IDL

What's New in IDL 6.2

RSI

IDL Version 6.2
July 2005 Edition

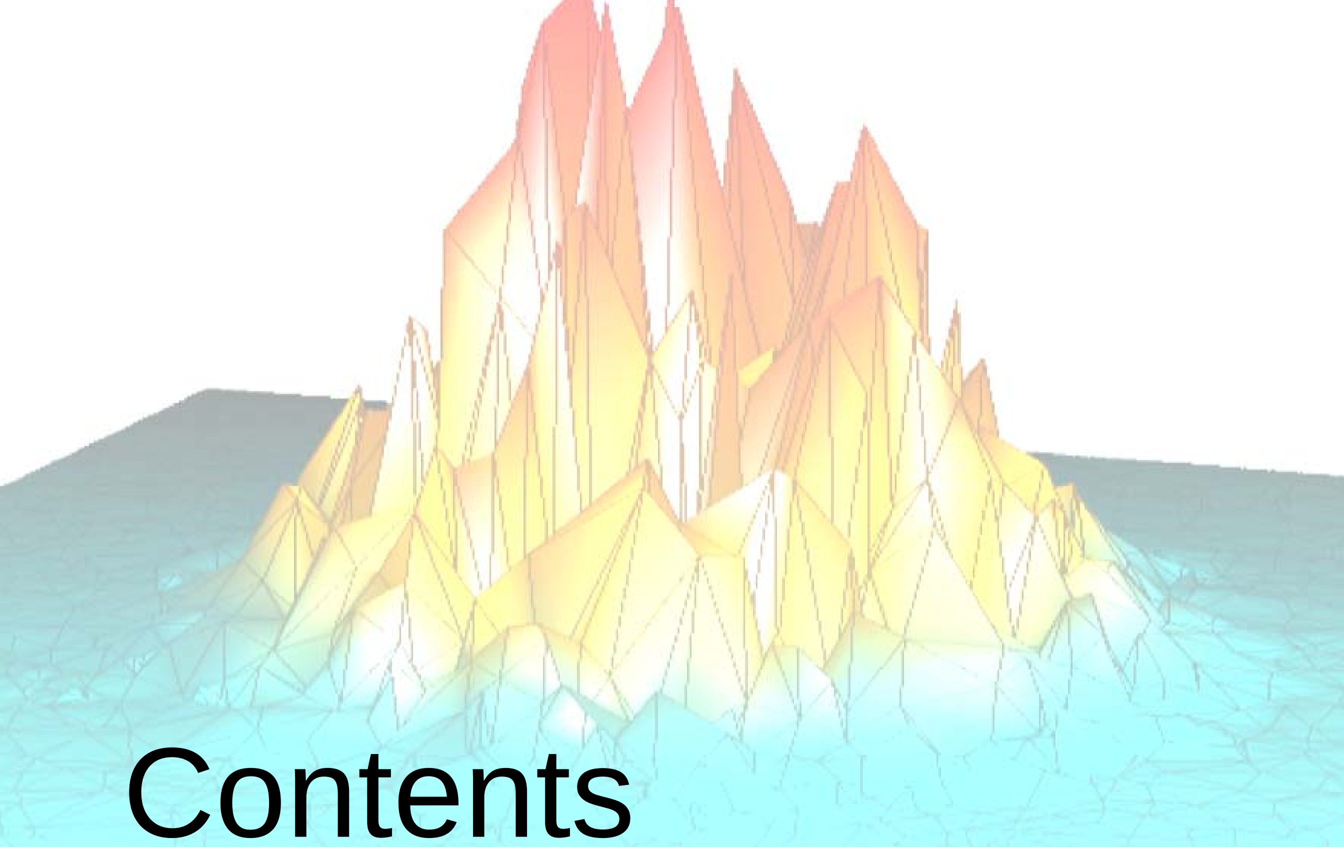
0705IDL62WN

Restricted Rights Notice

Limitation of Warranty

Permission to Reproduce this Manual

Acknowledgments



Contents

Chapter 1
Overview of New Features in IDL 6.2 7

New iTool Features 8

Title Annotations Created at the IDL Command Line 8

GIF File Format Import/Export Accessibility in iTools 8

Intermediate Manipulations in Macros 8

Macro Controls Dialog 9

GeoTIFF Registration in iMap 9

Visualization Enhancements 10

Image Rendering Enhancements 10

Image Tiling 10

Animation Functionality Enhancements 11

Enhancement to Color Conversion 11

Analysis Enhancements 12

Enhancements to CONVOL 12

Language Enhancements 13
New User Preference System 13
Persistent Command Recall Buffer 13
Ability to Selectively Save Heap Variables 14
New COMMAND_LINE_ARGS Function 14
New FILE_POLL_INPUT Function 14
Ability to Access the Interpreter Call Stack 14
New Bit-Level Information Functions 15
Enhancements to GET_KBRD 15
Enhancements to XDISPLAYFILE 15
Enhancements to .RESET_SESSION 15
File Access Enhancements 16
DICOM Network Services 16
JPEG2000 Analysis Enhancements 17
HDF5 Write Support and Library Upgrade 17
DataMiner Driver Enhancements 17
New QUERY_ASCII Function 17
Ability to Navigate XML DOM Trees 17
Ability to Validate XML Schemas 18
IDLDE Enhancements 19
Persistent Command Recall Buffer 19
User Interface Toolkit Enhancements 20
Enhancements to Table Widget Cells 20
Mouse-Wheel Events in Draw Widgets 21
Documentation Enhancements 22
New Cross-Platform HTML Help System 22
New IDL Routines 23
IDL Routine Enhancements 25
New IDL Object Classes 30
New IDL Object Properties 31
New IDL Object Methods 33
IDL Object Method Enhancements 35

