What's New in IDL 6.3

IDL Version 6.3
April 2006 Edition

0406IDL63WN

Acknowledgments

IDL® is a registered trademark and ION™, ION Script™, ION Java™, are trademarks of ITT Industries, registered in the United States Patent and Trademark Office, for the computer program described herein.

Numerical Recipes™ is a trademark of Numerical Recipes Software. Numerical Recipes routines are used by permission.

GRG2™ is a trademark of Windward Technologies, Inc. The GRG2 software for nonlinear optimization is used by permission.

NCSA Hierarchical Data Format (HDF) Software Library and Utilities
Copyright 1988-2001 The Board of Trustees of the University of Illinois
All rights reserved.

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities
Copyright 1998-2002 by the Board of Trustees of the University of Illinois. All rights reserved.

CDF Library
Copyright © 2002 National Space Science Data Center
NASA/Goddard Space Flight Center

NetCDF Library
Copyright © 1993-1999 University Corporation for Atmospheric Research/Unidata

HDF EOS Library
Copyright © 1996 Hughes and Applied Research Corporation

This software is based in part on the work of the Independent JPEG Group.

Portions of this software are copyrighted by DataDirect Technologies, 1991-2003.

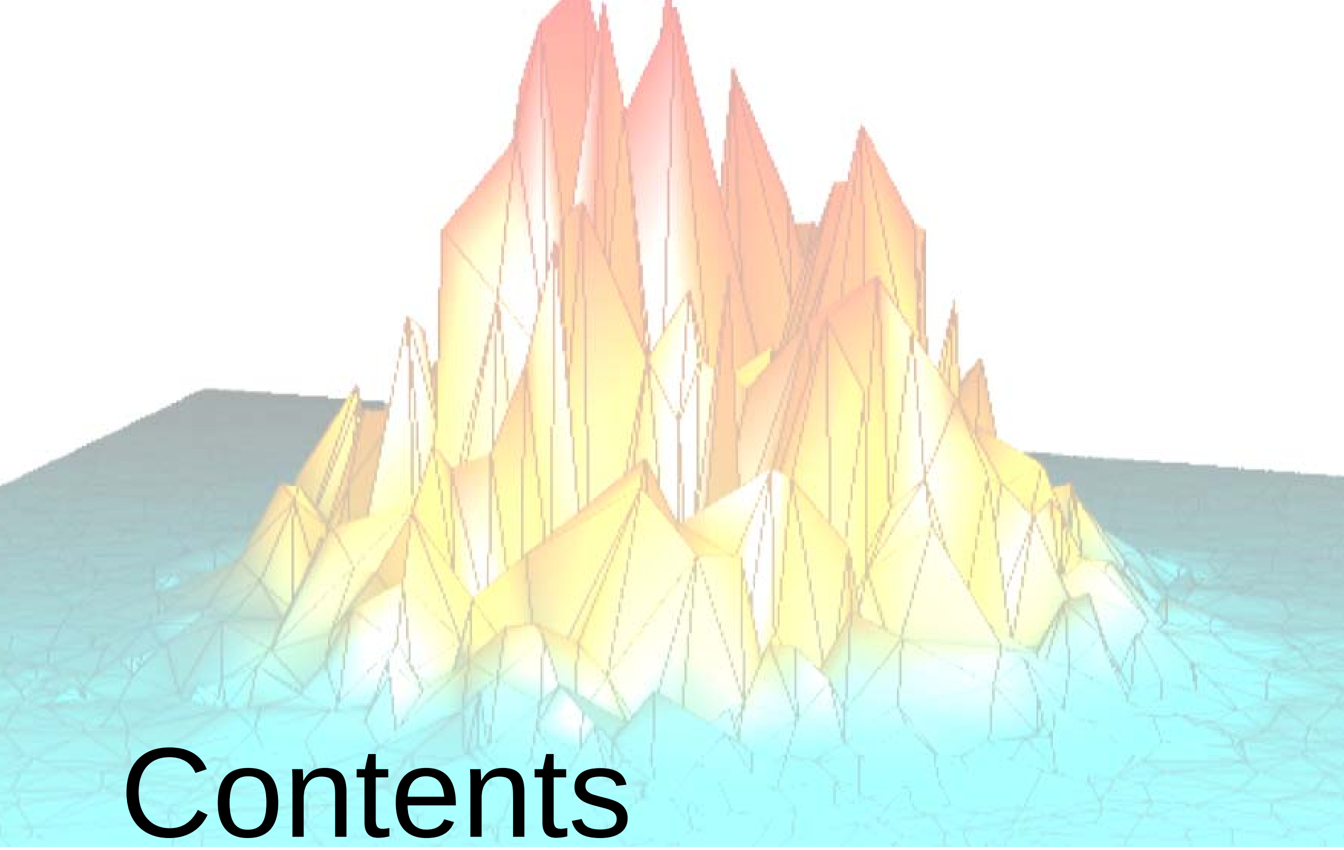
Portions of this software were developed using Unisearch's Kakadu software, for which Kodak has a commercial license. Kakadu Software. Copyright © 2001. The University of New South Wales, UNSW, Sydney NSW 2052, Australia, and Unisearch Ltd, Australia.

