Restricted Rights Notice

The IDL®, ION Script™, and ION Java™ software programs and the accompanying procedures, functions, and documentation described herein are sold under license agreement. Their use, duplication, and disclosure are subject to the restrictions stated in the license agreement. Research Systems, Inc., reserves the right to make changes to this document at any time and without notice.

Limitation of Warranty

Research Systems, Inc. makes no warranties, either express or implied, as to any matter not expressly set forth in the license agreement, including without limitation the condition of the software, merchantability, or fitness for any particular purpose.

Research Systems, Inc. shall not be liable for any direct, consequential, or other damages suffered by the Licensee or any others resulting from use of the IDL or ION software packages or their documentation.

Permission to Reproduce this Manual

If you are a licensed user of this product, Research Systems, Inc. grants you a limited, nontransferable license to reproduce this particular document provided such copies are for your use only and are not sold or distributed to third parties. All such copies must contain the title page and this notice page in their entirety.

Acknowledgments

IDL® is a registered trademark and ION™, ION Script™, ION Java™, are trademarks of Research Systems Inc., 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 program are copyright © 1995-1999 LizardTech, Inc. All rights reserved. MsSID is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

Portions of this software are copyrighted by Merge Technologies Incorporated.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/)

IDL Wavelet Toolkit Copyright © 2002 Christopher Torrence.

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

Chapter 1: Functional List of IDL Routines ................................................................. 7

3D Visualization ........................................................................................................ 8
Animation .................................................................................................................... 9
Array Creation .......................................................................................................... 9
Array Manipulation ................................................................................................. 9
Color Table Manipulation ....................................................................................... 10
Date and Time ......................................................................................................... 10
Debugging .............................................................................................................. 10
Dialog Routines ..................................................................................................... 10
Direct Graphics, General ....................................................................................... 10
Error Handling ...................................................................................................... 11
Executive Commands ......................................................................................... 11
External Linking ................................................................................................. 11
Font Manipulation ............................................................................................... 11
Help Routines ...................................................................................................... 11
Chapter 3:
Scientific Data Formats ................................................................. 123
# Functional List of IDL Routines

The following is a list of all routines included in IDL, categorized by functionality.

<table>
<thead>
<tr>
<th>Category</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Visualization</td>
<td>8</td>
</tr>
<tr>
<td>Animation</td>
<td>9</td>
</tr>
<tr>
<td>Array Creation</td>
<td>9</td>
</tr>
<tr>
<td>Array Manipulation</td>
<td>9</td>
</tr>
<tr>
<td>Color Table Manipulation</td>
<td>10</td>
</tr>
<tr>
<td>Date and Time</td>
<td>10</td>
</tr>
<tr>
<td>Debugging</td>
<td>10</td>
</tr>
<tr>
<td>Dialog Routines</td>
<td>10</td>
</tr>
<tr>
<td>Direct Graphics, General</td>
<td>10</td>
</tr>
<tr>
<td>Error Handling</td>
<td>11</td>
</tr>
<tr>
<td>Executive Commands</td>
<td>11</td>
</tr>
<tr>
<td>External Linking</td>
<td>11</td>
</tr>
<tr>
<td>Font Manipulation</td>
<td>11</td>
</tr>
<tr>
<td>Help Routines</td>
<td>11</td>
</tr>
<tr>
<td>Image Processing</td>
<td>11</td>
</tr>
<tr>
<td>Input/Output</td>
<td>13</td>
</tr>
<tr>
<td>Language Catalogs</td>
<td>14</td>
</tr>
<tr>
<td>Mapping</td>
<td>14</td>
</tr>
<tr>
<td>Mathematics</td>
<td>14</td>
</tr>
<tr>
<td>Object Class Library</td>
<td>18</td>
</tr>
<tr>
<td>Operating System Access</td>
<td>19</td>
</tr>
<tr>
<td>Performance Testing</td>
<td>20</td>
</tr>
<tr>
<td>Plotting</td>
<td>20</td>
</tr>
<tr>
<td>Programming and IDL Control</td>
<td>20</td>
</tr>
<tr>
<td>Query Routines</td>
<td>21</td>
</tr>
<tr>
<td>Saving/Restoring a Session</td>
<td>21</td>
</tr>
<tr>
<td>Scope Functions</td>
<td>22</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>22</td>
</tr>
<tr>
<td>String Processing</td>
<td>23</td>
</tr>
<tr>
<td>Structures</td>
<td>23</td>
</tr>
<tr>
<td>Type Conversion</td>
<td>23</td>
</tr>
<tr>
<td>Utilities</td>
<td>23</td>
</tr>
<tr>
<td>Wavelet Toolkit</td>
<td>23</td>
</tr>
<tr>
<td>Widget Routines</td>
<td>24</td>
</tr>
<tr>
<td>Widget Routines, Compound</td>
<td>24</td>
</tr>
<tr>
<td>Window Routines</td>
<td>24</td>
</tr>
</tbody>
</table>
3D Visualization