Chapter 2
Features Obsoleted in IDL 6.2 37

Obsolete Routines 38

Obsolete Arguments or Keywords 39

Chapter 3
Avoiding Backward Compatibility Issues 41

Visual Class Selection in Direct Graphics 42

Background Information 42

Chapter 4
Requirements for This Release 45

IDL 6.2 Requirements 46

Hardware and Operating System Requirements for IDL 6.2 46

Software Requirements for IDL 6.2 47

ION 6.2 Requirements 48

Hardware Requirements for ION 6.2 48

Web Server Requirements for ION 6.2 48

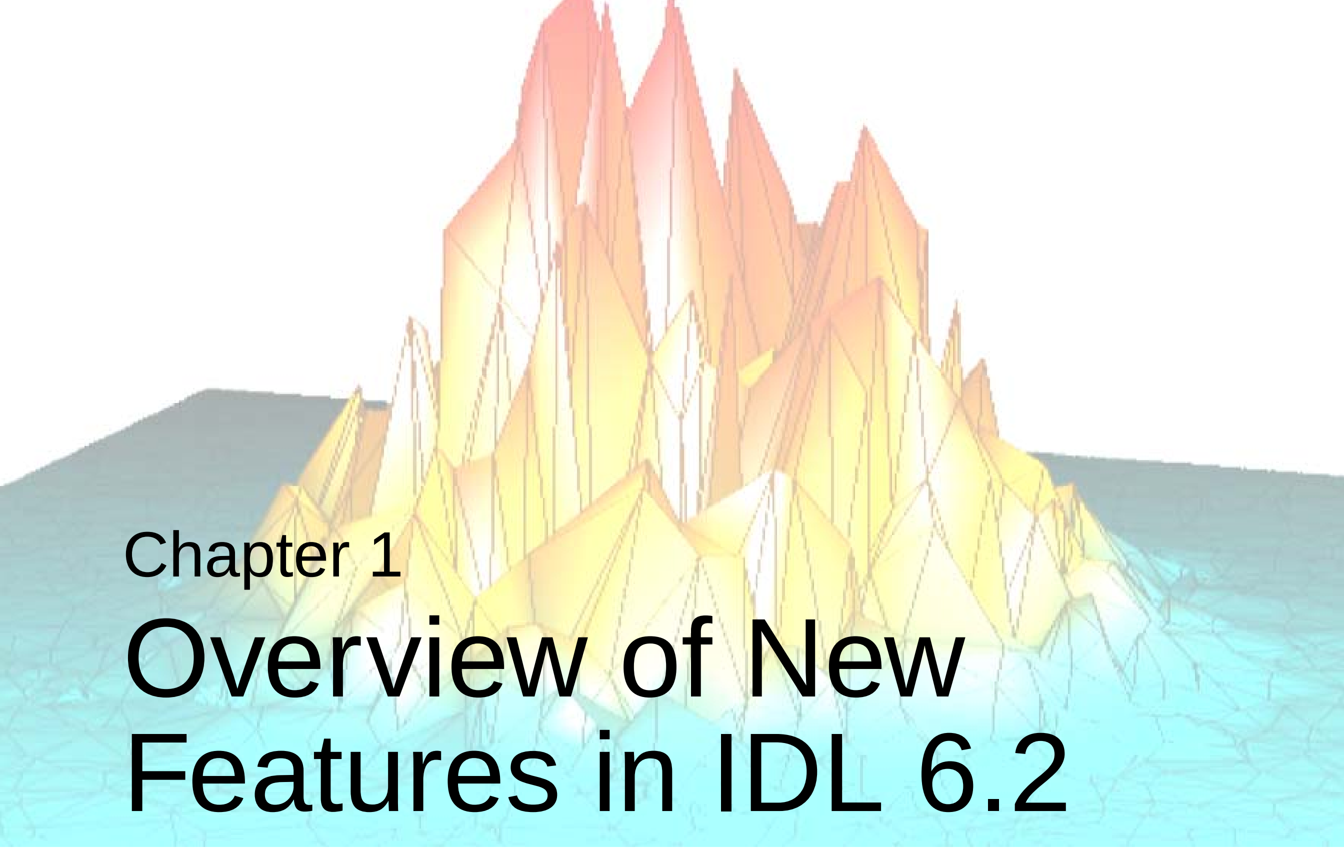
Web Browser Requirements for ION 6.2 49

Java Virtual Machine Requirements for ION 6.2 49

IDL-Java Bridge Requirements 50

Hardware Requirements for the IDL-Java Bridge 50

Java Virtual Machine Requirements for the IDL-Java Bridge 51



Chapter 1

Overview of New Features in IDL 6.2

This chapter contains the following topics:

New iTool Features 8
Visualization Enhancements 10
Analysis Enhancements 12
Language Enhancements 13
File Access Enhancements 16
IDLDE Enhancements 19
User Interface Toolkit Enhancements 20
Documentation Enhancements 22
New IDL Routines 23
IDL Routine Enhancements 25
New IDL Object Classes 30
New IDL Object Properties 31
New IDL Object Methods 33
IDL Object Method Enhancements 35

New iTool Features

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 6.2 release, see the following topics:

- "Title Annotations Created at the IDL Command Line" on page 8
- "GIF File Format Import/Export Accessibility in iTools" on page 8
- "Intermediate Manipulations in Macros" on page 8
- "Macro Controls Dialog" on page 9
- "GeoTIFF Registration in iMap" on page 9

Title Annotations Created at the IDL Command Line

The VIEW_TITLE keyword to the iTool launch routines (ICONTOUR, IIMAGE, IMAP, IPLOT, ISURFACE, and IVOLUME) allows you to specify a string that will be placed in a text annotation centered horizontally in the iTool view, near the top. The text will be created with the properties (size, style, color, etc.) defined for text annotations in the current tool style.

GIF File Format Import/Export Accessibility in iTools

A new GIF file reader and file writer have been added in IDL 6.2 adding input and output functionality in the iTools.

Intermediate Manipulations in Macros

An iTool manipulation such as Translate typically consists of a series of transformations based on the screen position of the mouse. When such a manipulation is recorded, a single macro item is created that contains the overall translation. When this macro item is processed during macro playback, the complete transformation is applied, but the individual transformations based on the individual mouse motions are not available.

A different mode of recording has been added that records the individual steps of a manipulation based on each mouse movement. This is useful if the purpose of the macro is to demonstrate the transformation such as a smooth 3D rotation of a surface. This new mode records each step of the manipulation while the mouse is down, based on the original motion events from the operating system.

For more information, see "Capturing Intermediate Manipulations" in Chapter 8 of the *iTool User's Guide* manual.

Macro Controls Dialog

While a macro is running, you have access to the new **Macro Controls** dialog, which lets you pause, step, and stop the macro. You can also hide or show intermediate steps in the macro, set the step delay, and hide or show the macro items.

For more information, see "Using the Macro Controls Dialog" in Chapter 8 of the *iTool User's Guide* manual.

GeoTIFF Registration in iMap

You can now import and register a GeoTIFF image in the iMap tool. In this case, such an image is either a TIFF file containing GeoTIFF tags or a JPEG2000 file containing a GeoTIFF UUID box.

For more information, see "Registering an Image" in Chapter 15 of the *iTool User's Guide* manual.

Visualization Enhancements

The following enhancements have been made to IDL's visualization functionality for the 6.2 release:

- "Image Rendering Enhancements" on page 10
- "Image Tiling" on page 10
- "Animation Functionality Enhancements" on page 11
- "Enhancement to Color Conversion" on page 11

Image Rendering Enhancements

IDL's ability to render images that fit into real memory has been improved, allowing rendering of images that exceed a computer's virtual storage capacity. IDLgrImage contains several new properties and enhancements to existing properties that allow this new functionality. Additionally, IDLgrImage now supports the ALPHA_CHANNEL property, which controls the overall transparency of an object.

For more information on image rendering, see "IDLgrImage" in Chapter 8 of the *IDL Reference Guide* manual.

Image Tiling

The IDLgrImage object now supports the display of very large images. Some satellite imagery ranges in size from hundreds of megabytes to gigabytes, which can be too large to place in memory as a single entity and display. However, IDL can now display these images as a series of tiles. After setting new properties on the image object that support tiling, you pass the image object and a view object, to the QueryRequiredTiles method on the destination object (an IDLgrWindow object, for example) that will display the tiled image. This method determines what image tiles are needed to fill the viewport, and returns this information in an array of structures. You can use this information to extract the data from the image file and then pass this data to the new IDLgrImage::SetTileData method.

In an application that supports zooming in addition to panning, IDL supports level-of-detail (LOD) rendering. This simply means that the image resolution level matches the zoom level. When you zoom out of a large image, showing a high percentage of it in a view at a single time, a lower resolution image can be displayed. This does require an image pyramid, a series of images where each image is half the resolution of the previous image. You can either create an image pyramid manually, or use the

IDLffJPEG2000 object, which provides inherent support for an image pyramid and image tiles.

The destination objects that support tiled images include IDLgrWindow, IDLgrBuffer, IDLgrClipboard and IDLgrPrinter objects. For more information about tiling in IDL, including copying and printing visible tiles, see "Image Tiling" in Chapter 4 of the *Object Programming* manual.

Animation Functionality Enhancements

IDL's object graphics support has been enhanced to include animation capabilities. A new rendering mode property causes the object to either draw all items in its container or draw only one item in the container. A new timer has been added to control the frame rate of animation.

See Chapter 10, "Animating Objects" in the *Object Programming* manual for additional information.

Enhancement to Color Conversion

The new CMYK_CONVERT procedure lets you convert images from the CMYK (cyan-magenta-yellow-black) color model to RGB (red-green-blue) and vice versa.

Analysis Enhancements

The following enhancement has been made to IDL’s data-analysis functionality for the 6.2 release:

- “Enhancements to CONVOL” on page 12

Enhancements to CONVOL

The CONVOL function has the following important enhancements:

- New keywords to handle missing or invalid data and edge elements of the input array
- New keywords to allow computation of a scale factor and bias

The convolution filter in the iTools system also uses these improvements. For more information, see “Convolution Filter” in Chapter 7 of the *iTool User’s Guide* manual.