Portions of this computer program are copyright © 1995-1999 LizardTech, Inc. All rights reserved. MrSID is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

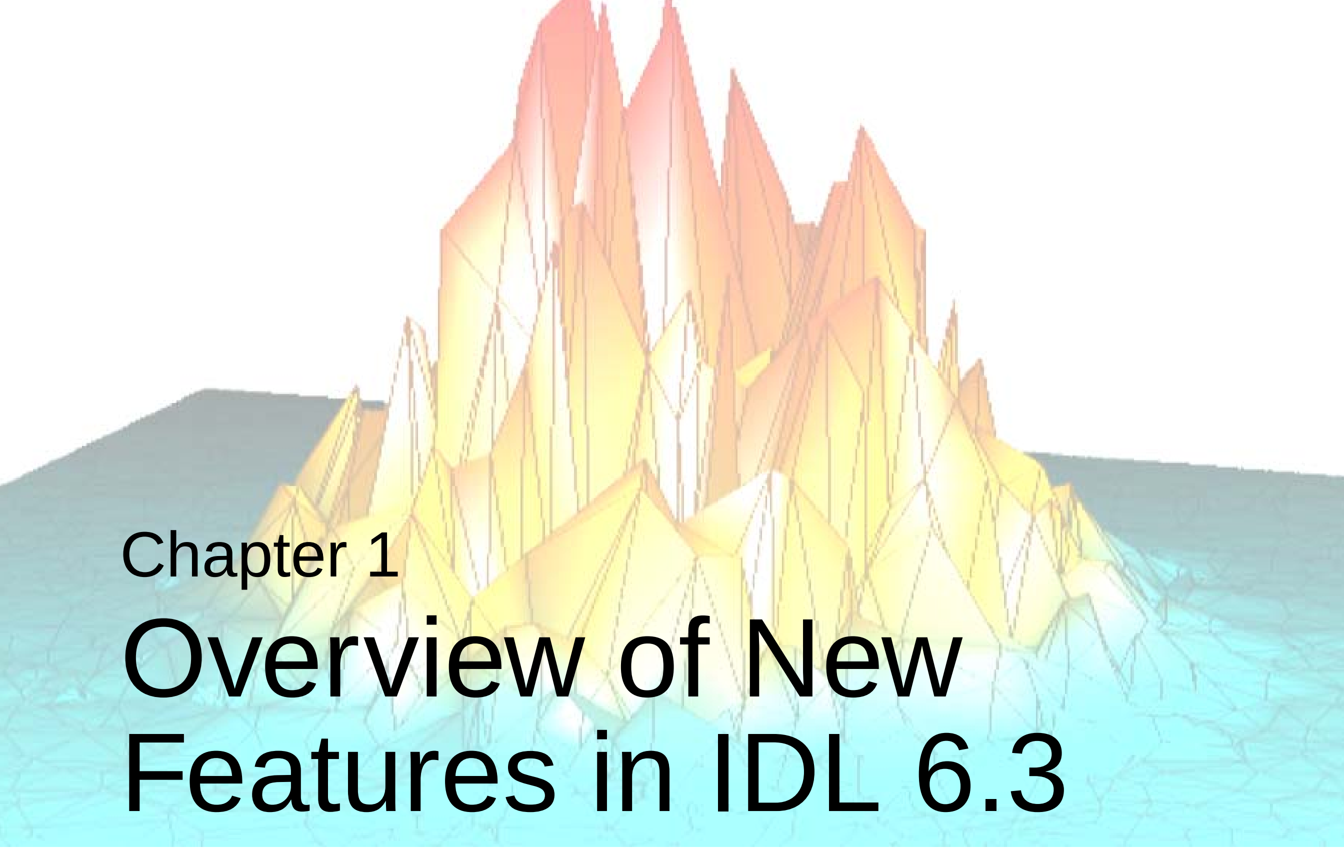
Portions of this software are copyrighted by Merge Technologies Incorporated.

IDL Wavelet Toolkit Copyright © 2002 Christopher Torrence.

Other trademarks and registered trademarks are the property of the respective trademark holders.



Contents



Chapter 1

Overview of New Features in IDL 6.3

This chapter contains the following topics:

Platform Support Changes

The following enhancements have been made to IDL's platform support for the 6.3 release:

Microsoft Windows 64-Bit Support

IDL 6.3 supports Windows XP Professional x64 Edition. See "Hardware and Operating System Requirements for IDL 6.3" on page 28 for additional details.

Note

Only the 32-bit version of the IDL Virtual Machine will run from a CD-ROM. If your application requires a 64-bit version of the IDL Virtual Machine, the 64-bit version must be installed on the end-user's computer.

Macintosh Large File Support

On Macintosh systems that use the Mac OS Extended Filesystem (HFS+), IDL 6.3 is able to read and write data from files up to 2^{63}-1 bytes in length. (Macintosh systems using the UNIX File System (UFS) will not be able to access large files.) See "Reading and Writing Very Large Files" in Chapter 18 of the *Building IDL Applications* manual for details.

iTool Enhancements

The IDL Intelligent Tools (iTools) are a set of interactive utilities that combine data analysis and visualization with the task of producing presentation quality graphics. Introduced in IDL 6.0, the iTools are designed to help you get the most out of your data with minimal effort. They allow you to benefit from the control of a programming language, while accelerating your data analysis through the use of interactive utilities.

For details on these additions and other enhancements that have been made to the IDL iTools system for the 6.3 release, see the following topic:

- “New iVector Tool” below

New iVector Tool

The new iVector tool allows you to easily display vectors (line segments that have both length, or magnitude, and direction). These vectors are referred to collectively as two-dimensional *vector flow fields*. Vector flow field data often represents ocean currents, wind fields, fluid flows, and so on. Such data can be visualized as vectors (arrows), wind barbs, or streamlines, and can be located on a regular or irregular grid, and can also be georeferenced. With the iVector tool, you can control all aspects of vector display, including arrow style and size, coloring by magnitude, direction, or multiple datasets, data location, missing data, streamline particles, and subsampling.

For complete information on the iVector tool, see Chapter 16, “Working with Vectors” in the *iTool User’s Guide* manual.