3D Transformations & Scene Setup

**CONVERT_COORD** - Transforms coordinates to and from the coordinate systems supported by IDL.

**COORD2TO3** - Returns 3D data coordinates given normalized screen coordinates.

**CREATE_VIEW** - Sets up 3D transformations.

**CV_COORD** - Converts 2D and 3D coordinates between coordinate systems.

**SCALE3** - Sets up axis ranges and viewing angles for 3D plots.

**SCALE3D** - Scales 3D unit cube into the viewing area.

**SET_SHADING** - Sets the light source shading parameters.

**SURFR** - Sets up 3D transformations by duplicating rotation, translation, and scaling of SURFACE.

**T3D** - Performs various 3D transformations.

**VERT_T3D** - Transforms a 3D array by a 4x4 transformation matrix.

**VOXEL_PROJ** - Generates volume visualizations using voxel technique.

Polygonal Mesh Routines

**COMPUTE_MESH_NORMALS** - Computes normal vectors for a set of polygons described by the input array.

**MESH_CLIP** - Clips a polygonal mesh to an arbitrary plane in space and returns a polygonal mesh of the remaining portion.

**MESH_DECIMATE** - Reduces the density of geometry while preserving as much of the original data as possible.

**MESH_ISSOLID** - Computes various mesh properties and enables IDL to determine if a mesh encloses space (is a solid).

**MESH_MERGE** - Merges two polygonal meshes.

**MESH_NUMTRIANGLES** - Computes the number of triangles in a polygonal mesh.

**MESH_OBJ** - Generates a polygon mesh for various simple objects.

**MESH_SMOOTH** - Performs spatial smoothing on a polygon mesh.

**MESH_SURFACEAREA** - Computes various mesh properties to determine the mesh surface area, including integration of other properties interpolated on the surface of the mesh.

**MESH_VALIDATE** - Checks for NaN values in vertices, removes unused vertices, and combines close vertices.

**MESH_VOLUME** - Computes the volume that the mesh encloses.

**POLYSHADE** - Creates a shaded surface representation from a set of polygons.

Surfaces and Contours

**CONTOUR** - Draws a contour plot.

**ICONTOUR** - Creates an iTool and associated user interface (UI) configured to display and manipulate contour data.

**IMAGE_CONT** - Overlays an image with a contour plot.

**ISURFACE** - Creates an iTool and associated user interface (UI) configured to display and manipulate surface data.

**MIN_CURVE_SURF** - Interpolates points with a minimum curvature surface or a thin-plate-spline surface. Useful with CONTOUR.