Language Enhancements

The following enhancements have been made to the core language for the 6.2 release:

- "New User Preference System" on page 13
- "Persistent Command Recall Buffer" on page 13
- "Ability to Selectively Save Heap Variables" on page 14
- "New COMMAND_LINE_ARGS Function" on page 14
- "New FILE_POLL_INPUT Function" on page 14
- "Ability to Access the Interpreter Call Stack" on page 14
- "New Bit-Level Information Functions" on page 15
- "Enhancements to GET_KBRD" on page 15
- "Enhancements to XDISPLAYFILE" on page 15
- "Enhancements to .RESET_SESSION" on page 15

New User Preference System

Preferences are values stored by IDL to control various aspects of the environment that IDL presents to its users. Preferences supply initial values for many system variables, control the layout of the IDL development environment (IDLDE), and affect a variety of other aspects of IDL's behavior.

The IDL preference system provides a general, cross-platform, programmatic mechanism for setting and retrieving preference values. For a discussion of IDL preferences, see Appendix E, "IDL Preferences" in the *IDL Reference Guide* manual.

Four new routines let you get information on and make changes to the preference system. The PREF_GET routine returns information on the current IDL preferences. The PREF_SET procedure lets you submit changes to preference values; to commit those changes, you can use either the COMMIT keyword to the procedure or the PREF_COMMIT procedure. The PREF_MIGRATE procedure lets you import preference settings from previous versions of IDL into the current working version.

Persistent Command Recall Buffer

The IDLDE command recall buffer now persists between sessions. You can access commands in the buffer by using the keyboard's directional arrows. You can use the IDL preference system to turn the buffer's persistence on or off (persistence is on by default) and set the size of the buffer. For details on these preferences, see IDL_RBUF_PERSIST and IDL_RBUF_SIZE in Appendix E, “IDL Preferences” in the *IDL Reference Guide* manual.

Ability to Selectively Save Heap Variables

The HEAP_NOSAVE and HEAP_SAVE routines allow you to control whether heap variables referenced by pointers or objects are saved in a SAVE file when the pointer or object reference is saved. See HEAP_NOSAVE and HEAP_SAVE for details.

New COMMAND_LINE_ARGS Function

The COMMAND_LINE_ARGS function returns strings supplied by the user when IDL was started with the `-arg` or `-args` command line options. If either of these options is specified at the command line when IDL is started, IDL saves them without examining their values or attaching any special meaning to them. They can be retrieved at any time within the IDL session via the COMMAND_LINE_ARGS function. This mechanism can be used to pass special application-defined values to a program written in the IDL language.

See COMMAND_LINE_ARGS for details.

New FILE_POLL_INPUT Function

Given a list of file logical units (LUNs), the FILE_POLL_INPUT function will *block* (not return) until it detects that a read operation for a byte of data from at least one of the specified files will succeed. On return, FILE_POLL_INPUT reports True (1) for each file for which a read operation will succeed, or False (0) for those that will not succeed.

See FILE_POLL_INPUT for details.

Ability to Access the Interpreter Call Stack

The SCOPE_TRACEBACK function lets you access the current interpreter call stack, which is a list describing the sequence of routine calls that have brought IDL execution to the current point. You can use this new function to supplement your use of the other SCOPE functions.

In general, you should write routines that execute properly no matter how they are called or by which routines. As such, your programs should rarely require call-stack information. Code that looks at what routine called it and alters its behavior accordingly can quickly become difficult to understand and maintain, and is generally considered to represent poor programming style. For this reason, RSI recommends against the use of the SCOPE functions.

New Bit-Level Information Functions

Two new functions can give you bit-level information on an integer. The BIT_FFS function returns the index of the first bit set (non-zero) in its integer argument. The BIT_POPULATION function returns the number of set (non-zero) bits in its integer argument.

See BIT_FFS and BIT_POPULATION for details.

Enhancements to GET_KBRD

The ESCAPE keyword to the GET_KBRD routine lets you retrieve the escape sequence generated by a key for which one is defined. The KEY_NAME keyword lets you retrieve the name of a named key (function or arrow keys, for example).

The *Wait* argument is no longer required. Omitting the *Wait* argument causes GET_KBRD to wait until there is a character in the terminal type ahead buffer, duplicating the behavior of setting the *Wait* argument to a non-zero value.

See GET_KBRD for details.

Enhancements to XDISPLAYFILE

The GROW_TO_SCREEN keyword to the XDISPLAYFILE routine allows you to specify that the displayed text window show as many lines of text as possible within the height of the computer screen. The RETURN_ID keyword allows you to retrieve the widget ID of the top level base widget in which the text is displayed.

See XDISPLAYFILE for details.

Enhancements to .RESET_SESSION

The .RESET_SESSION command now resets system variables (including key graphics variables), runs the startup file (if any) after the reset is complete, and resets the current direct graphics device and the state of the path cache (enabled or disabled).

See .RESET_SESSION for details.

File Access Enhancements

The following enhancements have been made to IDL's file-access capabilities in the IDL 6.2 release:

- "DICOM Network Services" on page 16
- "JPEG2000 Analysis Enhancements" on page 17
- "HDF5 Write Support and Library Upgrade" on page 17
- "DataMiner Driver Enhancements" on page 17
- "New QUERY_ASCII Function" on page 17
- "Ability to Navigate XML DOM Trees" on page 17
- "Ability to Validate XML Schemas" on page 18

DICOM Network Services

Building on the momentum of the last release, which added the ability to read and write DICOM files in IDL using the IDLffDicomEx object, IDL 6.2 introduces DICOM Network Services support. This provides an interface that lets you to query, retrieve and send DICOM files. The **DICOM Network Services** dialog also gives you the ability to create and modify Application Entities, manage the Storage SCP Service, and Echo a remote machine. See Chapter 2, "Using IDL DICOM Network Services" in the *Medical Imaging in IDL* manual for complete details.

DICOM Network Services is available as an optional, add-on module to IDL. This module also includes the DICOM Read/Write module.

Note

DICOM Network Services requires an additional-cost license key to access the functionality.

For current conformance statement information, visit www.rsinc.com/idl/dicom.

Note

If a DICOM Read/Write license or runtime Read/Write license is available, QUERY_DICOM and READ_DICOM will use the IDLffDicomEx object to query and read DICOM files by default. Prior releases used the IDLffDicom object. This behavior of prior releases can be duplicated by setting the new DICOMEX keyword to zero on QUERY_DICOM or READ_DICOM.

JPEG2000 Analysis Enhancements

IDL 6.2 includes enhanced support for analysis of JPEG2000 files. These enhancements allow for the control of compression bit-rate and the addition of UUID boxes to store application specific data. New BIT_RATE and UUIDS properties have been added to the IDLffJPEG2000 object as well as new GetUUID and SetUUID object methods. For further information on these, see “IDLffJPEG2000” in the *IDL Reference Guide* manual.

HDF5 Write Support and Library Upgrade

Support for writing Hierarchical Data Format version 5 (HDF5) format files — first introduced as a plug-in release to IDL version 6.1, has been fully incorporated into IDL 6.2. See Chapter 3, “Hierarchical Data Format - HDF5” in the *IDL Scientific Data Formats* manual for additional information.

In addition, IDL now uses the HDF5 library version 5-1.6.3.

DataMiner Driver Enhancements

The IDL DataMiner now uses version 5.0 of the DataDirect Connect ODBC driver set for all supported platforms except SGI IRIX. See “About the DataMiner ODBC Drivers” in Chapter 1 of the *DataMiner Guide* manual for details.

New QUERY_ASCII Function

The QUERY_ASCII function tests a file for compatibility with READ_ASCII and returns an optional structure containing information about the file.

See QUERY_ASCII for details.