Visualization Enhancements

The following enhancements have been made to IDL's visualization functionality for the 6.3 release:

- Multi-Monitor Support in IDL

Multi-Monitor Support in IDL

IDL now supports multi-monitor configurations, allowing you to control the location and movement of IDL windows on your multi-monitor desktop environment. A new object class gives you information on display monitors in your configuration.

For more information, see "IDLsysMonitorInfo" in the *IDL Reference Guide* manual and "Multi-Monitor Configurations" in Chapter 8 of the *Using IDL* manual.

Analysis Enhancements

The following enhancement has been made to IDL's data-analysis functionality for the 6.3 release:

- "Butterworth Filter" on page 9
- "Canny Edge Filter" on page 9
- "Impulse Response Filters" on page 9

Butterworth Filter

The new BUTTERWORTH function returns an array that contains the absolute value of the low-pass Butterworth kernel. For more information, see "New IDL Routines" on page 17.

Canny Edge Filter

The new CANNY function implements the Canny edge-detection algorithm. For more information, see "New IDL Routines" on page 17.

Impulse Response Filters

The new IR_FILTER function filters data with an infinite impulse response (IIR) or finite impulse response (FIR) filter. For more information, see "New IDL Routines" on page 17.

Language Enhancements

The following enhancements have been made to the core language for the 6.3 release:

- "IDL to IDL Bridge" on page 10
- "Java and COM Export Bridges" on page 10
- "Semaphores" on page 11

IDL to IDL Bridge

The IDL_IDLBridge object class allows an IDL session to create and control other IDL sessions, each of which runs as a separate process. Each instantiation of an IDL_IDLBridge object corresponds to one such *child* process. Child processes are controlled by the *parent IDL process*, or the IDL process that created the IDL_IDLBridge objects.In each child process, you can execute arbitrary commands (synchronously or asynchronously) and pass data between the parent and child processes. The IDL_IDLBridge object also lets you query the status of a child process and abort a currently running asynchronous process.

The IDL_IDLBridge object is designed to perform computationally intensive tasks in the background, allowing the main IDL session to continue to be responsive. A single parent session can create and control any number of child sessions, which can work in parallel with the parent session.

See "IDL_IDLBridge" in the *IDL Reference Guide* manual for details and examples.

Java and COM Export Bridges

The Export Bridge technology lets object-oriented clients (COM and Java) quickly take advantage of the power of IDL without requiring intimate knowledge of IDL. Interaction with IDL is through native Java and COM *wrapper objects* that are generated for each IDL object with which client applications need to interact. The wrapper objects transparently manage all aspects of IDL loading, initialization, process management, and cleanup. The Export Bridge technology lets clients focus their resources on generating proprietary content that leverages IDL processing and visualization capabilities while minimizing the time required to understand IDL in order to use it.

For more information, see Chapter 6, "Exporting IDL Objects" in the *IDL Connectivity Bridges* manual.

Semaphores

A *semaphore* is a mechanism used to provide synchronization between multiple processes. Often, semaphores are used to protect or restrict access to shared or key resources, such as shared memory segments. It is important to note that the semaphore itself is independent of any IDL process, but that *references* to the semaphore can be created in any IDL process.

Four new routines allow you to use semaphores within IDL processes:

- SEM_CREATE creates a reference to a semaphore in the current IDL process. It also creates the semaphore itself, if it does not already exist in the operating system.
- SEM_DELETE destroys a reference to a semaphore in the current IDL process. Under some conditions, it also destroys the semaphore itself.
- SEM_LOCK creates a lock on an existing semaphore. If one IDL process has the semaphore's lock, other IDL processes cannot create a lock until the initial lock is released.
- SEM_RELEASE releases a lock on an existing semaphore.

See the descriptions of the SEM_ routines in the *IDL Reference Guide* for details and examples.

Note

The semaphore routines are not supported under AIX.

File Access Enhancements

The following enhancements have been made to IDL's file-access capabilities in the IDL 6.3 release:

- "Motion JPEG 2000" on page 12
- "Enhanced CDF Support" on page 12
- "Enhanced HDF5 Support" on page 13
- "DICOM Network Services API" on page 14

Motion JPEG 2000

Motion JPEG2000 is an extension of the still image JPEG2000 image format that is designed for storing animations. A Motion JPEG2000 file (MJ2) consists of a collection of frames. Each frame is an independent JPEG2000 image, and like JPEG2000 images, each frame may be made up of one or more components (bands or channels of data). The individual frame components may also be composed of tiles or contain regions. This file format also provides the ability to store lossless data.

The ability to maintain the original data during compression (lossless compression), the flexibility of displaying frames, components, tiles or regions, and the ability to access individual frames make Motion JPEG2000 an excellent format for scientific animations.

You can create and play Motion JPEG2000 (MJ2) files in IDL using the IDLffMJPEG2000 object. Details about creating Motion JPEG2000 animations can be found in Chapter 9, "Animations" in the *Using IDL* manual.

Enhanced CDF Support

IDL now uses version 3.1 of the Common Data Format library. CDF versions beginning with 3.0 support files larger than 2 Gbytes and use a 64-bit file offset. As a result of this change, the CDF_CREATE routine has been modified to create single-file CDF files by default.

Files created with earlier versions of IDL or CDF libraries can be read by IDL 6.3, but files created by IDL 6.3 cannot be read by previous versions. The new routine CDF_SET_CDF27_BACKWARD_COMPATIBLE allows you to create files that can be read by previous versions.

