplication, since all map data is read from its original data stores.
- A map theme and map data, which improves loading time and makes sharing maps easier. All the map data is stored in a single proprietary map format to provide very fast loading times. This file format is documented and can be easily extended to include your application data.
- ESRI (see Shapefile format) files (`.shp`) for vectorial information.
- GeoTIFF format files for raster map information for integration with external map software.
- Oracle Spatial database.
- KML/ KMZ (Google Earth™ format) files to visualize or share the map on top of the 3D maps provided by this tool.

Open APIs are provided for exporting specific map formats.

Map animation

Describes the ways in which you can animate a map.

In this section

Overview
Provides a brief overview of how you can animate a map.

Performance
Lists the techniques used to optimize the performance of maps.

Map manipulation components
Lists the components that are available for advanced map manipulation.

Symbology
Explains how to use symbols on maps.

Overview

Map animation allows you to perform such tasks as rotating a map, zooming in for more detail, and loading more data without compromising system performance. You can also add symbols to your map, but this must be done outside the Map Builder using Designer for JViews Diagrammer.

Performance

To provide the required performance for importing and rendering map data JViews Maps leverages the following techniques:

- Optimized hierarchical quadtree rendering engine
- Double and Triple buffering
- Load on demand and hard-disk and RAM caching
- Raster and vector map tiling
- Raster map subsampling
- Multithreaded environment

The results:

- Display time when opening the prepared map: 2 seconds
- Required RAM for display: 16MB, or only 6% of the map data size
- Zooming in and out: 64 frames per second (FPS) with only 50% CPU usage
- Panning: maximum 220 FPS with only 30% CPU
- Rotating Map: 40 FPS
- Map reprojection: 5 seconds
- Moving overlaid symbols:
- 30 FPS for 1,000 objects
- 5 FPS for 15,000 objects

Map manipulation components

JViews Maps provides numerous advanced components for map manipulation including:

- Interactive zooming with progressive information display
- Optimized panning
- Storage and navigation of Areas of Interest
- Map rotation
- Synchronized overviews
- Current map scale
- Zoom to area or layer, or fit to zoom
- Layer, map data, and symbol trees

Symbology

There are facilities for advanced symbology management (creation and animation)

Libraries

You can create specific symbology libraries by:

- Creating iconic symbols, like airports or military camps, based on either raster, vector, or composite graphical elements.
- Implementing a basic editor GUI and managing symbol rendering.

Using the specific SDM Renderer, symbols can be natively georeferenced. You can:

- Drag and drop symbols onto a map.
- Take full control using the APIs for creation, deletion, and fast animation.

Every symbol is moveable, selectable, and editable. Additionally, symbol groups can be collapsed, expanded, grouped, and ungrouped. There are symbol editing facilities to help you create your own custom symbols, see the JViews Diagrammer document Using the Symbol Editor.

Architecture

Presents the different components of the product and what they are used for.

In this section

Overview
Lists the architectural components of JViews Maps.

Populating the map
This section describes how business data is imported and handled in map displays. A map application makes use of the data connectivity and styling features of JViews Diagrammer for the foreground symbols, but not the background map data.

Styling
Explains the principle of styling in JViews Maps.

Overview

JViews Maps includes the following parts of other IBM® ILOG® visualization products:

- Styling and Data Mapping of JViews Diagrammer
- JViews Framework

This section describes the relationship between the features managed by these products for creating dynamic map applications.

Populating the map

This section describes how business data is imported and handled in map displays. A map application makes use of the data connectivity and styling features of JViews Diagrammer for the foreground symbols, but not the background map data.

In this section

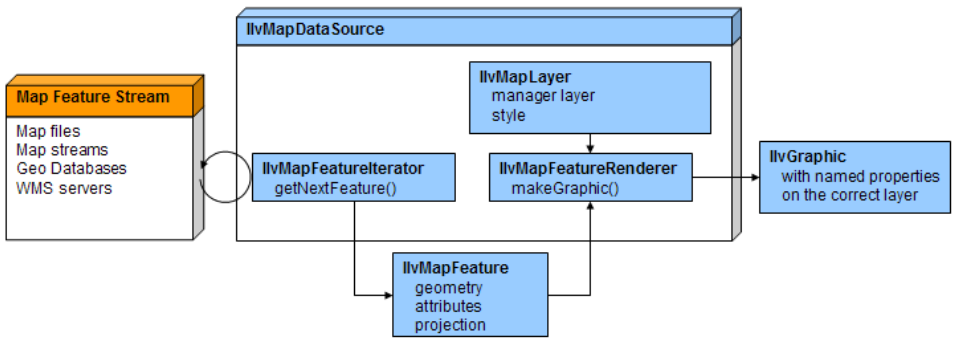
Map data sources
Describes the specificities of map data sources.