**POLAR_CONTOUR** - Draws a contour plot from data in polar coordinates.

**SHADE_SURF** - Creates a shaded-surface representation of gridded data.

**SHADE_SURF_IRR** - Creates a shaded-surface representation of an irregularly gridded dataset.

**SHOW3** - Displays array as image, surface plot, and contour plot simultaneously.

**SURFACE** - Plots an array as a wireframe mesh surface.

**XSURFACE** - Provides GUI to SURFACE and SHADE_SURF.

Tetrahedral Mesh Routines

**TETRA_CLIP** - Clips a tetrahedral mesh to an arbitrary plane in space and returns a tetrahedral mesh of the remaining portion.

**TETRA_SURFACE** - Extracts a polygonal mesh as the exterior surface of a tetrahedral mesh.

**TETRA_VOLUME** - Computes properties of tetrahedral mesh array.

Vector Field Visualization

**FLOW3** - Draws lines representing a 3D flow/velocity field.

**INTERPOL** - Performs linear interpolation on vectors.

**PARTICLE_TRACE** - Traces the path of a massless particle through a vector field.

**STREAMLINE** - Generates the visualization graphics from a path.

**VECTOR_FIELD** - Places colored, oriented vectors of specified length at each vertex in an input vertex array.

**VEL** - Draws a velocity (flow) field with streamlines.

**VELOVECT** - Draws a 2D velocity field plot.

Volume Visualization

**EXTRACT_SLICE** - Returns 2D planar slice extracted from volume.

**IDLgrVolume** - Represents a mapping from a 3D array of data to a 3D array of voxel colors, which, when drawn, are projected to two dimensions.

**INTERVAL_VOLUME** - Generates a tetrahedral mesh from volumetric data.

**ISOSURFACE** - Returns topologically consistent triangles by using oriented tetrahedral decomposition.

**IVOLUME** - Creates an iTool and associated user interface (UI) configured to display and manipulate volume data.

**PROJECT_VOL** - Returns a translucent rendering of a volume projected onto a plane.

**QGRID3** - Interpolates the dependent variable values to points in a regularly sampled volume.

**QHULL** - Constructs convex hulls, Delaunay triangulations, and Voronoi diagrams.

**RECON3** - Reconstructs a 3D representation of an object from 2D images.

**SEARCH3D** - Finds “objects” or regions of similar data values within a volume.
### Functional List of IDL Routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHADE_VOLUME</td>
<td>Contours a volume to create a list of vertices and polygons that can be displayed using POLYSHADE.</td>
</tr>
<tr>
<td>SLICER3</td>
<td>Interactive volume visualization tool.</td>
</tr>
<tr>
<td>VOXEL_PROJ</td>
<td>Generates volume visualizations using voxel technique.</td>
</tr>
<tr>
<td>XOBJVIEW</td>
<td>Displays object viewer widget.</td>
</tr>
<tr>
<td>XOBJVIEW_ROTATE</td>
<td>Programmatically rotate the object currently displayed in XOBJVIEW.</td>
</tr>
<tr>
<td>XOBJVIEW_WRITE_IMAGE</td>
<td>Write the object currently displayed in XOBJVIEW to an image file.</td>
</tr>
<tr>
<td>XVOLUME</td>
<td>Utility for viewing and interactively manipulating volumes and isosurfaces.</td>
</tr>
</tbody>
</table>

### Animation

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CW_ANIMATE</td>
<td>Creates a compound widget for animation.</td>
</tr>
<tr>
<td>CW_ANIMATE_GETP</td>
<td>Gets pixmap window IDs used by CW_ANIMATE.</td>
</tr>
<tr>
<td>CW_ANIMATE_LOAD</td>
<td>Loads images into CW_ANIMATE.</td>
</tr>
<tr>
<td>CW_ANIMATE_RUN</td>
<td>Displays images loaded into CW_ANIMATE.</td>
</tr>
<tr>
<td>FLICK</td>
<td>Causes the display to flicker between two images.</td>
</tr>
<tr>
<td>MPEG_CLOSE</td>
<td>Closes an MPEG sequence.</td>
</tr>
<tr>
<td>MPEG_OPEN</td>
<td>Opens an MPEG sequence.</td>
</tr>
<tr>
<td>MPEG_PUT</td>
<td>Inserts an image array into an MPEG sequence.</td>
</tr>
<tr>
<td>MPEG_SAVE</td>
<td>Saves an MPEG sequence to a file.</td>
</tr>
<tr>
<td>XINTERANIMATE</td>
<td>Displays animated sequence of images.</td>
</tr>
</tbody>
</table>