IDL’s CDF implementation now supports the EPOCH16 data type, which accommodates timestamp resolution of up to picoseconds. The following routines allow you to work with the EPOCH16 data type:

- CDF_ENCODE_EPOCH16
- CDF_EPOCH16
- CDF_PARSE_EPOCH16

In addition, the following routines have been updated to work with the EPOCH16 data type:

- CDF_ATTGET
- CDF_VARCREATE

Enhanced HDF5 Support

IDL now allows you to work with the following Hierarchical Data Format version 5 (HDF5) datatypes:

- Variable Length Array. A *variable length array* datatype is a sequence of an existing datatypes (atomic, variable length, or compound) which are not fixed in length from one dataset location to another. See “Variable Length Array Datatypes” in Chapter 3 of the *IDL Scientific Data Formats* manual for additional details.
- Enumeration. An *enumeration* datatype is a one-to-one mapping between a set of symbols and an ordered set of integer values. The symbols are passed between IDL and the underlying HDF5 library as character strings. All the values for a particular enumeration datatype are of the same integer type. See “Enumeration Datatypes” in Chapter 3 of the *IDL Scientific Data Formats* manual for additional details.
- Opaque. An *opaque* datatype is a mechanism for describing data which cannot be otherwise described by HDF5. The only properties associated with opaque types are the size in bytes and an ASCII tag string. See “Opaque Datatypes” in Chapter 3 of the *IDL Scientific Data Formats* manual for additional details.

In addition, the new *Datatype_ID* argument to the H5D_READ and H5A_READ functions allows you to read individual parts of an HDF5 compound dataset or attribute.

DICOM Network Services API

IDL now includes three classes that allow you to programmatically interact with IDL's DICOM Network Services feature. (The DICOM Network Services feature was introduced in IDL 6.2, but was only exposed via a graphical user interface.)

The new object classes are:

- IDLffDicomExCfg. This object allows you to set and retrieve the values of IDL DICOM Network Service configuration parameters.
- IDLffDicomExQuery. This object allows you to query the contents of a remote DICOM node that has been configured to work with IDL's DICOM Network Service feature, and to retrieve DICOM files available on that node.
- IDLffDicomExStorScu. This object implements a local DICOM Storage SCU (Service Class User) that allows you to transmit files to a remote destination that is identified as a DICOM Storage SCP (Service Class Provider).

For more information on the new classes, see Chapter 3, "IDL DICOM Reference" in the *Medical Imaging in IDL* manual. For more information on the DICOM Network Services feature in general, see Chapter 2, "Using IDL DICOM Network Services" in the *Medical Imaging in IDL* manual.

User Interface Toolkit Enhancements

The following enhancements have been made to the IDL's graphical user interface toolkit in the IDL 6.3 release:

- "Tree Widget Drag and Drop" on page 15
- "Draw Widget Drag and Drop" on page 15

Tree Widget Drag and Drop

In IDL versions 6.3 and later, you can create applications that allow users to "drag and drop" tree widget nodes within a tree widget or between tree widgets. You can drag single or multiple nodes and can drop these nodes above, below, or on other nodes.

Drag and drop functionality is not enabled by default. To enable drag and drop in your tree widget application, you must use keywords to the WIDGET_TREE function (or the WIDGET_CONTROL procedure) to specify which nodes can be dragged, which can be dropped on, and how IDL should behave when a drop event occurs. When your application receives a drop event, it must move or copy the tree nodes involved; the new WIDGET_TREE_MOVE procedure can accomplish this task in the most common cases.

For complete information on using the drag and drop features, see "Dragging and Dropping Tree Nodes" in Chapter 30 of the *Building IDL Applications* manual.

Draw Widget Drag and Drop

In IDL versions 6.3 and later, you can create applications that allow users to "drag and drop" tree widget nodes onto a draw widget.

Drag and drop functionality is not enabled by default. To enable drag and drop in your draw widget application, you must use keywords to the WIDGET_TREE and WIDGET_DRAW functions (or the WIDGET_CONTROL procedure) to specify which nodes can be dragged, which draw widgets can be dropped on, and how IDL should behave when a drop event occurs.

For complete information on using the drag and drop features, see "Implementing Drag and Drop Functionality" in Chapter 30 of the *Building IDL Applications* manual.

Documentation Enhancements

In addition to documentation for new and enhanced IDL features, the following enhancement to the IDL documentation set is included in the 6.3 release:

- "New IDL Connectivity Bridges Book" on page 16

New IDL Connectivity Bridges Book

The IDL 6.3 documentation set includes a new volume: *IDL Connectivity Bridges*. This manual contains information on the following:

- The new Java Export connectivity bridge, which allows you to package IDL objects as Java classes.
- The new COM Export connectivity bridge, which allows you to package IDL objects as COM dlls or ActiveX controls.
- The Java Import connectivity bridge, which allows you to package Java classes as IDL objects for use within IDL. This material was formerly included in the *External Development Guide*.
- The COM Import connectivity bridge, which allows you to package COM dlls as IDL objects for use within IDL. This material was formerly included in the *External Development Guide*.
- Information about the IDL DrawWidget ActiveX control. This material was formerly included in the *External Development Guide*.

New IDL Routines

The following new functions and procedures were added to IDL in this release. See the following topics in the *IDL Reference Guide* for complete reference information unless otherwise noted.

BUTTERWORTH — Returns an array that contains the absolute value of the low-pass Butterworth kernel.

CANNY — Implements the Canny edge-detection algorithm.

CDF_ENCODE_EPOCH16 — Encodes CDF_EPOCH16 variable into a string.

CDF_EPOCH16 — Computes/breaks down CDF_EPOCH16 values.

CDF_PARSE_EPOCH16 — Parses input string into a double precision value properly formatted for use as CDF_EPOCH16 variable.

CDF_SET_CDF27_BACKWARD_COMPATIBLE — Allows users of IDL version 6.3 and later to create a CDF file that can be read by IDL 6.2 or earlier

H5T_COMPOUND_CREATE — Creates a compound datatype object.