Diagram data sources
Describes the specificities of diagram data sources.

The SDM engine
Explains what the SDM engine is about.

Map data sources

A map data source is the preferred way to connect your application to georeferenced data sets. A map data source connects a feature iterator, a renderer, and a map layer, see *Connecting a Feature Iterator with a Renderer and a Map Layer*.



Connecting a Feature Iterator with a Renderer and a Map Layer

The feature iterator is an object that iterates through the features of a map to obtain the data for the map overall. The features include the coordinates, the bounding box, and other features to be parsed.

The renderer is used to create the graphic objects that will represent the data model objects in the map.

Each data source is associated with a particular map layer, which manages the order and the style of the objects.

Specific map data sources are dedicated to accepting specific types of map format. See *Data formats* for a list of formats.

The Map Builder provides menu commands to import and export specific types of map data through an easy-to-use GUI. The SDK provides specialized interfaces for reading and writing map data of the supported types and for setting the appropriate data source.

The management of the graphic objects and their properties is part of the JViews Framework and is explained in detail in Managers and Graphic objects in *The Essential JViews Framework*.

Diagram data sources

The styling and data mapping facilities of JViews Diagrammer allow you to connect to data sources and to style their representations.

The role of the diagram data source is to load the data to display in the diagram, and possibly write back the data if it has been modified.

There are the following predefined types of data source: flat files (in the Designer only), XML, and JDBC.

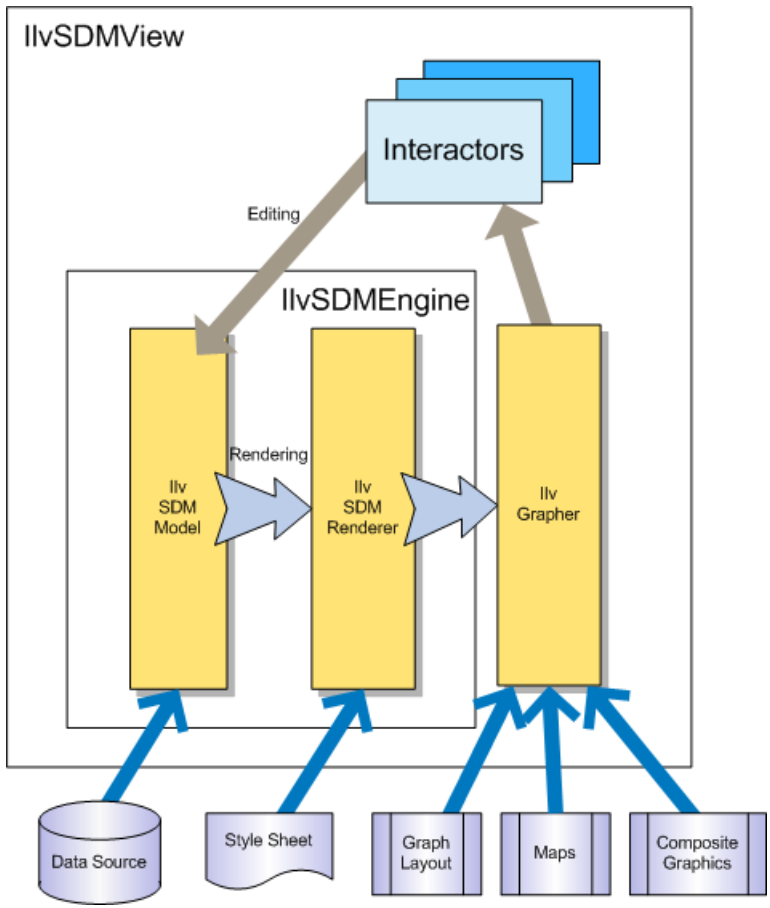
When you cannot use flat files, XML, or JDBC, you can always connect the data model to your data by implementing the data model interface in Java™ .