### Array Creation

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINDGEN</td>
<td>Returns byte array with each element set to its subscript.</td>
</tr>
<tr>
<td>BYTARR</td>
<td>Creates a byte vector or array.</td>
</tr>
<tr>
<td>CINDGEN</td>
<td>Returns a complex array with each element set to its subscript.</td>
</tr>
<tr>
<td>COMPLEXARR</td>
<td>Creates a complex, single-precision, floating-point vector or array.</td>
</tr>
<tr>
<td>DBLARR</td>
<td>Creates a double-precision array.</td>
</tr>
<tr>
<td>DCINDGEN</td>
<td>Returns a double-precision, complex array with each element set to its subscript.</td>
</tr>
<tr>
<td>DCOMPLEXARR</td>
<td>Creates a complex, double-precision vector or array.</td>
</tr>
<tr>
<td>DINDGEN</td>
<td>Returns a double-precision array with each element set to its subscript.</td>
</tr>
<tr>
<td>FINDGEN</td>
<td>Returns a floating-point array with each element set to its subscript.</td>
</tr>
<tr>
<td>FLTARR</td>
<td>Returns a single-precision, floating-point vector or array.</td>
</tr>
<tr>
<td>IDENTITY</td>
<td>Returns an identity array (an array with ones along the main diagonal and zeros elsewhere) of the specified dimensions.</td>
</tr>
<tr>
<td>INDGEN</td>
<td>Return an integer array with each element set to its subscript.</td>
</tr>
<tr>
<td>INTARR</td>
<td>Creates an integer vector or array.</td>
</tr>
<tr>
<td>L64INDGEN</td>
<td>Returns a 64-bit integer array with each element set to its subscript.</td>
</tr>
<tr>
<td>LINDGEN</td>
<td>Returns a longword integer array with each element set to its subscript.</td>
</tr>
<tr>
<td>LON64ARR</td>
<td>Returns a 64-bit integer vector or array.</td>
</tr>
<tr>
<td>LONARR</td>
<td>Returns a longword integer vector or array.</td>
</tr>
<tr>
<td>MAKE_ARRAY</td>
<td>Returns an array of the specified type, dimensions, and initialization.</td>
</tr>
<tr>
<td>OBJARR</td>
<td>Creates an array of object references.</td>
</tr>
<tr>
<td>PTRARR</td>
<td>Creates an array of pointers.</td>
</tr>
<tr>
<td>REPLICATE</td>
<td>Creates an array of given dimensions, filled with specified value.</td>
</tr>
<tr>
<td>SINDGEN</td>
<td>Returns a string array with each element set to its subscript.</td>
</tr>
<tr>
<td>STRARR</td>
<td>Returns string array containing zero-length strings.</td>
</tr>
<tr>
<td>TIMEGEN</td>
<td>Returns an array of double-precision floating-point values that represent times in Julian dates.</td>
</tr>
<tr>
<td>UINDGEN</td>
<td>Returns unsigned integer array with each element set to its subscript.</td>
</tr>
<tr>
<td>UINTARR</td>
<td>Returns an unsigned integer vector or array.</td>
</tr>
<tr>
<td>UL64INDGEN</td>
<td>Returns an unsigned 64-bit integer array with each element set to its subscript.</td>
</tr>
<tr>
<td>ULINDGEN</td>
<td>Returns an unsigned longword array with each element set to its subscript.</td>
</tr>
<tr>
<td>ULONG64ARR</td>
<td>Returns an unsigned 64-bit integer vector or array.</td>
</tr>
<tr>
<td>ULONGARR</td>
<td>Returns an unsigned longword integer vector or array.</td>
</tr>
</tbody>
</table>

