IDL in Java and COM Environments

Bill Okubo
IDL Product Manager
May 15, 2006

Visual Information Solutions

ITT Visual Information Solutions • 4990 Pearl East Circle Boulder, CO 80301 • P: 303.786.9900 • F: 303.786.9909 •
www.ittvis.com

Table of Contents

IDL in Java and COM Environments 1
I. Summary 3
II. Introduction 3
III. Discussion 6
Java or COM Applications with IDL 6
Export Bridges 6
Connector Objects 7
Extending IDL with Java or COM Code 8
Import Bridges 8
V. Conclusion 9

Table of Figures

Figure 1. The IDL Data Analysis and Development Environment 4
Figure 2. IDL Object and Wrapper Object for Java or COM 6
Figure 3. The Export Bridge Assistant Module interface 7
Figure 4. Connector Objects integrate IDL visualization with Microsoft Excel 8
Figure 5. Custom map data imported into IDL 8

I. Summary

IDL is a powerful, cross-platform language for data analysis and visualization of large, multi-dimensional datasets. IDL's core of graphics and data manipulation capabilities, simple syntax, and array-oriented architecture make it far easier than C, C++, Java, or COM environments for developing complex data analysis and visualization applications.

With the growing trend towards using Java and COM for applications and enterprise architectures, developers need a way to take advantage of the strengths of external environments like IDL. IDL has historically allowed for interoperation with applications written in other languages via mechanisms such as spawning external processes, a call_external function to access compiled code written in other languages, Remote Procedure Calls, and callable IDL. Now, IDL includes new bridging technologies that allow developers to integrate IDL's data visualization and analysis functionality directly into their COM and Java applications. The ease of incorporating IDL's data analysis and visualization capabilities into Java and COM applications dramatically speeds application development time and can significantly increase the functionality of these applications.

II. Introduction

What is the problem?

In the scientific and engineering computing community, Java and COM are rapidly becoming the environments of choice for new applications. At the same time, scientists and engineers who require tools that perform complex visualization of large, multi-dimensional data often choose to use a data analysis software program that cannot adequately communicate with these Java or COM environments. In many cases, scientists or engineers use legacy FORTRAN and C code that is difficult to integrate with more modern object-oriented environments.

In order to develop a new, advanced visualization application today, the scientist or engineer is faced with several challenges, including coding in Java or Microsoft's .NET framework to create the overall program logic, rewriting portions of the existing code to work with the Java or COM application, and developing graphics rendering code from scratch. These tasks may be outside their area of expertise, and certainly take away from the time they need to perform their jobs.

In contrast, the system developer who creates and maintains the Java or COM application environment is an expert at software programming, but is not necessarily familiar with the science that defines the data processing and complex visualizations that form the core of the scientists' or engineers' preferred tools.

These scientists and engineers would like to integrate the existing tools and legacy code into a new advanced visualization application written in Java or COM, but they prefer not to re-write the existing algorithms and data analysis code. Similarly, they may want to integrate other existing analysis code, written in Java, COM, C/C++ or FORTRAN, into their data analysis software program of choice. This can prove to be a difficult task if programming is not their area of expertise.

IDL solves many of these problems by providing:

- A powerful, cross-platform environment for data analysis and visualization of large, multi-dimensional datasets;
- Java or COM wrapper objects that make it simple to run IDL from other applications;
- A graphical interface called the *Export Bridge Assistant Module* that makes it very easy to export custom IDL objects as Java or COM wrapper objects;

- An easy interface to import Java or COM into the IDL data analysis and visualization environment.

What is IDL?