H5T_ENUM_CREATE — Creates an enumeration datatype object.

H5T_ENUM_GET_DATA — Retrieves the name/value pairs from an enumeration datatype object.

H5T_ENUM_INSERT — Inserts a new member into an existing enumeration datatype.

H5T_ENUM_NAMEOF — Retrieves the name of a member of an enumeration datatype corresponding to the specified value.

H5T_ENUM_SET_DATA — Sets all the name/value pairs on an enumeration datatype object.

H5T_ENUM_VALUEOF — Retrieves the value of a member of an enumeration datatype corresponding to the specified name.

H5T_ENUM_VALUES_TO_NAMES — Converts values to the corresponding names of an enumeration datatype.

H5T_GET_MEMBER_INDEX — Retrieves the index of a specified member of a compound or enumeration datatype.

H5T_GET_MEMBER_VALUE — Retrieves the value of an enumeration datatype member.

H5T_GET_TAG — Retrieves a tag string from an opaque data type.

H5T_SET_TAG — Sets a tag string on an opaque data type.

H5T_STR_TO_VLEN — Converts an IDL string array to an IDL_H5_VLEN array of strings.

H5T_VLEN_CREATE — Creates a variable length array datatype object.

H5T_VLEN_TO_STR — Converts an IDL_H5_VLEN array of strings to an IDL string array.

IR_FILTER — Filters data with an infinite impulse response (IIR) or finite impulse response (FIR) filter.

IVECTOR — Creates an iTool and associated user interface (UI) configured to display and manipulate vector data.

SEM_CREATE — Creates a reference to a semaphore in the current IDL process. Creates the semaphore itself if necessary.

SEM_DELETE — Destroys a reference to the specified semaphore in the current IDL process. Optionally destroys the semaphore itself.

SEM_LOCK — Attempts to gain the lock on an existing semaphore (created by a call to the SEM_CREATE function) for the current IDL process.

SEM_RELEASE — Releases the lock on the specified semaphore held by the current IDL process.

WIDGET_TREE_MOVE — Copies and moves tree widget nodes from one tree to another or within the same tree.

IDL Routine Enhancements

The following IDL routines have updated keywords, arguments, or return values in this release. See the following topics in the *IDL Reference Guide* for complete reference information unless otherwise noted.

CDF_ATTGET — The CDF_ATTGET procedure has been enhanced to support the EPOCH16 variable type.

CDF_CREATE — The CDF_CREATE function has been modified to create single-file CDF files by default.

CDF_VARCREATE — The CDF_VARCREATE function has been enhanced to support the EPOCH16 variable type.

H5A_READ — The H5A_READ function has the following new optional argument:

- *Datatype_ID* is a long integer containing the identifier of the memory datatype to read. This argument is used only when reading part of a compound attribute. If *Datatype_id* is not supplied, the entire attribute is read.

H5D_READ — The H5D_READ function has the following new optional argument:

- *Datatype_ID* is a long integer containing the identifier of the memory datatype to read. This argument is used only when reading part of a compound dataset. If *Datatype_id* is not supplied, the entire dataset is read.

H5T_GET_MEMBER_NAME — The H5T_GET_MEMBER_NAME function now supports enumeration datatypes.

H5T_IDL_CREATE — The H5T_IDL_CREATE function has the following new keywords:

- OPAQUE specifies that an opaque datatype should be created.The size of the datatype will be determined by the size of the *Data* argument.

WIDGET_CONTROL — The WIDGET_CONTROL function has the following new keywords:

- SET_DRAG_NOTIFY specifies the name of a function that will be automatically called as the end-user drags over the widget.
- SET_DRAGGABLE sets the draggability of a tree widget node.
- SET_DROP_EVENTS tells the widget to generate an event when a dragged item is dropped on it.
- SET_MASK creates a tree node icon that has transparency (use in conjunction with SET_TREE_BITMAP).

- SET_TREE_INDEX changes the position of a tree widget node relative to its siblings.

WIDGET_DRAW — The WIDGET_DRAW function has the following new keywords:

- DRAG_NOTIFY specifies the name of a function that will be automatically called as the end-user drags over the widget.
- DROP_EVENTS specifies whether or not the widget will generate drop events.
- Drop event is generated when the user drops a dragged item over a tree widget.

WIDGET_INFO — The WIDGET_INFO function has the following new keywords:

- DROP_EVENTS returns whether or not a widget generates drop events.
- MASK returns whether a widget's bitmap makes use of transparency.
- DRAGGABLE returns whether the tree is or is not draggable.
- DRAG_NOTIFY returns the name of the drag notification callback.
- TREE_FOLDER returns whether a tree widget node is a folder or leaf node.
- TREE_BITMAP returns the tree widget node's bitmap.
- TREE_DRAG_SELECT returns a modified selection state where nodes that have selected ancestors are reported as “not selected.”
- TREE_INDEX returns the position of the specified tree widget.
- ALL_CHILDREN returns an array that contains the widget IDs of the given widget's immediate children.
- N_CHILDREN returns the number of children that belong to a widget. This keyword complements the ALL_CHILDREN keyword.

WIDGET_TREE — The WIDGET_TREE function has the following new keywords and events:

- BITMAP now accepts scalar 0 (zero) to indicate that the default system bitmap is to be used.
- DRAG_NOTIFY specifies the name of a function that will be automatically called as the end-user drags over the widget.
- DRAGGABLE sets the node as a valid drag item.
- DROP_EVENTS sets the tree widget to generate an event when a dragged item is dropped on it.

- INDEX sets the position of the new tree widget node relative to its siblings.
- MASK creates a tree node icon that has transparency (use in conjunction with BITMAP).
- Drop event is generated when the user drops a dragged item over a tree widget.