### Array Manipulation

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY_EQUAL</td>
<td>Provides fast test for data equality in cases where the positions of the differing data elements is not required.</td>
</tr>
<tr>
<td>ARRAY_INDICES</td>
<td>Converts one-dimensional subscripts of an array into corresponding multi-dimensional subscripts.</td>
</tr>
<tr>
<td>BLAS_AXPY</td>
<td>Updates existing array by adding a multiple of another array.</td>
</tr>
<tr>
<td>INVERT</td>
<td>Computes the inverse of a square array.</td>
</tr>
<tr>
<td>MAX</td>
<td>Returns the value of the largest element of Array.</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>Returns the median value of Array or applies a median filter.</td>
</tr>
<tr>
<td>MIN</td>
<td>Returns the value of the smallest element of an array.</td>
</tr>
<tr>
<td>REFORM</td>
<td>Changes array dimensions without changing the total number of elements.</td>
</tr>
<tr>
<td>REPLICATE_INPLACE</td>
<td>Updates an array by replacing all or selected parts of it with a specified value.</td>
</tr>
<tr>
<td>REVERSE</td>
<td>Reverses the order of one dimension of an array.</td>
</tr>
<tr>
<td>ROT</td>
<td>Rotates an image by any amount.</td>
</tr>
<tr>
<td>ROTATE</td>
<td>Rotates/transposes an array in multiples of 90 degrees.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>Shifts elements of vectors or arrays by a specified number of elements.</td>
</tr>
<tr>
<td>SIZE</td>
<td>Returns array size and type information.</td>
</tr>
<tr>
<td>SORT</td>
<td>Returns indices of an array sorted in ascending order.</td>
</tr>
<tr>
<td>TOTAL</td>
<td>Sums of the elements of an array.</td>
</tr>
<tr>
<td>TRANSPOSE</td>
<td>Transposes an array.</td>
</tr>
</tbody>
</table>
### UNIQ
- Returns subscripts of the unique elements in an array.