A variety of users from all over the world, including scientists, engineers, and medical imaging professionals, rely on IDL for their complex data visualization and analysis needs. At its core, IDL is an extensible interpreted programming language providing unlimited abilities for interactive analysis. IDL's fast, array-based processing and ability to handle multi-dimensional data, combined with simple access to visualization of that data, makes it especially well suited to image processing and 2D/3D graphics rendering. IDL offers an easy to learn environment with powerful capabilities built into the core, as well as add-on modules with additional functionality. IDL includes:

- An integrated development environment to create, edit, run, debug, and deploy IDL programs;
- Functions for numerical analysis, image processing, and signal processing, plus the *IDL Wavelet Toolkit*;
- Graphical User Interface tools for developing robust cross-platform applications;
- Interactive *iTools* that provide data visualization and analysis power with minimal programming;
- *Dataminer*, a SQL database connectivity module;
- The *IDL DICOM Toolkit* and *IDLDICOM Network Services Module* for medical image data applications

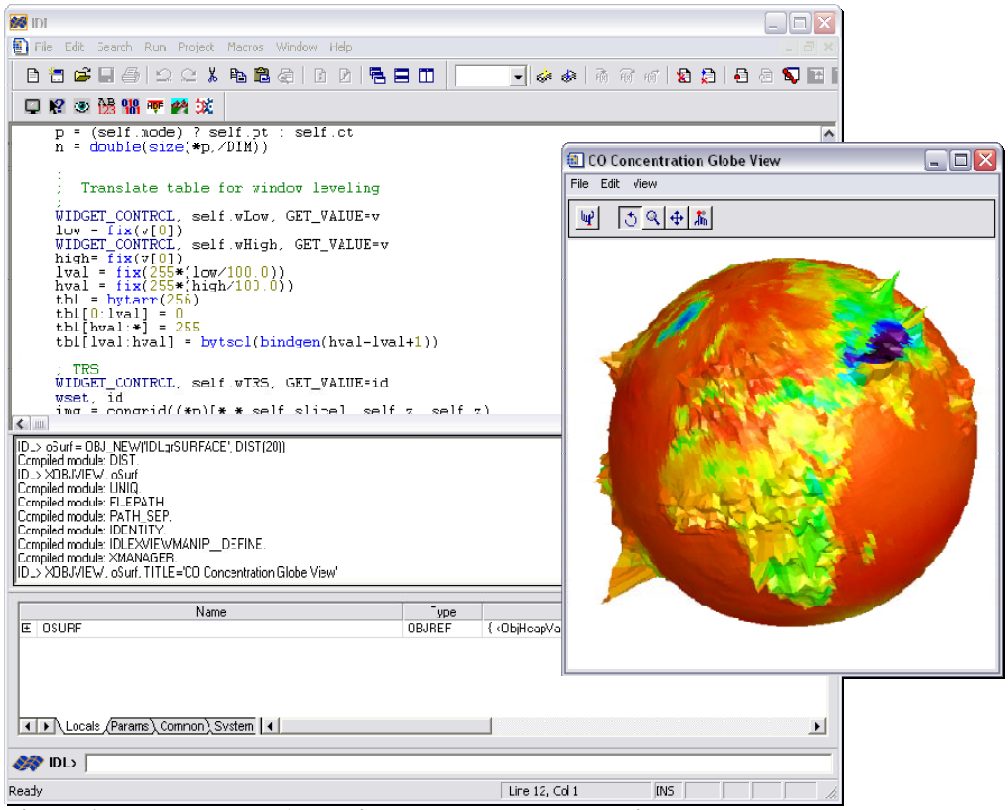

Figure 1. The IDL Data Analysis and Development Environment
Images CNES property, realized by ITT Visual Information Solutions

How does the IDL Bridge technology help?
In addition to its data analysis and visualization capabilities, IDL allows for easy integration with other application development environments including Java, COM, C/C++, and FORTRAN. Using the connectivity capabilities in IDL, developers can both use the power of IDL to enhance applications in other heterogeneous development environments or use those environments to extend IDL.

A *bridge* is a technology path that lets applications in different programming languages or environments share information. For example, bridge technology allows an application written in Java to use native Java-language constructs to make method calls on objects written in IDL. This allows you to take advantage of both environments to solve a problem that might be otherwise difficult for either environment separately: for example, embedding an IDL data object in a Java GUI to display a complex data transformation.

Using the connectivity capabilities in IDL, developers can both use the power of IDL to enhance applications in other heterogeneous development environments, and use the capabilities in those other environments to extend IDL.

IDL's bridges fall into two major classes. IDL's *Import Bridge* allows you to use Java or COM objects to extend IDL's own capabilities within the IDL environment. IDL's *Export Bridge* does the opposite – it allows you to extend applications written in Java or COM by "wrapping" IDL functionality in actual Java or COM objects, which can be included in the Java or COM application just like any other object class.