The SDM engine

The SDM engine is one of the most important pieces of JViews Diagrammer as it controls the data-to-graphics mapping. There are four key elements in the data-to-graphics mapping process:

- A data model that interfaces to the data to display or edit. This data model is completely independent of the GUI, and refers only to the business objects of your application.
- Renderers that style the diagram as a whole and the graphic objects in it. Renderers apply the styles specified in the style sheets.
- A grapher in which the graphic objects representing the data model are created as nodes and links. It provides the infrastructure that is minimally necessary to draw a diagram.
- Interactors that permit user actions on graphic objects. Common requirements are for zoom, pan, select, and object creation functions.

An example of the SDM Engine and the Data-to-Graphics Mapping is shown in *The SDM Engine and the Data-to-Graphics Mapping*.



The SDM Engine and the Data-to-Graphics Mapping

As shown in *The SDM Engine and the Data-to-Graphics Mapping*, the mapping between the data model and the graphical representation is bidirectional:

- Data model to graphics: the rendering process is controlled by the style sheet, which lets you tell the SDM engine how you want each particular kind of data object to be displayed in the grapher. The rendering process is performed by specialized renderers.
  - When the data model is loaded, the SDM engine explores it and creates graphic objects representing the nodes and links defined by the data model in the grapher.
  - When the state of an object in the data model changes, the SDM engine updates the graphic object representing the modified data object.
- Graphics to data model: the editing process relies on built-in editing facilities that act directly on the underlying data model. The actions in an editing application are implemented by interactors. For example:

- When the user moves a graphic object (for example, in an editor), the SDM engine updates the geometric properties of the object in the data model.
- When the user expands or collapses a node (for example, in a navigation application), the SDM engine updates the expand/collapse status of the object in the data model.

The Data Model Interface

The SDM data model is the interface that tells the SDM engine how to get the data to be displayed. The SDM data model is an abstract description of a set of nodes and links between nodes. Nodes and links have a user-defined type (also called the tag), and a set of named properties.

JViews Diagrammer provides prebuilt data models, and you can implement the data model interface to connect your data.

Styling

Explains the principle of styling in JViews Maps.

In this section

Overview
Explains in a few words how styling is achieved in JViews Maps.

The map theme
Describes what a map theme consists of.

JViews Diagrammer style sheets
Explains the principle of style sheets.

Symbols
Explains the use of symbols in maps.

Graphics SDK for experts
Explains how to make an advanced use of graphics through the JViews Framework SDK.

Overview

Styling is a key feature in IBM® ILOG® mapping products. In JViews Maps, styling of maps is achieved through properties associated with layers. In JViews Diagrammer, styling is applied through CSS style rules written to a style sheet.

The map theme

The map theme is the sum of all the style sets defined in the map layers.

In addition to data sources, most layers also contain *styling* information, for example, about grids, labels, terrain analysis, measurements. See *Themes and styling* for more details about styling in maps.

Most map layers are used to manage graphic objects created from map features imported from various map data files. The styling options differ according to the content of each file and the map features imported by the different readers.

JViews Diagrammer style sheets

Styling in JViews Diagrammer involves the following constructs:

- Style rules
- Composite graphics
- Backgrounds

JViews Diagrammer style rules

The StyleSheet renderer applies a style sheet to a data model. The style sheet contains style rules in CSS2 format that describe how the objects in the data model are displayed in the diagram.

A style rule consists of two parts:

- The condition part called the selector applies to the data model, and is used for pattern-matching.
- The action part called the declarations applies to the corresponding graphic objects, and is used for rendering.

When designing a notation, you create many style rules, each of them matching a particular case in the data model. You define rules that apply to objects represented as nodes, and rules that apply to objects represented as links.

The style rules are usually defined from the most generic to the most specific. The generic rules usually create the base symbol for each type of object. The more specific rules add new shapes, or change graphics properties for the symbol defined in the generic rule.

The style rules are also used to specify the options of a diagram. Such rules have no selector and the declarations customize the way the options operate.

The Designer for JViews Diagrammer is perfectly suited for creating style sheets that define the look-and-feel of diagrams. Within the Designer, styling takes place, but the CSS syntax is largely hidden: selectors are defined in a natural-language editor and declarations are defined by setting graphic properties through panels called Styling Customizers. The style sheet generated by the Designer can be loaded into your application at run time.