Ability to Navigate XML DOM Trees

Two new object classes let you navigate an XML DOM tree:

- IDLffXMLDOMNodeIterator (using iterative linear walkthrough)
- IDLffXMLDOMTreeWalker (using tree-wise walkthrough)

The IDLffXMLDOMDocument class has two new corresponding methods for creating instances of these new objects:

- IDLffXMLDOMDocument::CreateNodeIterator
- IDLffXMLDOMDocument::CreateTreeWalker

Ability to Validate XML Schemas

You can now indicate the type of validation the parser should perform on your XML documents when you are using IDL's XML DOM functionality. XML schemas describe the structure and allowed contents of an XML document. Schemas are more robust than, and are envisioned as a replacement for, DTDs.

For more information, see the SCHEMA_CHECKING keyword to IDLffXMLDOMDocument::Init or IDLffXMLDOMDocument::Load in the *IDL Reference Guide*.

IDLDE Enhancements

The following enhancement has been made to the IDL Development Environment in the IDL 6.2 release:

- "Persistent Command Recall Buffer" on page 19

Persistent Command Recall Buffer

The IDLDE command recall buffer now persists between sessions. You can access commands in the buffer by using the keyboard's directional arrows. You can use the IDL preference system to turn the buffer's persistence on or off (persistence is on by default) and set the size of the buffer. For details on these preferences, see IDL_RBUF_PERSIST and IDL_RBUF_SIZE in Appendix E, "IDL Preferences" in the *IDL Reference Guide* manual.

User Interface Toolkit Enhancements

The following enhancements have been made to the IDL's graphical user interface toolkit in the IDL 6.2 release:

- "Enhancements to Table Widget Cells" on page 20
- "Mouse-Wheel Events in Draw Widgets" on page 21

Enhancements to Table Widget Cells

The table widget now supports a variety of cell attributes that you can apply either to the entire table or to a subset of its cells. You can set them at table creation with the WIDGET_TABLE function and change them after creation with the WIDGET_CONTROL procedure. You can also query for some of them with the WIDGET_INFO function. The following table describes the table cell attributes.

Attribute	Description
ALIGNMENT	Horizontal alignment of text within a cell (left, middle, right)
BACKGROUND_COLOR	Color of the background of a cell
EDITABLE	Indication of whether you can edit a cell
FONT	Font to use when drawing a cell's text
FOREGROUND_COLOR	Color of the foreground of a cell
FORMAT	Formatting string to use when drawing a cell's value

Table 1-1: Table Cell Attributes

You can also add and delete cells by any of a number of methods. One is to use the WIDGET_CONTROL procedure's row and column insertion and deletion keywords. Another is to change the table's XSIZE or YSIZE attributes. You can also make the change by setting the table's value.

For more information, see "Cell Attributes" in Chapter 30 of the *Building IDL Applications* manual.

Mouse-Wheel Events in Draw Widgets

Under Microsoft Windows, you can now capture and get information about mouse-wheel events in draw widgets.

For more information, see “Button, Motion, Wheel, and Keyboard Events” 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 6.2 release:

- "New Cross-Platform HTML Help System" on page 22

New Cross-Platform HTML Help System

IDL 6.2 provides a new, cross-platform online help system using a help viewer — *IDL Assistant* — based on the help viewer used by the Qt development toolkit from Trolltech. The new online help system has the following advantages:

- IDL now uses the same help viewer on all platforms.
- The IDL Assistant help viewer is fully integrated with IDL, providing full help (?) functionality on all platforms.
- The entire IDL documentation set is now included in the online help system. Users can perform full-text searches for words and phrases appearing anywhere in the documentation set.
- IDL users can create their own online help systems using the IDL Assistant viewer. Help files are in HTML format.

Documentation for using the IDL Assistant help viewer is available in "Getting Help with IDL" in Chapter 1 of the *Using IDL* manual. For information on creating help content that uses IDL Assistant for your own IDL applications, see Chapter 23, "Providing Online Help For Your Application" in the *Building IDL Applications* manual.

Note

The first time IDL Assistant runs, it creates an index of the full documentation set to aid in searching. This indexing step may take several minutes. After the index has been created, IDL Assistant will launch more quickly.

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.

**BIT_FFS — ** The BIT_FFS routine returns the index to the first bit set (non-zero) in an integer.

**BIT_POPULATION — ** The BIT_POPULATION routine returns the number of set (non-zero) bits in an integer.

**CMYK_CONVERT — ** The CMYK_CONVERT procedure converts from the CMYK (cyan-magenta-yellow-black) color model to RGB (red-green-blue) and vice versa.

**COMMAND_LINE_ARGS — ** The COMMAND_LINE_ARGS function returns string values supplied when the user starts IDL with the `-arg` or `-args` command line options.

DICOMEX_GETCONFIGFILEPATH — ** The DICOMEX_GETCONFIGFILEPATH function returns the location of the local or system configuration file associated with Application Entities defined in the **Dicom Network Services utility.

**DICOMEX_GETSTORSCPDIR — ** The DICOMEX_GETSTORSCPDIR function returns the location of the directory associated with the Storage SCP Service.

DICOMEX_NET — ** The DICOMEX_NET procedure launches the **Dicom Network Services utility described in "DICOM Network Services" on page 16.

**FILE_POLL_INPUT — ** The FILE_POLL_INPUT function blocks processing until it detects that a read operation on a specified file will succeed.

**HEAP_NOSAVE — ** The HEAP_NOSAVE procedure is used to clear the *save attribute* of pointer or object heap variables.

**HEAP_SAVE — ** The HEAP_SAVE function is used to query whether a pointer or object heap variable can be saved. It can also be used to change the heap variable save attribute.

**PREF_COMMIT — ** The PREF_COMMIT procedure commits the pending changes to preference values.

**PREF_GET — ** The PREF_GET function returns information about IDL preferences.

**PREF_MIGRATE — ** The PREF_MIGRATE procedure is used to import IDL user preferences from other versions of IDL for use by the currently running version.

**PREF_SET — ** The PREF_SET procedure is used to set new values for IDL preferences.

QUERY_ASCII — The QUERY_ASCII function tests a file for compatibility with READ_ASCII and returns an optional structure containing information about the file.

SCOPE_TRACEBACK — The SCOPE_TRACEBACK function is used to obtain an array containing the current interpreter call stack.

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.

CONVOL — The CONVOL function has the following new keywords:

- BIAS sets the bias offset to be added to each result value, after any scale factor has been applied.
- EDGE_ZERO forces the function to compute the values of elements at the edge of the input array as if the array were padded with zeroes.
- INVALID sets the value that the function should use to indicate missing or invalid data within the input array.
- NORMALIZE forces the function to automatically compute a scale factor and bias and apply them to the result values.

DIALOG_MESSAGE — The DIALOG_MESSAGE function has the following new keyword:

- CENTER centers the dialog on the screen (ignored under Microsoft Windows).

GET_KBRD — The GET_KBRD function has the following new keywords:

- ESCAPE causes the function to return a complete escape sequence when the user presses a key that generates one.
- KEY_NAME causes the function to return the name of the key pressed by the user (function or arrow keys, for example).

In addition, omitting the Wait argument is now equivalent to setting it to a non-zero value.

HELP — The HELP procedure has the following new keyword:

- PREFERENCES displays information about the current status of IDL's preferences.

ICONTOUR — The ICONTOUR procedure has the following new keywords:

- DISABLE_SPLASH_SCREEN lets you disable the display of the iTools splash screen.
- VIEW_TITLE allows you to specify a text annotation that will be placed in the current view when creating the iTool.

IDLITSYS_CREATETOOL — The IDLITSYS_CREATETOOL function has the following new keywords:

- DISABLE_SPLASH_SCREEN lets you disable the display of the iTools splash screen.
- USER_INTERFACE lets you select a previously registered custom user interface for a new iTool.