Who is this for?
IDL itself is designed for two main types of users – the analyst programmer and the application developer.

- Analyst programmers use IDL as an environment for reading, processing, and visualizing data in order to derive information from it. They rely on the high-level language capabilities of IDL for graphics rendering and the libraries of functions available to do this work. However, they are primarily scientists or engineers who have some knowledge of programming skills but they are not formally-trained software programmers. The analyst programmer requires relatively simple bridge technologies if they are to integrate IDL with other environments.

- Application developers often support the work of the analyst programmer by developing new applications in IDL that make the analysis process easier, faster, or repeatable. They are trained software developers who may work for the same organization as the analyst programmers, or they may be contract ITT Visual Information Solutions Global Services engineers. The application developer is likely to be quite familiar with other environments including Java, COM, or C/C++, and they are looking for more advanced ways to integrate IDL with other applications.

In many cases, an IDL user assumes both roles, analyzing data and also developing the tools to assist in this work or to share the work with peers.

In modern software development workflows, the role of the *system developer* who supports software applications and system or enterprise architectures in Java or COM environments has grown very rapidly. System developers are very familiar with object oriented environments like Java or COM but they are unlikely to have experience with IDL. However the efficiency of IDL for data analysis and visualization applications, combined with the familiarity of the bridge technologies for those with Java or COM expertise, makes the system developer a specialized type of IDL user. In many cases, a system developer will work closely with an IDL application developer to incorporate IDL objects into these other environments.

Since the IDL Import and Export Bridges are technology for connecting to other programming languages, it is obvious that they will benefit the two different types of "developers" of IDL. The analyst programmer will benefit from the bridge technology as well, perhaps mostly through indirect use of the bridges, given

the close relationship between the IDL analyst programmer and the application or system developer user types. All three types of IDL users will gain from these bridges in IDL, but each will find utility in the different connectivity options that follow.

III. Discussion

Java or COM Applications with IDL

Export Bridges

Introduced in the IDL 6.3 release, the main goal of the IDL Export Bridge technology is to simplify the ability to integrate IDL analysis and visualization into external environments using the latest component based frameworks and technology, without requiring users to possess a high level of programming knowledge. The Export Bridge technology lets object-oriented COM and Java clients quickly leverage the power of IDL using native language constructs without requiring intimate knowledge of IDL internals.

Using the Export Bridge technology, interaction with IDL is through native wrapper objects that are generated for each IDL object with which client applications want to interact. The wrapper objects manage all aspects of IDL loading, initialization, process management and cleanup. Users need only be familiar with the client language to embed the wrapper into the client application.

The wrapper object is a native language object (COM or Java) that exposes an IDL object's behavior to a client. It uses native-language constructs (Methods, Properties and Events) and data types for interaction with the client. The wrapper object does not contain the IDL object; it only provides an interface between it and the client application. There are two kinds of wrapper objects in IDL: Those that are customized through use of the Export Bridge Assistant Module, and the Connector Objects that are pre-built and shipped with IDL.

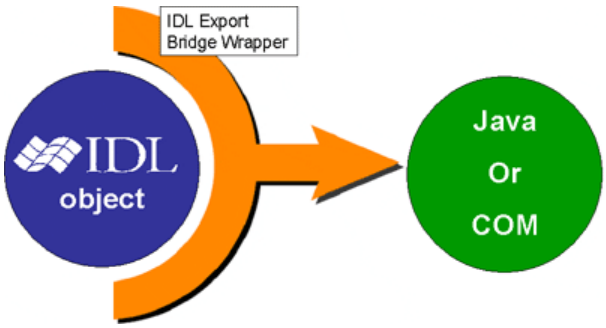

Figure 2. IDL Object and Wrapper Object for Java or COM

The *IDL Export Bridge Assistant Module* is the key to creating your own exported IDL objects that can be used with client applications such as a Java canvas or COM-based applications such as Microsoft Excel. It generates the Java or COM wrapper object for an IDL object that can be either graphical (ActiveX in COM) or non-graphical.