New IDL Object Classes

The following new object classes were added to IDL in this release. See the following topics in the *IDL Reference Guide* for complete reference information unless otherwise noted.

IDL_IDLBridge — This object lets an IDL session create and control other IDL sessions, each of which runs as a separate process.

IDLffDicomExCfg — This object allows you to set and retrieve the values of IDL DICOM Network Service configuration parameters.

IDLffDicomExQuery — This object allows you to query the contents of a remote DICOM node that has been configured to work with IDL's DICOM Network Service feature, and to retrieve DICOM files available on that node.

IDLffDicomExStorScu — This object implements a local DICOM Storage SCU (Service Class User) that allows you to transmit files to a remote destination that is identified as a DICOM Storage SCP (Service Class Provider).

IDLffMJPEG2000 — This object lets you create and play Motion JPEG2000 animations.

IDLitDirectWindow — This object is a representation of an on-screen area on a display device that serves as a graphics destination. It uses Direct Graphics.

IDLsysMonitorInfo — This object contains information about the display monitor or monitors attached to your system.

New IDL Object Methods

The following IDL object classes have new methods in this release. See the following topics in the *IDL Reference Guide* for complete reference information.

IDLitComponent::NotifyBridge — This method broadcasts a notification message to the Export Bridge object that is wrapping the IDLitComponent object.

IDLitWindow::OnEnter — This method handles notification from the window that a mouse enter event has occurred.

IDLitWindow::OnExit — This method handles notification from the window that a mouse exit event has occurred.

IDLitWindow::OnExpose — This method handles notification from the window that an expose event has occurred.

IDLitWindow::OnResize — This method handles notification from the window that a resize event has occurred.

IDLgrWindow::OnEnter — This method handles notification from the window that a mouse enter event has occurred.

IDLgrWindow::OnExit — This method handles notification from the window that a mouse exit event has occurred.

IDLgrWindow::OnExpose — This method handles notification from the window that an expose event has occurred.

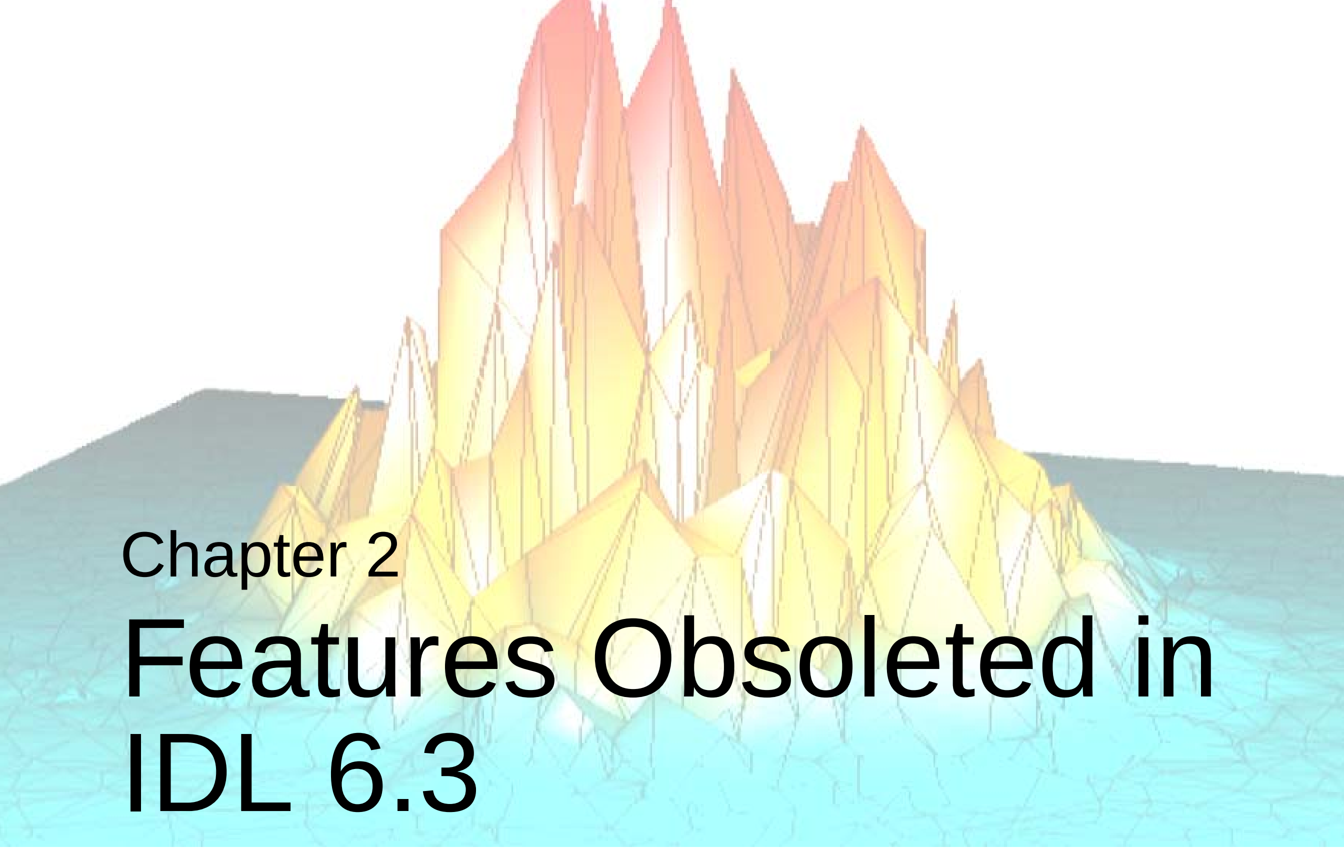
IDLgrWindow::OnKeyboard — This method handles notification from the window that a keyboard event has occurred.