IIMAGE — The IIMAGE procedure has the following new keywords:

- DISABLE_SPLASH_SCREEN lets you disable the display of the iTools splash screen.
- VIEW_TITLE allows you to specify a text annotation that will be placed in the current view when creating the iTool.

IMAP — The IMAP procedure has the following new keywords:

- DISABLE_SPLASH_SCREEN lets you disable the display of the iTools splash screen.
- VIEW_TITLE allows you to specify a text annotation that will be placed in the current view when creating the iTool.

IPLOT — The IPLOT procedure has the following new keywords:

- DISABLE_SPLASH_SCREEN lets you disable the display of the iTools splash screen.
- VIEW_TITLE allows you to specify a text annotation that will be placed in the current view when creating the iTool.

ISURFACE — The ISURFACE procedure has the following new keywords:

- DISABLE_SPLASH_SCREEN lets you disable the display of the iTools splash screen.
- VIEW_TITLE allows you to specify a text annotation that will be placed in the current view when creating the iTool.

IVOLUME — The IVOLUME procedure has the following new keywords:

- DISABLE_SPLASH_SCREEN lets you disable the display of the iTools splash screen.
- VIEW_TITLE allows you to specify a text annotation that will be placed in the current view when creating the iTool.

MAP_PROJ_FORWARD — The MAP_PROJ_FORWARD function has the following new keywords:

- Thread pool keywords make use of IDL's *thread pool*, which can increase execution speed on systems with multiple CPUs.

MAP_PROJ_IMAGE — The MAP_PROJ_IMAGE function has the following new keywords:

- XINDEX contains the *x* index used to warp the image when warping multiple images (or image channels) that have the same dimensions and map projection.
- YINDEX contains the *y* index used to warp the image when warping multiple images (or image channels) that have the same dimensions and map projection.

MAP_PROJ_INVERSE — The MAP_PROJ_INVERSE function has the following new keywords:

- Thread pool keywords make use of IDL's *thread pool*, which can increase execution speed on systems with multiple CPUs.

POLYFILLV — The POLYFILLV function has the following enhancement:

- The subscripts that POLYFILLV returns will be LONG64 values if the dimensions of the input array ($S_x * S_y$) warrant that size for proper addressing.

QUERY_DICOM — The QUERY_DICOM function has the following new keyword:

- DICOMEX specifies whether to use the IDLffDicomEx object or IDLffDICOM object to query a DICOM file. Unless set to zero, the IDLffDicomEx object will be used by default whenever a license is available.

READ_DICOM — The READ_DICOM routine has the following new keyword:

- DICOMEX specifies whether to use the IDLffDicomEx object or IDLffDICOM object to query a DICOM file. Unless set to zero, the IDLffDicomEx object will be used by default whenever a license is available.

SAVE — The SAVE routine has the following new keyword:

- EMBEDDED instructs IDL to create a SAVE file with an embedded license if the current IDL session has access to an IDL Developer's Kit license. This is equivalent to selecting the **Licensed Save File (.sav)** option in the **Project Options** dialog.

WIDGET_CONTROL — The WIDGET_CONTROL procedure has the following new keywords:

- BACKGROUND_COLOR specifies the background color of table cells.
- DRAW_WHEEL_EVENTS returns information about the settings of a created draw widget.
- FONT specifies the font of table cells.
- FOREGROUND_COLOR specifies the foreground (text) color of table cells.
- IGNORE_ACCELERATORS specifies the accelerators that should be ignored while the target widget has the keyboard focus.

The EDITABLE, ROW_HEIGHTS, SET_TABLE_SELECT, and USE_TABLE_SELECT keywords have also been enhanced to take advantage of the new cell-attributes functionality of table widgets.

WIDGET_DRAW — The WIDGET_DRAW procedure has the following new keyword:

- WHEEL_EVENTS enable mouse scroll wheel events.

WIDGET_INFO — The WIDGET_INFO function has the following new keywords:

- DRAW_WHEEL_EVENTS retrieves whether mouse wheel events are enabled or not.
- STRING_SIZE retrieves the dimensions of a string.
- TABLE_BACKGROUND_COLOR returns the background colors of table cells.
- TABLE_FONT returns the font names of table cells.
- TABLE_FOREGROUND_COLOR returns the text colors of table cells.

The ROW_HEIGHTS, TABLE_EDITABLE, and USE_TABLE_SELECT keywords have also been enhanced to take advantage of the new cell-attributes functionality of table widgets.

WIDGET_TABLE — The WIDGET_TABLE function has the following new keywords:

- BACKGROUND_COLOR specifies the cell background color.
- EDITABLE allows or denies direct user editing of table cells.
- FONT specifies the names of the fonts used by the widget.

- FOREGROUND_COLOR specifies the cell foreground color.

The ROW_HEIGHTS keyword has also been enhanced to take advantage of the new cell-attributes functionality of table widgets.

XDISPLAYFILE — The XDISPLAYFILE procedure has the following new keywords:

- GROW_TO_SCREEN causes the display window to show as much of the text file as possible within the constraints of the user's computer screen.
- RETURN_ID allows you to retrieve the widget ID of the base widget in which the text is displayed.

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.

IDLffXMLDOMNodeIterator — A class that lets you navigate an XML DOM tree in iterative linear fashion.

IDLffXMLDOMTreeWalker — A class that lets you navigate an XML DOM tree in tree-walking fashion.

New IDL Object Properties

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

IDLffJPEG2000 — The IDLffJPEG2000 object includes the following new properties:

- BIT_RATE contains the bit rate for each layer, measured in bits per pixel per component (or band).
- UUIDS returns provides a mechanism to retrieve the UUIDs for the UUID boxes in the JPEG2000 file.

IDLffXMLDOMDocument — The IDLffXMLDOMDocument object includes the following new property:

- NODE_DESTRUCTION_POLICY specifies the node-destruction policy for all node objects created directly or indirectly by a document object.

IDLgrImage — The IDLgrImage object includes the following new properties:

- ALPHA_CHANNEL specifies the opacity of an image.
- DEPTH_OFFSET specifies an offset in depth to be used when rendering images.
- RENDER_METHOD controls whether an image is rendered with a texture map or 2D pixel primitive.
- TILE_COLOR sets the default background color displayed when no tile data is visible.
- TILE_CURRENT_LEVEL specifies the level (resolution) of data that will be requested by a destination object that supports tiling.
- TILE_DIMENSIONS defines the width and height of the tiles in a tiled image.
- TILE_LEVEL_MODE defines whether the level of tiled data requested is limited to a single level (manual), or if IDL automatically calculates the level based on zoom level (automatic).
- TILE_NUM_LEVELS automatically calculates the number of image levels used in tiling based on image size and tile size. This is only in effect when TILE_LEVEL_MODE is set to automatic.
- TILED_IMAGE_DIMENSIONS defines the size of the original image that will be displayed as a tiled image.

- TILING defines an image object as one that supports tiling.
- TRANSFORM_MODE determines how an image renders with rotation transforms.

IDLgrModel — The IDLgrModel object includes the following new properties:

- ACTIVE_POSITION is a scalar integer property that is a zero-based index into the model container. It indicates the container index of the object that is drawn by this model when the value of the RENDER_METHOD property is 1.
- RENDER_METHOD is a scalar integer property that specifies how this model object renders the objects in its container.

IDLgrWindow — The IDLgrWindow object includes the following new property:

- VIEWPORT_DIMENSIONS is a two-element floating-point vector of the form [*width*, *height*] specifying the dimensions of the *viewport* of the draw widget that contains the window. If the window is not within a draw widget, this property is equivalent to the DIMENSIONS property.

IDLitCommand — The IDLitCommand object includes the following new properties:

- SKIP_REDO indicates whether the command object should be included in the iTools Redo buffer.
- SKIP_UNDO indicates whether the command object should be included in the iTools Undo buffer.

IDLitDataOperation — The IDLitDataOperation object includes the following new property:

- WITHIN_UI indicates whether the operation is currently displaying a dialog.