Backgrounds and maps

For applications that require a geographic map as a background, you can install a map renderer on your diagram that uses the IBM® ILOG® JViews Maps facilities to read map formats—vector or raster—and to display nodes according to their latitude and longitude.

Symbols

Symbols are moving objects on top of a background map.

You should use the Map Builder to create a map, and then load the map into the *Designer for JViews Diagrammer* and add nodes or links to the model. You can then select which symbols should represent your model. If your symbols are all known at design time, you can also prepare and specify them in the Map Builder. In this case, you should not use the Designer.

The *Symbol Editor for JViews Diagrammer* provides a way to design your own set of rich symbols. Symbols can be nested to reuse simpler symbols within more complex ones. Style rules are bundled within symbols and the symbols themselves are grouped into palettes.

Graphics SDK for experts

The JViews Framework SDK is used to manage graphic objects. You need to be comfortable with writing Java™ code to use this SDK. There is no user-friendly GUI to help you out, although some classes are available as JavaBeans™ for use in your favorite IDE. See Framework classes available as JavaBeans(TM) in *The Essential JViews Framework*.

The following summarizes the main features of handling graphic objects. The mapping products do not have wrapped classes of their own to do this, so if you write your own code you will have to use JViews Framework directly.

For map applications, the supplied source code of the Map Builder shows you how to do this. The easiest way is to adapt the Map Builder code to the needs of your application.

An `IlvManager` object is the data structure that contains graphic objects, such as rectangle, polylines, and so on.

The base class for graphic objects is `IlvGraphic`.

Graphic objects can be stored in the manager. The manager is composed of several storage areas called layers and you can specify in which layer an object is to be stored. Objects in a higher layer are displayed in front of objects in lower layers. A layer is an instance of the class `IlvManagerLayer`.

Note: Do not confuse the storage layer in JViews Framework with the map layer, which is used mainly for styling.

Graphic objects stored in a manager can be displayed in several views. A view is an instance of the class `IlvManagerView`. This class is a subclass of the AWT class `java.awt.Container`.

`IlvGrapher` is a manager that contains nodes and links. Nodes can be any type of graphic object. (Links are instances of the class `IlvLinkImage`.) A graphic object becomes a node of a graph if it is added to the grapher by the `addNode` method. (A link must be added by the `addLink` method.) When a node is removed from a grapher, all the links that come from and go to this node are also removed. When changing the position of a node, the connection points of the links are automatically recomputed.

`IlvGraphic` is the abstract base class of graphic objects managed by an instance of `IlvManager`.

The extensive graphics API of JViews Framework does far more than the basic management of graphic objects. It is a powerful tool for customizing and refining Java™ development. The mapping products give you access to the full JJViews Framework. For example, JViews Framework also includes APIs for deploying thin-client applications.

Developing a dynamic map

Describes the overall process flow for creating applications with background maps that end users can pan and zoom and which display overlaid graphics depicting managed entities.

In this section

Overview
Gives a brief overview of how to develop a dynamic map.

The process flow
Explains the process of building a map.

Creating a map with the Map Builder
Describes the steps involved in creating a map with the Map Builder

Handling symbols
Explains how to create symbols to be used on maps.

Prototyping the application with the designer
Explains how to make use of the Designer for JViews Diagrammer to rapidly create a prototype for an application.

Overview

This section shows the overall process flow for creating a dynamic map. It introduces you to the GUI-based tools that handle specialized aspects of the process and the powerful SDKs that give you the possibility of full and open customization.

The process flow

Explains the process of building a map.