IDLgrWindow::OnMouseDown — This method handles notification from the window that a mouse-down event has occurred.

IDLgrWindow::OnMouseMotion — This method handles notification from the window that a mouse motion event has occurred.

IDLgrWindow::OnMouseUp — This method handles notification from the window that a mouse-up event has occurred.

IDLgrWindow::OnResize — This method handles notification from the window that a resize event has occurred.



Chapter 2

Features Obsoleted in IDL 6.3

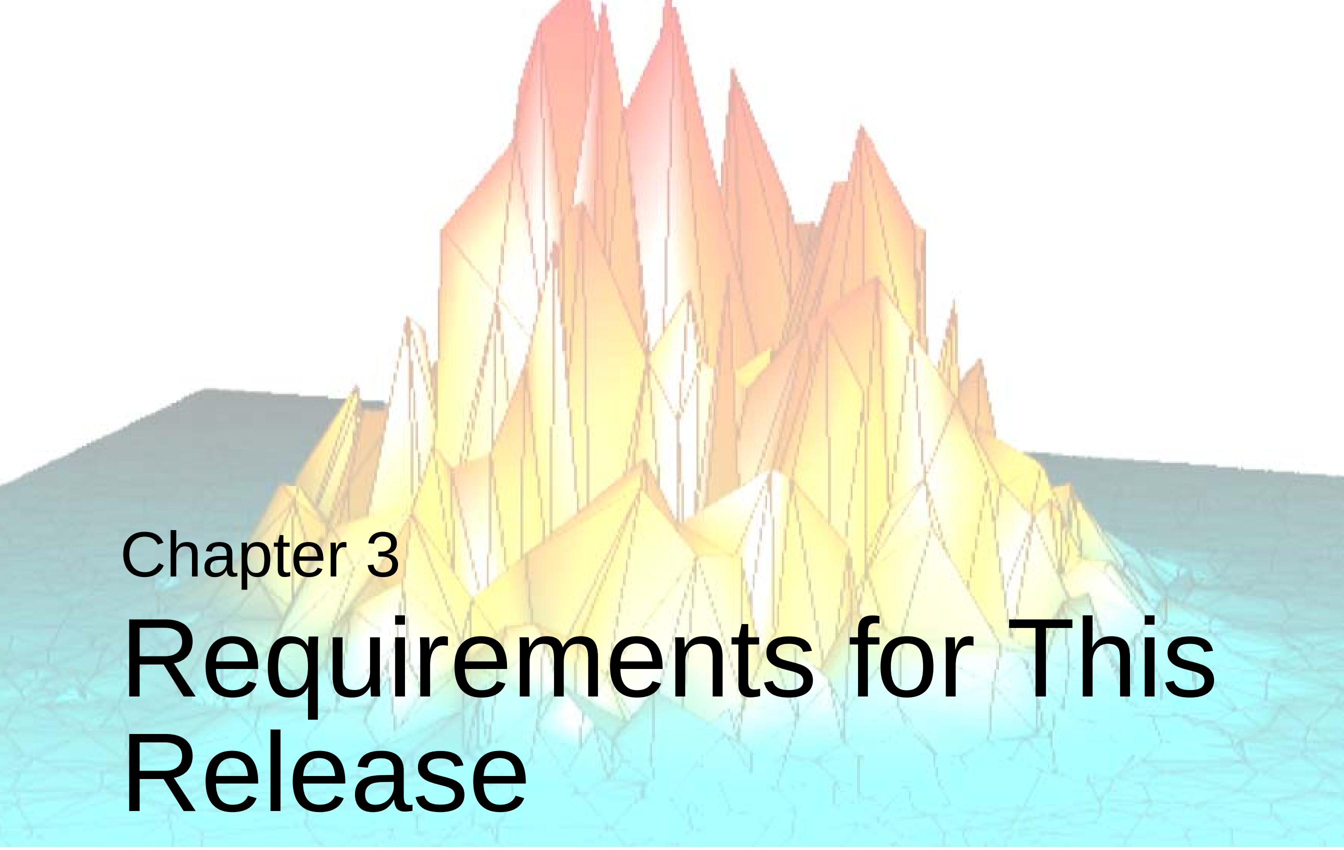
The following features were present in IDL Version 6.2 but became obsolete in Version 6.3. Obsoleted features should not be used in new IDL code.

Obsolete Arguments or Keywords

The arguments or keywords to the following routines have been removed:

Routine	Argument or Keyword
WIDGET_TREE	TOP keyword

Table 2-1: Obsolete Keywords



Chapter 3

Requirements for This Release

This chapter contains the following topics:

IDL 6.3 Requirements

Hardware and Operating System Requirements for IDL 6.3

The following table describes the supported platforms and operating systems for IDL 6.3:

Platform	Vendor	Hardware	Operating System	Supported Versions
Windows	Microsoft	Intel/AMD x86 32-bit	Windows	2000, XP
		Intel/AMD x86_64 64-bit	Windows	XP
Macintosh [a]	Apple	PowerMac G4, G5 32-bit	OS X	10.3, 10.4
UNIX [a]	HP	PA-RISC 32-bit	HP-UX	11.0
	HP	PA-RISC 64-bit	HP-UX	11.0
	IBM	RS/6000 32-bit	AIX	5.1
	IBM	RS/6000 64-bit	AIX	5.1
	SGI	Mips 32-bit	IRIX	6.5.1
	SGI	Mips 64-bit	IRIX	6.5.1
	SUN	SPARC 32-bit	Solaris	8, 9, 10
	SUN	SPARC 64-bit	Solaris	8, 9, 10
	various	Intel/AMD x86 32-bit	Linux [b]	Kernel version 2.4 Kernel version 2.6 glibc version 2.3
	various	Intel/AMD x86_64 64-bit	Linux [b]	Kernel version 2.4 Kernel version 2.6 glibc version 2.3

Table 3-1: Hardware Requirements for IDL 6.3

On UNIX platforms that provide 64-bit support, IDL can be run as either a 32-bit or a 64-bit application. When both versions are installed, the 64-bit version is the default.