### WHERE
- Returns subscripts of nonzero array elements.

### XVAREDIT
- Provides widget-based editor for IDL variables.

## Color Table Manipulation

### COLOR_CONVERT
- Converts color triples to and from RGB, HLS, and HSV.

### COLOR_QUAN
- Converts true-color (24-bit) image to pseudo-color (8-bit) image.

### CT_LUMINANCE
- Calculates the luminance of colors.

### CW_PALETTE_EDITOR
- Provides widget to display and edit color palettes.

### CW_PALETTE_EDITOR_GET
- Gets CW_PALETTE_EDITOR properties.

### CW_PALETTE_EDITOR_SET
- Sets CW_PALETTE_EDITOR properties.

### GAMMA_CT
- Applies gamma correction to a color table.

### H_EQ_CT
- Histogram-equalizes the color tables for an image or a region of the display.

### H_EQ_INT
- Interactively histogram-equalizes the color tables of an image or a region of the display.

### HLS
- Creates color table in Hue, Lightness, Saturation color system.

### HSV
- Creates color table based on Hue and Saturation Value color system.

### LOADCT
- Loads one of the predefined IDL color tables.

### MODIFYCT
- Saves modified color tables in the IDL color table file.

### MULTI
- Replicates current color table to enhance contrast.

### PSEUDO
- Creates pseudo-color table based on Hue, Lightness, and Brightness system.

### REDUCE_COLORS
- Reduces the number of colors used in an image by eliminating unused pixel values.

### STRETCH
- Stretches color table for contrast enhancement.

### TEK_COLOR
- Loads color table based on Tektronix printer.

### TVLC
- Loads display color tables.

### XLOADCT
- Provides GUI to interactively select and load color tables.

### XPalette
- Displays widget used to create and modify color tables.

## Debugging

### .CONTINUE
- Continues execution of a stopped program.

### .SKIP
- Skips over the next n statements and then single steps.

### .STEP
- Executes one or n statements from the current position.

### .STEPOVER
- Executes a single statement if the statement doesn’t call a routine.

### .TRACE
- Similar to .CONTINUE, but displays each line of code before execution.

### BREAKPOINT
- Sets and clears breakpoints for debugging.

### SHMDEBUG
- Print debugging information when a variable loses reference to an underlying shared memory segment.

### STOP
- Stops the execution of a running program or batch file.

## Dialog Routines

### DIALOG_MESSAGE
- Creates modal message dialog.

### DIALOG_PICKFILE
- Creates native file-selection dialog.

### DIALOG_PRINTERSETUP
- Opens native dialog used to set properties for a printer.

### DIALOG_PRINTJOB
- Opens native dialog used to set parameters for a print job.

### DIALOG_READ_IMAGE
- Presents GUI for reading image files.

### DIALOG_WRITE_IMAGE
- Presents GUI for writing image files.

## Direct Graphics, General

### ANNOTATE
- Starts IDL widget used to interactively annotate images and plots with text and drawings.

### ARROW
- Draws line with an arrow head.

### BOX_CURSOR
- Emulates operation of a variable-sized box cursor.

### CONVERTCOORD
- Transforms coordinates to and from the coordinate systems supported by IDL.

### CURSOR
- Reads position of the interactive graphics cursor.

### CVTTOBM
- Creates a bitmap byte array for a button label.

### DEVICE
- Sets to plot in device coordinates.

### EMPTY
- Empties the graphics output buffer.

### ERASE
- Erases screen of the current graphics device, or starts a new page if the device is a printer.

### FORMAT_AXIS_VALUES
- Formats numbers as strings for use as axis values.

### PLOTS
- Plots vectors and points.

### POLYFILL
- Fills the interior of a polygon.

### PROFILE
- Extracts a profile from an image.

### PROFILES
- Interactively examines image profiles.

### SET_PLOT
- Sets the output device used by the IDL direct graphics procedures.

### THREAD
- Plots a 2D array as a pseudo 3D plot.

### TV
- Displays an image.
**Functional List of IDL Routines**

**TVCRS** - Manipulates the image display cursor.
**TVSCL** - Scales and displays an image.
**XYOUTS** - Draws text on currently-selected graphics device.
**ZOOM** - Zooms portions of the display.
**ZOOM_24** - Zooms portions of true-color (24-bit) display.

**Error Handling**

**CATCH** - Intercepts and processes error messages, and continues program execution.
**MESSAGE** - Issues error and informational messages.
**ON_ERROR** - Designates the error recovery method.
**ON_IOERROR** - Declares I/O error exception handler.
**STRMESSAGE** - Returns the text of a given error number.

**Executive Commands**

**.COMPILE** - Compiles programs without running.
**.CONTINUE** - Continues execution of a stopped program.
**.EDIT** - Opens files in editor windows of the IDLDE (Windows and Motif only).
**.FULL_RESET_SESSION** - Does everything .RESET_SESSION does, plus additional reset tasks such as unloading sharable libraries.
**.GO** - Executes previously-compiled main program.
**.OUT** - Continues execution until the current routine returns.
**.RESET_SESSION** - Resets much of the state of an IDL session without requiring the user to exit and restart the IDL session.
**.RETURN** - Continues execution until RETURN statement.
**.RNEW** - Erases main program variables and then does .RUN.
**.RUN** - Compiles and executes IDL commands from files or keyboard.
**.SKIP** - Skips over the next n statements and then single steps.
**.STEP** - Executes one or n statements from the current position.
**.STEPOVER** - Executes a single statement if the statement doesn't call a routine.
**.TRACE** - Similar to .CONTINUE, but displays each line of code before execution.