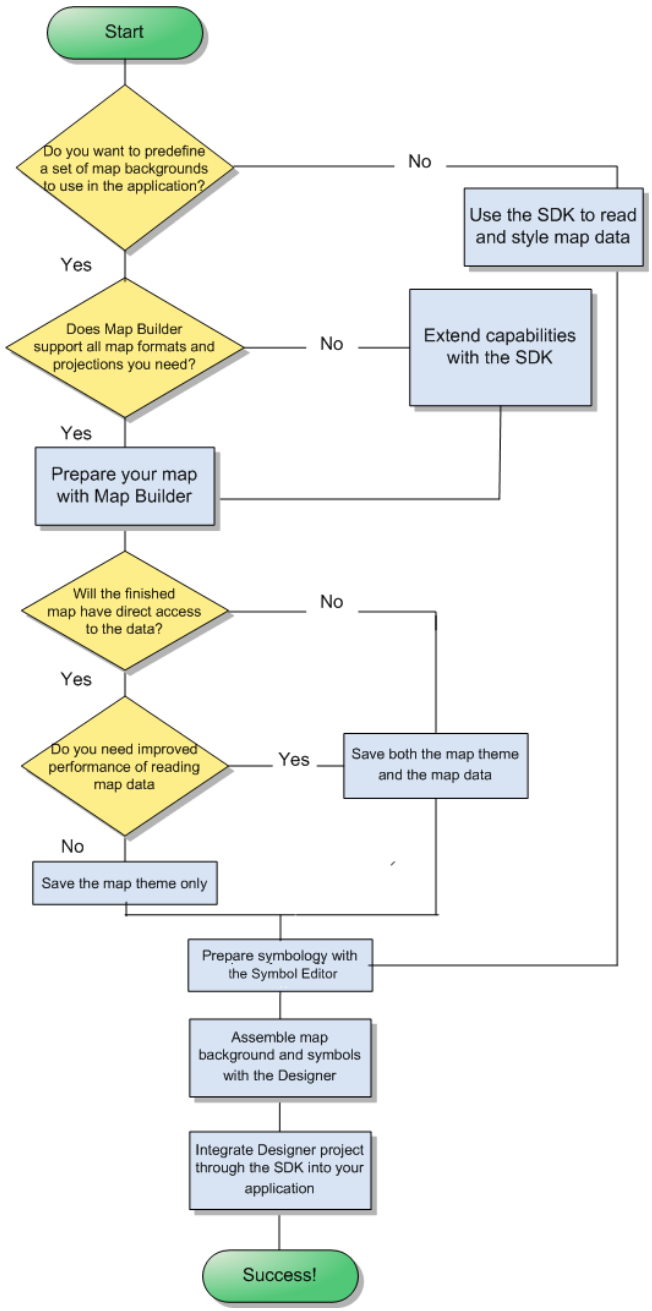
In this section

Overview
Illustrates with a diagram the process of building a map.

Defining a set of background maps
Explains how to define background maps with the Map Builder or with the SDK.

Toolchain for an application without predefined background maps
Explains the tools involved in creating an application without predefined background maps.

Overview



The Map Building Process

Defining a set of background maps

If you want to define a set of background maps to use later in your application, it is best to start with the Map Builder. If the Map Builder does not contain what you need, you can extend the code of the sample with the SDK.

Using the Map Builder

The Map Builder is a ready made sample for preparing and styling maps. You can use this sample code as is or customize it. The Map Builder supports the following types of map:

- TIFF file-based interchange format for georeferenced raster imagery (`.tif`) (see GeoTIFF format)
- Non-Georeferenced image file (`.gif`, `.jpg`, `.png`, `.tif`)
- Environmental Systems Research Institute (ESRI) (see Shapefile format) (`.shp`)
- MapInfo Interchange Format (`.mif`) (see MIF file)
- Topologically Integrated Geographic Encoding and Referencing system (see TIGER/Line) files (`.rt*`)
- Drawing Interchange Format (AutoCAD DXF format) files (`.dxf`)
- Google Earth™ Keyhole Markup Language (KML) and KML Zipped (KMZ) formats (`.kml`, `.kmz`)(see KML/ KMZ)
- Digital Terrain Elevation Data `0`, `1` and `2` (see DTED format)
- Global Topographic Data `DEM` (see GTOPO30)
- Oracle Spatial
- Web Map Server (WMS standard) images
- Scalable Vector Graphic (.SVG) files

It supports the following coordinate systems:

- Geographical
- Albers Equal Area
- Azimuthal Equidistant
- Cassini
- Cylindrical Equal Area
- Eckert IV and Eckert VI
- Equidistant Cylindrical Projection
- French Lambert

- Gnomonic
- Lambert Azimuthal Equal Area, Lambert Conformal Conic, and Lambert Equal Area Conic
- Mercator, Oblique Mercator, and Transverse Mercator
- Miller Cylindrical
- Mollweide
- Orthographic
- Polyconic
- Robinson
- Sinusoidal
- Stereographic
- Universal Polar Stereographic and Universal Transverse Mercator
- Wagner IV

See *Creating a map with the Map Builder* for how to prepare a map using one of these format and coordinate systems.