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.

IDLffJPEG2000::GetUUID — This new method allows you to get the data field from the specified UUID box.

IDLffJPEG2000::SetUUID — This new method allows you to add UUID boxes when creating a new JPEG2000 file.

IDLffXMLDOMDocument::CreateNodeIterator — This new method creates an instance of the IDLffXMLDOMNodeIterator object, which lets you navigate an XML DOM tree in iterative (linear) fashion.

IDLffXMLDOMDocument::CreateTreeWalker — This new method creates an instance of the IDLffXMLDOMTreeWalker object, which lets you navigate an XML DOM tree in tree-walking fashion.

IDLgrBuffer::QueryRequiredTiles — This new method determines what tiles are visible in a view and returns an array of structures that describe these tiles. After extracting the data associated with this information, pass the result to IDLgrImage::SetTileData.

IDLgrClipboard::QueryRequiredTiles — This new method determines what tiles are visible in a view and returns an array of structures that describe these tiles. After extracting the data associated with this information, pass the result to IDLgrImage::SetTileData.

IDLgrImage::DeleteTileData — This new method removes tile data from the image object. You can remove tiles identified by QueryRequiredTiles as being visible in the view, or remove all tile data.

IDLgrImage::SetTileData — This new method assigns tile data to the image object.

IDLgrPrinter::QueryRequiredTiles — This new method determines what tiles are visible in a view and returns an array of structures that describe these tiles. After extracting the data associated with this information, pass the result to IDLgrImage::SetTileData.

IDLgrWindow::QueryRequiredTiles — This new method determines what tiles are visible in a view and returns an array of structures that describe these tiles. After extracting the data associated with this information, pass the result to IDLgrImage::SetTileData.

IDLitOperation::QueryAvailability — This new method determines whether the operation is applicable for the selected data and/or visualization.

IDLitWindow::OnTimer — This new method handles notification (from the native window device) that a timer event has occurred, and passes that notification to all observers in the list of window event observers by calling each observer's OnTimer method.

IDLitWindow::SetTimerInterval — This new method specifies the floating-point number of seconds between timer events.

IDL Object Method Enhancements

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

IDLffXMLDOMDocument::Init — This method features the following new keyword:

- SCHEMA_CHECKING is an integer value used to indicate the type of validation the parser should perform.

IDLffXMLDOMDocument::Load — This method features the following new keyword:

- SCHEMA_CHECKING is an integer value used to indicate the type of validation the parser should perform.

IDLffXMLDOMDocument::Save — This method features the following new keyword:

- ENCODING lets you specify the encoding to be used when the XML DOM document is written to a file.

IDLgrBuffer::GetDeviceInfo — This method features the following new keyword:

- MAX_TILE_DIMENSIONS returns the maximum tile size supported by the destination object.

IDLgrClipboard::GetDeviceInfo — This method features the following new keyword:

- MAX_TILE_DIMENSIONS returns the maximum tile size supported by the destination object.

IDLgrWindow::GetDeviceInfo — This method features the following new keyword:

- MAX_TILE_DIMENSIONS returns the maximum tile size supported by the destination object.

IDLitContainer::AddByIdentifier — This method features the following new keyword:

- FOLDER_CLASSNAME allows you to specify the name of an iTool object class to be used when creating folders in an iTools container object hierarchy.

IDLitReader::Init — This method's *Extensions* argument is now optional.

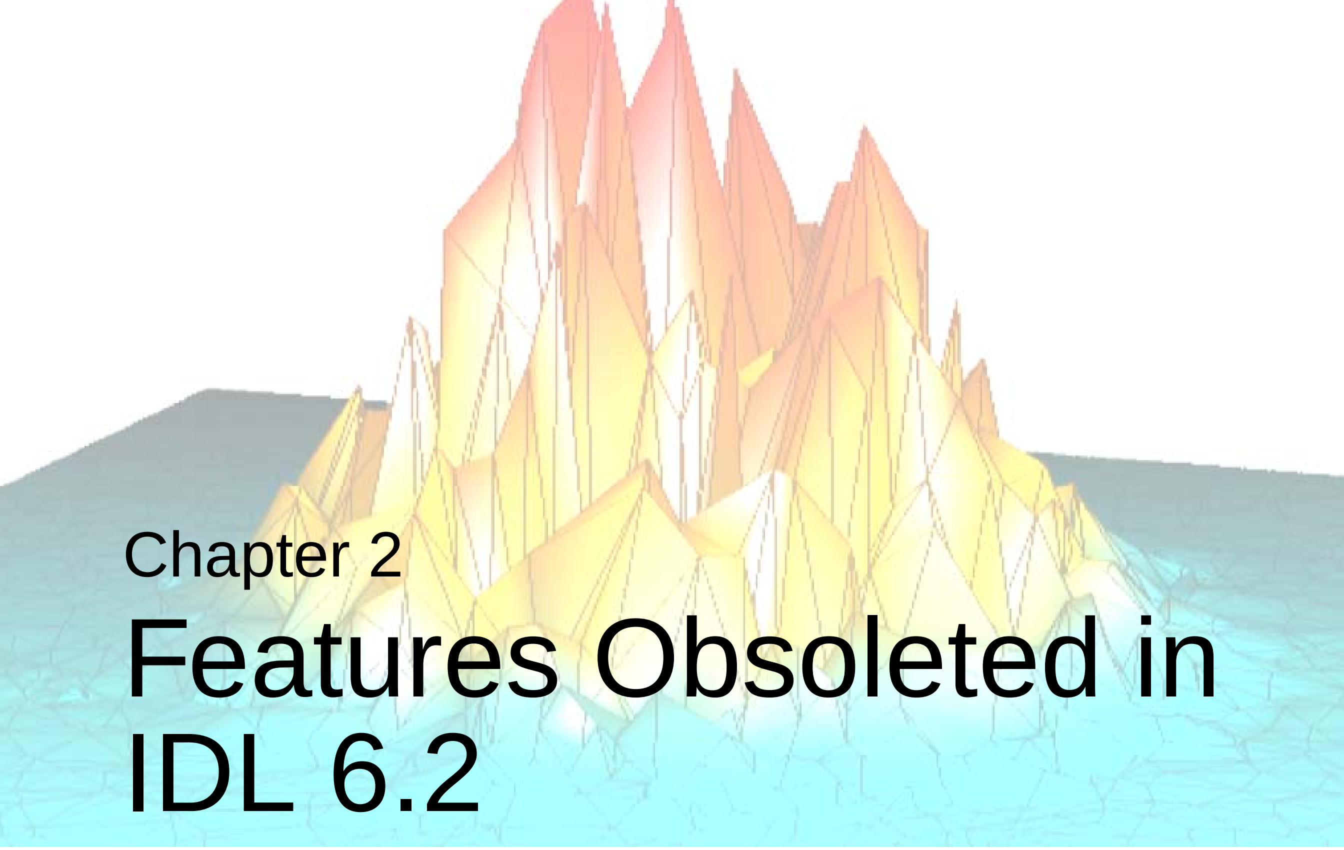
IDLitWindow::GetEventMask — This method features the following new keyword:

- TIMER_EVENTS returns a 1 if timer events are currently enabled for the window, or a 0 otherwise.

IDLitWindow::SetEventMask — This method features the following new keyword:

- TIMER_EVENTS indicates that timer events are to be enabled for this window.

IDLitWriter::Init — This method's *Extensions* argument is now optional.



Chapter 2

Features Obsoleted in IDL 6.2

The following features were present in IDL Version 6.1 but became obsolete in Version 6.2. Obsoleted features should not be used in new IDL code.