The Export Bridge Assistant Module lets the user customize the wrapper object that is created from the underlying IDL object. The Export Bridge Assistant Module user selects the methods, parameters, and properties to export, as well as other information about the IDL object such as whether to convert array majority for parameters. Exporting a custom-designed IDL object via the Export Bridge Assistant allows

the use of custom object methods, an interactive IDL drawing interface with standard interactive mouse or keyboard event processing.

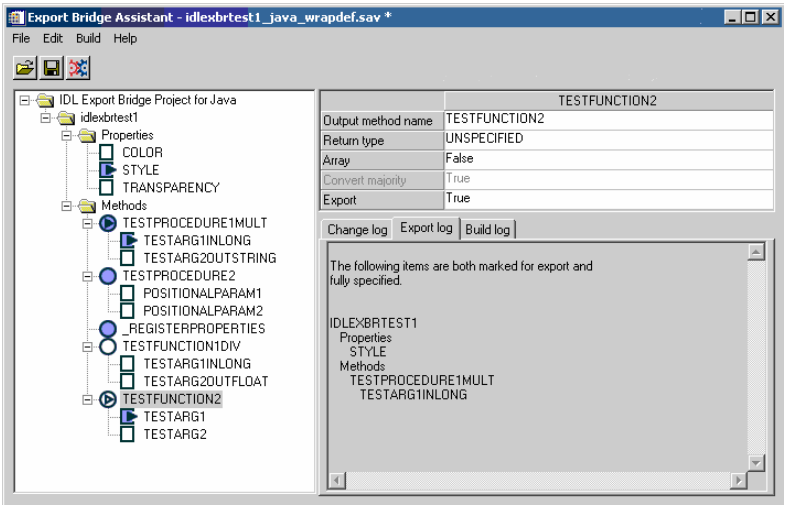

Figure 3. The Export Bridge Assistant Module interface

A basic knowledge of IDL is required to access and manipulate data and IDL processes to utilize the Export Bride technology. In the typical organization, application developers who have advanced skills in IDL create IDL objects that contain visualization and analysis features; these object can than be repurposed for others who have no familiarity with or expertise in IDL. Using the IDL Export Bridge Assistant Module, the Java or COM system developer with limited knowledge of IDL can access the exported IDL objects and incorporate them into other applications or enterprise systems, thereby making them available for use by an organization-wide community. The Export Bridge Assistant Module is licensed separately as a module to IDL.

Connector Objects

Also new in the IDL 6.3 release, the *Connector Objects* are pre-built COM and Java objects that are shipped with the IDL distribution. They allow the user to quickly incorporate the processing power of IDL into an application developed in an external, object-oriented COM or Java environment without the need to create and export a custom IDL object. The stock objects provide a basic, nongraphical wrapper that includes the ability to get and set IDL variables and execute IDL command statements in the IDL process associated with the stock object. The stock objects expose all of the standard wrapper object methods.

The Connector Objects are intended for basic IDL processing capabilities where the user wants to start working with other applications immediately. They are suitable for use by the IDL analyst programmer who requires a simple bridge technology between IDL and other COM or Java environments. For example, the COM connector objects are useful for users of Microsoft Office who want to integrate IDL functionality through the use of Visual Basic applications. The Connector Objects are included in the standard IDL license.