The 32-bit version can be run by specifying the `-32` switch at the command line, as follows:

```
% idl -32
```

Under Microsoft Windows, the 32-bit and 64-bit versions are started via separate shortcuts and Start menu entries.

[a] For UNIX and Mac OS X, the supported versions indicate that IDL was either built on (the lowest version listed) or tested on that version. You can install and run IDL on other versions that are binary compatible with those listed.

[b] IDL 6.3 for 32-bit Linux systems was tested on Red Hat Enterprise 3, Red Hat Enterprise 4, and Fedora Core 3. IDL 6.3 for 64-bit Linux systems was tested on Red Hat Enterprise 3 and Red Hat Enterprise 4. If your version of Linux is compatible with the listed kernel and glibc versions, you should be able to install and run IDL 6.3.

Software Requirements for IDL 6.3

The following table describes the software requirements for IDL 6.3:

Platform	Software Requirements
Windows	Internet Explorer 5.0 or higher
Macintosh	Apple X11 X-Windows manager

Table 3-2: Software Requirements for IDL 6.3

ION 6.3 Requirements

ION™ (IDL On the Net) is a family of products that allow you to run IDL-driven applications in a networked environment, giving intranet or Internet users access to IDL visualization and analysis.

The ION family consists of two products:

- ION Script™
- ION Java™

ION Script and ION Java can be used separately or together to create interactive Web applications powered by IDL. These two products are available as options during installation of IDL for Windows or UNIX.

Hardware and Operating System Requirements for ION 6.3

ION 6.3 works with IDL 6.3 on the following platforms:

- Microsoft Windows (32-bit)
- SGI IRIX (32-bit)
- Sun Solaris (32-bit)
- Linux (32-bit)

See the Hardware and Operating System Requirements for IDL 6.3 for details.

Web Server Requirements for ION 6.3

In order to use ION, you must install an HTTP Web server. ION has been tested with the following Web server software:

- Apache Web Server version 2.0 for Windows, Linux, and Solaris
- Apache Web Server version 1.3.14 for IRIX (included with the IRIX operating system)
- Microsoft Internet Information Server (IIS) version 5.0 for Windows 2000 Server and version 5.1 for Windows XP Professional

Web Browser Requirements for ION 6.3

ION 6.3 supports the HTTP 1.0 protocol. The following are provided as examples of popular Web browsers that support HTTP 1.0:

- Mozilla Firefox version 1.0 and later
- Netscape Navigator — Versions 4.7 and later

- Microsoft Internet Explorer — Versions 5.5 and later

Browsers differ in their support of HTML features. As with any Web application, you should test your ION Script or Java application using Web browsers that anyone accessing your application is likely to be using.

Java Virtual Machine Requirements for ION 6.3

The following are provided as examples of popular Web browsers that are shipped with the required JVMs:

- Mozilla Firefox version 1.0 and later
- Netscape Navigator versions 4.7 and later
- Microsoft Internet Explorer versions 5.5 and later

Browsers differ in their support of Java features. As with any Web application, you should test your ION Java application using Web browsers that anyone accessing your application is likely to be using.

Feature Support by Operating System

Technologies not listed in this table are assumed to work on all supported platforms.

Feature	Windows		OS X	Linux		Solaris		HP-UX		AIX		IRIX	
	32-bit	64-bit	32-bit	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit
ActiveX: IDLDrawWidget	●	●											
ActiveX: WIDGET_ ACTIVEX (IDLcomActiveX object)	●	●											
COM Object –Export (via Connectivity Assistant)	●												
COM Object –Import (IDLcomIDispatch object)	●												
DataMiner	●			●[a]		●		●		●		●	
DICOM Network Services	●		●	●		●							
DICOM Read/Write	●		●	●		●							
DXF file format (IDLffDXF object)	●			●		●		●		●		●	
GUIBuilder	●	●											
IDL_IDLBridge	●	●	●	●		●	●						

Table 3-3: Feature Support

Feature	Windows		OS X	Linux		Solaris		HP-UX		AIX		IRIX	
	32-bit	64-bit	32-bit	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit
ION 6.3	●			●		●						●	
Java Object –Export (via Connectivity Assistant)	●		●[c]	●		●	●						
Java Object –Import (IDLjavaObject)[b]	●		●	●		●	●					●	
Script Node for NI LabView	●												
Motion JPEG2000 (IDLffMJPEG2000)	●	●	●	●	●	●	●	●	●				
MrSID (IDLffMrSID)	●												
Personal use license	●	●	●	●	●								
Remote Procedure Calls (RPCs)			●	●	●	●	●	●	●	●	●	●	●
Semaphores	●	●	●	●	●	●	●	●	●			●	●
tty-based interface			●	●	●	●	●	●	●	●	●	●	●

Table 3-3: Feature Support (Continued)

[a] DataMiner is not supported on Red Hat Enterprise 4 or Fedora Core 3.

[b] Java Virtual Machine Requirements for the IDL-Java Bridge — IDL supports version 1.3.1 and greater on all platforms except for Macintosh (the supported version is 1.3.x) and SUN SPARC 64-bit (the supported version is 1.4.x and greater).

[c] Graphical Java objects cannot be exported under Macintosh OS X.