Extending the Map Builder with the SDK

You can extend the Map Builder code through the SDK to integrate a different format or coordinate system. The following sample shows you how to do that:

Extending the Map Builder

See also Developing a new data source and Developing a new reader in *Programming with JViews Maps*.

You can then prepare the map as indicated in *Creating a map with the Map Builder*.

Toolchain for an application without predefined background maps

The simplest flow, but not necessarily the easiest approach, is to use the SDK to develop an application that reads and styles map data. You will need to feel comfortable using the Java™ API and syntax. See *Developing with the SDKs* for more information.

When you have processed your map data, you can add symbology with the Designer for JViews Diagrammer to design symbols to place on the map. You can do this through an easy-to-use point-and-click GUI. See *Creating a Symbol with the Symbol Editor*.

See *Handling symbols* for more about adding symbology to a map.

You need to integrate the Designer project file into the application that you developed with the SDK. The integration requires a short piece of uncomplicated Java code. See the sample **Loading maps and symbols with JViews Diagrammer**.

Creating a map with the Map Builder

Describes the steps involved in creating a map with the Map Builder

In this section

Overview
Lists the basic steps to create a map with the Map Builder.

Importing map data sources
Describes the various options of importing map data.

Styling properties
Describes the styling properties applicable to map layers.

Setting preferences
Lists the type of preferences that you can set when creating a map.

Saving the map
Describes the different save options of a map.

Overview

The basic steps for creating a map with the Map Builder are:

1. Import a series of map formats.
2. Style the properties of the map.
3. Set your preferences for various functions.
4. Save the look and feel of the map or save the look and feel and the map data.

Importing map data sources

You can import data in different formats to build up the map to use in your application. See Map data in *Programming with JViews Maps* for a list of Web sites where you can download data from.

Styling properties

The Map Builder has a Property Sheet for assigning values to layer style properties and thus styling the different layers of the map. The properties available depend on the format imported on a particular layer. The scope of properties covers the following types:

- Boolean values
- Options in a list
- Free text
- Values that can be set through editors, such as the Color Editor or the Paint Editor.

Layers can be organized in hierarchies, where parent layers contain sublayers. Sublayers can be set to inherit the properties of their parent or can be set to be independent and override the properties of their parent.

Layers can be set to be shown or hidden. The ability to hide layers from the end user can be useful when you want to retain control of some construction layers.

Dynamic styling allows you to map property values to a specific scale level. The values change as you cross the scale threshold to another layer.

Setting preferences

You can set your preferences for particular functions such as the preferred projection, or the preferred way to display distances, altitudes, or coordinates.

These functions, such as coordinates or projection, are used directly by various Beans supplied in the product.

Saving the map

You can save the styling of the map by saving the map theme. In this case, the map is rebuilt at load time by reading the data source again.