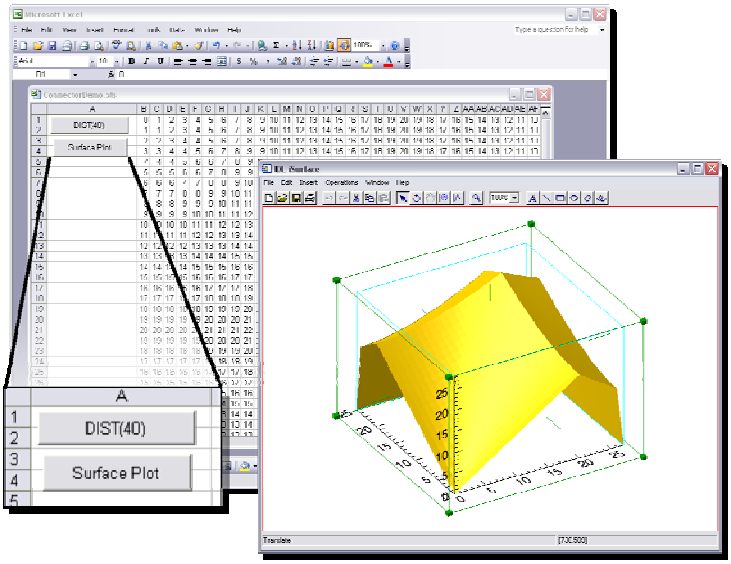

Figure 4. Connector Objects integrate IDL visualization with Microsoft Excel

Extending IDL with Java or COM Code

Import Bridges

IDL also supports the inclusion of Java and COM objects in IDL applications by encapsulating the object or control in an IDL object. Full access to the Java or COM object and its methods and parameters is available in this manner, allowing you to incorporate features that are not available in IDL-only programs. IDL's Import Bridge functionality provides a way to incorporate existing Java and COM objects into IDL applications. If you work exclusively in a Windows environment, in some cases incorporating a pre-written COM component into your IDL program may be faster than coding the same functionality in IDL.

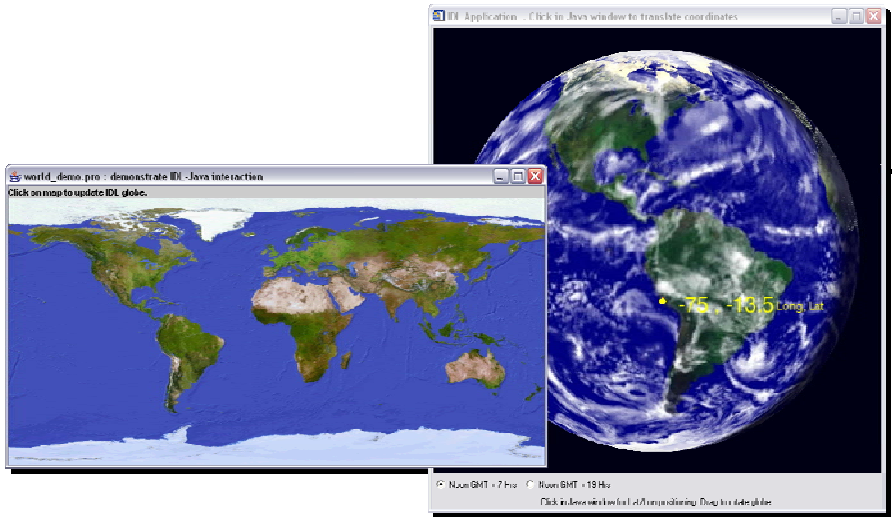

Figure 5. Custom map data imported into IDL

There are two methods for using COM objects with IDL:

1. You can use the IDL-COM Import Bridge to create an IDL object that communicates with an underlying COM object. You can then call the COM object's methods and get and set its properties using standard IDL object conventions and syntax.
2. You can place an ActiveX control in an IDL widget interface, and receive widget events directly from the control.

The IDL-Java Import Bridge allows you to create an IDL object that communicates with an underlying Java object. Java objects imported into IDL behave like normal IDL objects. You can access the Java object's methods and its standard Java data members, and then release the java object when done with it.

V. Conclusion

The prevalence of Java and COM environments in the scientific and engineering continues to grow rapidly. For the scientist or engineer who uses tools for data analysis and visualization on a daily basis, this may present difficulties where there are requirements to integrate existing software or workflows into these Java and COM environments. When existing software programs or legacy code written in FORTRAN or C are the scientists' or engineers' tools of choice, the expertise required to integrate these into Java and COM environments, along with the in-depth knowledge of the science behind the analysis and complex visualization routines, is not easily found in any one individual.

The solution to the problem is to use IDL – a powerful cross-platform environment for data analysis and visualization of large, multi-dimensional datasets. In addition to its efficient processing and graphics capability, IDL offers unique bridge technology that makes it easy to integrate into modern Java and COM applications and system architectures.

To learn more about IDL and how it can integrate with your Java or COM development, visit www.ittvis.com/idl or contact your ITT Visual Information Solutions account representative at 303-786-9900.