Obsolete Routines 38
Obsolete Arguments or Keywords 39

Obsolete Routines

The following routines have been replaced with new functionality:

Routine	Replaced By
ONLINE_HELP_PDF_INDEX	Nothing: the IDL-Acrobat plugin is no longer supported
SETUP_KEYS	Nothing; no longer needed

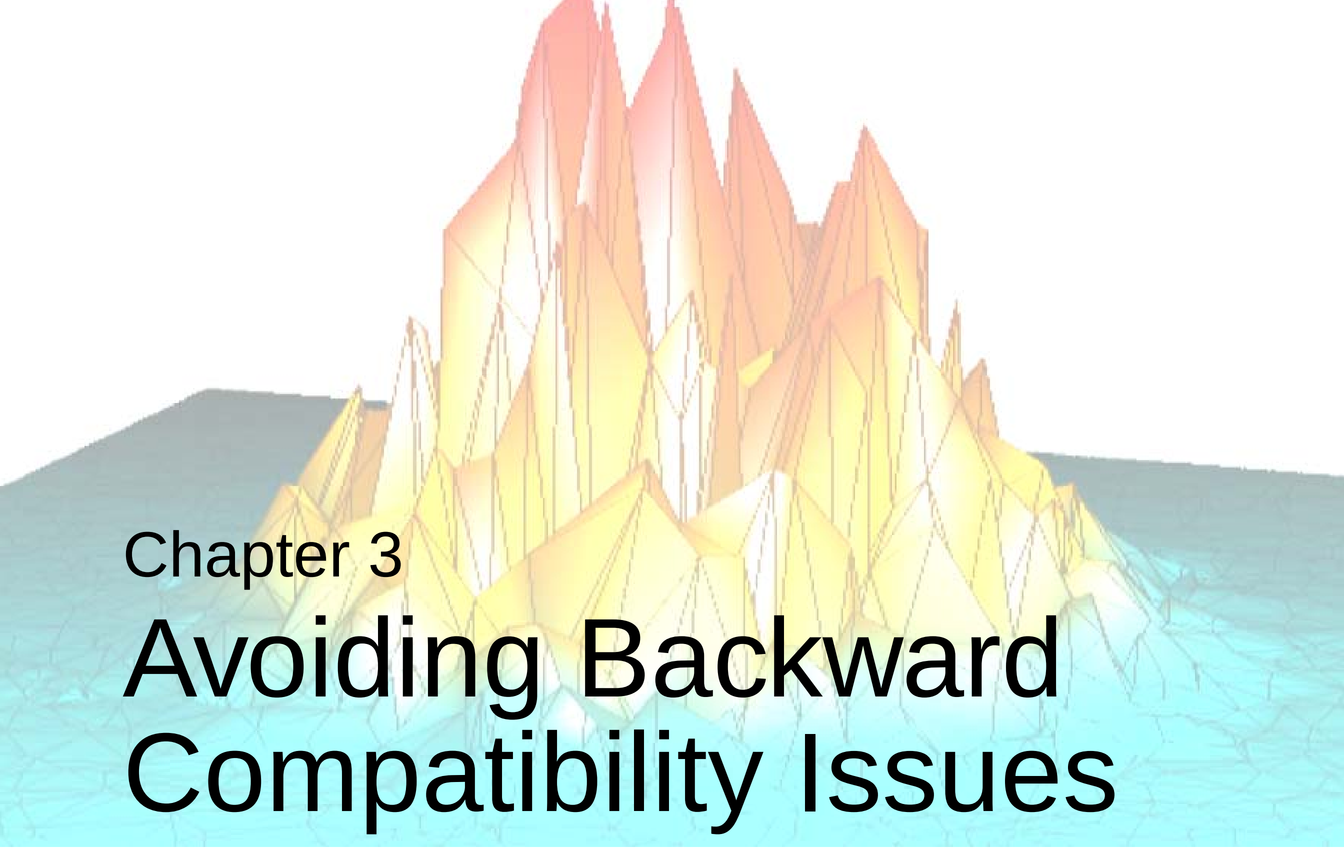
Table 2-1: Obsolete Routines

Obsolete Arguments or Keywords

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

Routine	Argument or Keyword
HELP	ALL_KEYS keyword
	CALLS keyword
IDLitTool::RegisterOperation	DISABLE keyword
ONLINE_HELP	PAGE keyword
	SUPPRESS_PLUGIN_ERRORS keyword
	TOPICS keyword
WIDGET_CONTROL	CANCEL_BUTTON keyword
	DEFAULT_BUTTON keyword

Table 2-2: Obsolete Keywords



Chapter 3

Avoiding Backward Compatibility Issues

Although RSI strives to maintain backward compatibility with previous versions, some enhancements can require changes to your code.

Visual Class Selection in Direct Graphics . . 42

Visual Class Selection in Direct Graphics

The order in which IDL selects X Window System visuals has changed. When opening the display, IDL asks the display for the following visuals, in order, until a supported visual class is found. In IDL 6.2, the visual class selection order has changed to:

1. TrueColor, 24-bit, then 16-bit, then 15-bit
2. PseudoColor, 8-bit, then 4-bit
3. DirectColor, 24-bit
4. StaticColor, 8-bit, then 4-bit
5. GrayScale, any depth
6. StaticGray, any depth

Previously, DirectColor appeared first in the list.

Background Information

The X Window system has 6 different possible visual classes, many of which can be provided at varying depths. It is common for high quality X servers to support more than one such visual class, and to support them at multiple depths. As such, graphically intensive applications like IDL must query the server to determine which visuals are available, and choose one. This decision is complicated by the fact that different visuals have different abilities that may or may not be important to different IDL programs. For example, the PseudoColor visual uses a hardware colormap with a limited number of simultaneous colors, whereas the TrueColor visual offers a very large number of simultaneous colors, but no colormap.

IDL makes the decision about which visual to use in the following way:

1. If a visual is explicitly specified via the IDL preference system or via the DEVICE command, and the X server supports that visual, then that visual is used.
2. If no visual is specified, IDL queries the server and chooses the “best” visual in the order given in the previous list.

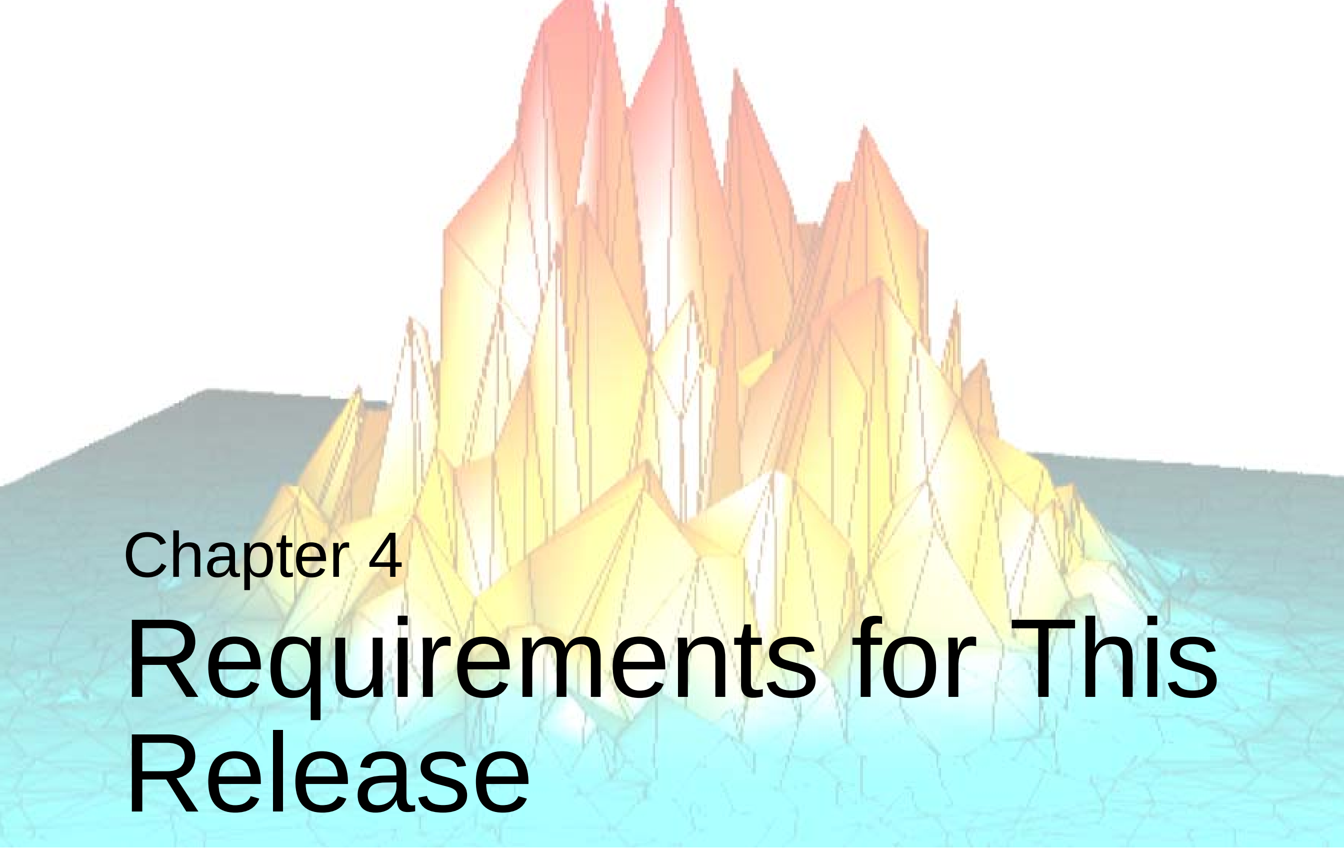
The original order of the default list gave higher priority to more sophisticated visuals. DirectColor is the most powerful visual, and as such was placed at the top. It must be noted that at the time the original list was made DirectColor was very rare and therefore rarely selected. Over the years since that original list was made, we have observed the following:

- IDL Programs that require DirectColor almost always ask for it explicitly.
- Almost all IDL programs that require 24-bit color abilities are written to use the TrueColor visual. It is rare for 24-bit programs to need the added abilities of DirectColor. The added abilities of the DirectColor visual relative to the TrueColor visual are not important to the vast majority of IDL users.
- DirectColor visuals are becoming common, largely because the X servers strive to offer every possible option, and the hardware is becoming more capable. This means that IDL is increasingly likely to choose it.
- When a DirectColor visual is used by IDL on modern X desktop systems, hardware colormap flashing almost always results.

As a result, IDL has become increasingly likely to select a default visual (DirectColor) that has features the user does not need, and that causes a side effect (colormap flashing) that is widely disliked. Moving the priority of DirectColor down therefore allows IDL to pick a suitable default for most users.

This change does not affect you if you specify a visual class explicitly. If you or your application relies on IDL selecting the DirectColor visual class by default, then you will need to explicitly request this visual class by either using the IDL preference system or by issuing the following at the command line:

```
DEVICE, DIRECT_COLOR=24
```



Chapter 4

Requirements for This Release

This chapter contains the following topics:

IDL 6.2 Requirements 46
ION 6.2 Requirements 48
IDL-Java Bridge Requirements 50

IDL 6.2 Requirements

Hardware and Operating System Requirements for IDL 6.2

The following table describes the supported platforms and operating systems for IDL 6.2:

Platform	Vendor	Hardware	Operating System	Supported Versions
Windows	Microsoft	Intel x86 32-bit	Windows	2000, XP
Macintosh[a,b,c]	Apple	PowerMac G4, G5	OS X	10.3, 10.4
UNIX [c]	HP [e]	PA-RISC 32-bit	HP-UX	11.0
	HP [e]	PA-RISC 64-bit[a,b]	HP-UX	11.0
	IBM [e]	RS/6000 32-bit	AIX	5.1
	IBM [e]	RS/6000 64-bit[a,b]	AIX	5.1
	Intel	Intel x86 32-bit	Linux[d]	Red Hat Enterprise 3.x, Fedora Core 3[b]
	Intel/AMD	x86_64 64-bit[a,b,e]	Linux[d]	Red Hat Enterprise 3.x
	SGI [e]	Mips 32-bit	IRIX	6.5.1
	SGI [e]	Mips 64-bit[a,b]	IRIX	6.5.1
	SUN	SPARC 32-bit	Solaris	8, 9
	SUN	SPARC 64-bit[a,b,e]	Solaris	8, 9

Table 4-1: Hardware Requirements for IDL 6.2

On 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
```

[a] The DXF file format is not supported on Mac OS X or 64-bit platforms.

[b] The IDL DataMiner is not supported on Mac OS X, Fedora Core 3, or 64-bit platforms.

[c] 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.

[d] IDL 6.2 was built on the Linux 2.4 kernel with `glibc` 2.3.2 using Red Hat Linux. If your version of Linux is compatible with these, it is possible that you can install and run IDL on your version.

[e] IDLffDicomEx and IDL DICOM Network Services are not supported on HP, IBM, SGI, or any 64-bit platforms.

Software Requirements for IDL 6.2

The following table describes the software requirements for IDL 6.2:

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

Table 4-2: Software Requirements for IDL 6.2

ION 6.2 Requirements

Hardware Requirements for ION 6.2

The following table describes the supported platforms and operating systems for ION 6.2:

Platform	Vendor	Hardware	Operating System	Supported Versions
Windows	Microsoft	Intel x86 32-bit	Windows	2000, XP
UNIX[a]	Intel	Intel x86 32-bit	Linux[b]	Red Hat Enterprise 3.x, Fedora Core 3
	SGI	Mips 32-bit	IRIX	6.5.1
	SUN	SPARC 32-bit	Solaris	8, 9

Table 4-3: Hardware Requirements for ION 6.2

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

[b] ION 6.2 was built on the Linux 2.4 kernel with `glibc` 2.3.2 using Red Hat Linux. If your version of Linux is compatible with these, it is possible that you can install and run ION 6.2 on your version.

Web Server Requirements for ION 6.2

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. This version is 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.

Note

For more information on Apache software for your platform, see http://www.apache.org.

Web Browser Requirements for ION 6.2

ION 6.2 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 6.2

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

- Mozilla Firefox version 1.0 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. It is recommended that Sun's JVM be installed and used with Internet Explorer because Microsoft's JVM does not support current Java features.

IDL-Java Bridge Requirements

IDL now supports the use of Java objects. You can access Java objects within your IDL code using the IDL-Java bridge, a built-in feature of IDL 6.2. The IDL-Java bridge enables you to take advantage of special Java I/O, networking, and third party functionality.

The IDL-Java bridge is installed by default in a standard IDL installation. See Chapter 8, “Using Java Objects in IDL” in the *External Development Guide* manual for details.

Hardware Requirements for the IDL-Java Bridge

The following table describes the platforms and operating systems for the IDL-Java bridge:

Platform	Vendor	Hardware	Operating System	Supported Versions
Windows	Microsoft	Intel x86 32-bit	Windows	2000, XP
Macintosh[a]	Apple	PowerMac G4	OS X	10.3, 10.4
UNIX[a]	Intel	Intel x86 32-bit	Linux[b]	Red Hat Enterprise 3.x, Fedora Core 3
	SGI	Mips 32-bit	IRIX	6.5.1
	SUN	SPARC 32-bit	Solaris	8, 9
	SUN	SPARC 64-bit	Solaris	8, 9

Table 4-4: Hardware Requirements for the IDL-Java Bridge

On 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
```

[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 6.2 was built on the Linux 2.4 kernel with `glibc` 2.3.2 using Red Hat Linux. If your version of Linux is compatible with these, it is possible that you can install and run IDL on your version.

Java Virtual Machine Requirements for the IDL-Java Bridge

IDL supports version 1.3.1 and greater on all platforms with the following exceptions:

- The supported version on Macintosh is 1.3.x
- SUN SPARC 64-bit supports only version 1.4.x and greater