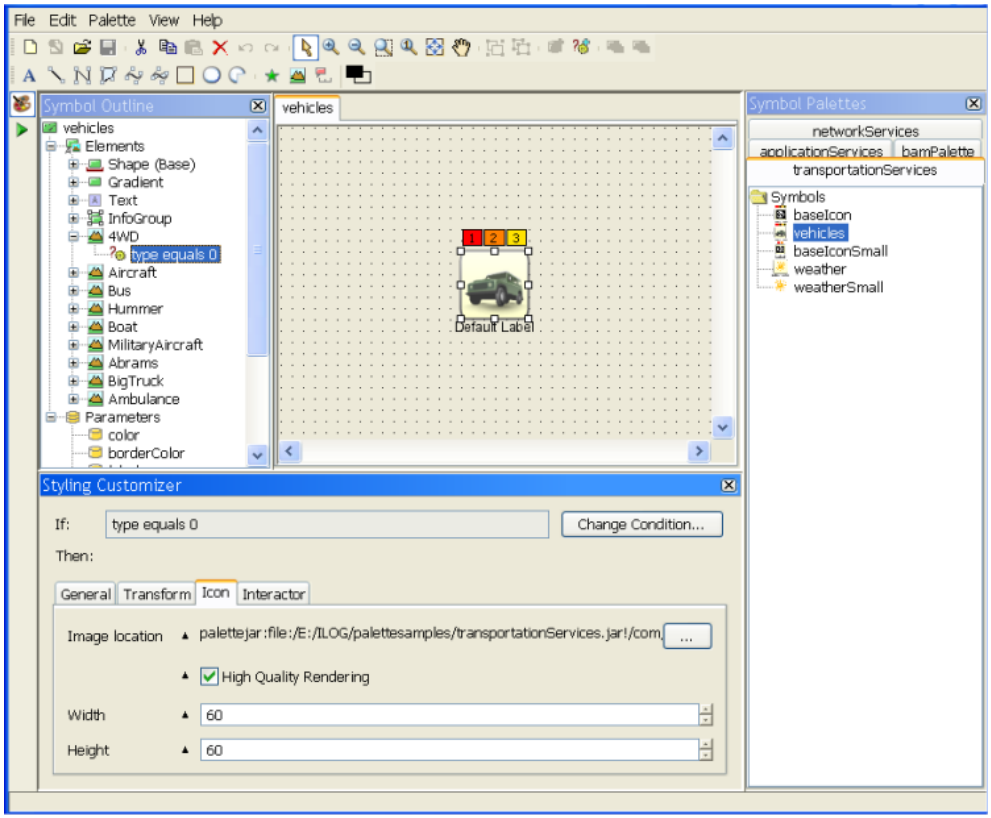
If you want to save the map data as well as the look and feel, you can save the map theme and the data.

If you want to use the map outside the Map Builder, you can export map layers in Shape format, GeoTIFF format, or use the KML/ KMZ export bridge or an Oracle Spatial server. You can also use the JViews SDKs, since maps prepared using the Map Builder can be read by applications developed with JViews.

Handling symbols

You can create symbols for display on a map using the new Symbol Editor.

The symbols are stored in a palette and then reused in the Designer for JViews Diagrammer or any other end-user application. The dedicated Symbol Editor can be used for configuring the symbols you have chosen to represent your nodes. See Symbols in *Using the Designer* for JViews Diagrammer for more about using symbols. The following figure shows an example of symbol creation using the Symbol Editor.



Creating a Symbol with the Symbol Editor

You can add symbols on a map with the Map Builder, but this feature is usually for demonstration purposes and prototyping only. You can load a map developed with the Map Builder or directly in the API as a background for an application developed with JViews Diagrammer. The easiest way to do this is to use the Symbol Editor and the *Designer for JViews Diagrammer*.

To load a map developed with the Map Builder or directly in the API as a background for an application developed with JViews Diagrammer:

1. Edit the map with the Map Builder and then save it.

2. Create your symbols and assemble them with existing ones to create palettes. You can then prototype your application in the Designer.
3. Open Designer, use the map as the backdrop, add nodes or links to the model, design them, and then save the whole as a diagrammer project.
4. Load the diagrammer project in your application.

For more information, see Using the Designer.

Prototyping the application with the designer

If you are starting a project and you want to prototype your application rapidly, you will probably adopt the fastest way of populating the data model.

JViews Diagrammer offers a prebuilt in-memory data model that you can populate manually in the Designer or programmatically, or that can be populated from XML or text (CSV) files through data sources. Another prebuilt data model is available that can be populated from a database using the JDBC data source. If you can use JDBC, XML, or a flat file, you can directly import your data into the Designer.

When you consider services for populating and setting up the map diagram, there are two cases:

- If your application data is compliant with one of the JViews Diagrammer data sources, you can easily populate the data model using the data source facility or using the API.
- If your application data is not compliant with one of the JViews Diagrammer data sources, you should probably create a sample of your data in an XML file or in Microsoft® Excel® , to emulate your data.

As soon as you can populate the data model, you can display the map diagram using the default style sheet. Then you can define your own styling to create specific representations.

When you develop the operational application, you can still use the in-memory data model and one of the data sources provided with JViews Diagrammer. This may imply that your application can generate an XML file that JViews Diagrammer can read or that you use XSL-T to transform your application's XML format.

If your application data is in an object model, you can implement the data model interface of JViews Diagrammer to map your object model closely to the data model of JViews Diagrammer. This approach requires more development, but it prevents any data duplication, and ensures perfect synchronization between the data and the graphics.

Index

U