**External Linking**

**CALL_EXTERNAL** - Calls a function in an external sharable object and returns a scalar value.
**DLM_LOAD** - Explicitly causes a DLM to be loaded.
**IDLcomActiveX** - Creates an IDL object that encapsulates an ActiveX control.
**IDLcomDispatch** - Creates an IDL object that encapsulates a COM object.
**IDLjavaObject** - An IDL object encapsulating a Java object. IDL provides data type and other translation services, allowing IDL programs to access the Java object’s methods and properties using standard IDL syntax.
**LINKIMAGE** - Merges routines written in other languages with IDL at run-time.
**MAKE_DLL** - Compiles and links sharable libraries (DLLs).

**Font Manipulation**

**EFONT** - Interactive vector font editor and display tool.
**PS_SHOW_FONTS** - Displays all the PostScript fonts that IDL knows about.
**PSAFM** - Converts Adobe Font Metrics file to IDL format.
**SHOWFONT** - Displays a TrueType or vector font
**XFONT** - Creates modal widget to select and view an X Windows font.

**Help Routines**

**?** - Invokes the IDL Online Help facility when entered at the IDL command line.
**DOC_LIBRARY** - Extracts documentation headers from IDL programs.
**HELP** - Provides information about the current IDL session.
**MEMORY** - Returns information about dynamic memory currently in use by the IDL session.
**MK_HTML_HELP** - Converts text documentation headers to HTML files.
**ONLINE_HELP** - Invokes online help viewer from programs.
**STRUCT_HIDE** - Prevents the IDL HELP procedure from displaying information about structures or objects.

**Image Processing**

**Contrast Enhancement and Filtering**

**ADAPT_HIST_EQUAL** - Performs adaptive histogram equalization
**BYTSCL** - Scales all values of an array into range of bytes.
**CONVOL** - Convolves two vectors or arrays.
**DIGITAL_FILTER** - Calculates coefficients of a non-recursive, digital filter.
**FFT** - Returns the Fast Fourier Transform of an array.
**HILBERT** - Constructs a Hilbert transform.
**HIST_EQUAL** - Histogram-equalizes an image.
**LEEFILT** - Performs the Lee filter algorithm on an image array.
**MEDIAN** - Returns the median value of Array or applies a median filter.
**ROBERTS** - Returns an approximation of Roberts edge enhancement.
**SMOOTH** - Smooths with a boxcar average.
**SOBEL** - Returns an approximation of Sobel edge enhancement.
### Functional List of IDL Routines

#### Image Processing

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNSHARP_MASK</td>
<td>Performs an unsharp-mask sharpening filter on a two-dimensional array or a truecolor image.</td>
</tr>
<tr>
<td>See Also</td>
<td>Wavelet Toolkit</td>
</tr>
</tbody>
</table>

#### Feature Extraction/Image Segmentation

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTOUR</td>
<td>Draws a contour plot.</td>
</tr>
<tr>
<td>DEFROI</td>
<td>Defines an irregular region of interest of an image.</td>
</tr>
<tr>
<td>HISTOGRAM</td>
<td>Computes the density function of an array.</td>
</tr>
<tr>
<td>HOUGH</td>
<td>Returns the Hough transform of a two-dimensional image.</td>
</tr>
<tr>
<td>IMAGE_STATISTICS</td>
<td>Computes sample statistics for a given array of values.</td>
</tr>
<tr>
<td>ISOCONTOUR</td>
<td>Interprets the contouring algorithm found in the IDLgrContour object.</td>
</tr>
<tr>
<td>ISOSURFACE</td>
<td>Returns topologically consistent triangles by using oriented tetrahedral decomposition.</td>
</tr>
<tr>
<td>LABEL_REGION</td>
<td>Labels regions (blobs) of a bi-level image.</td>
</tr>
<tr>
<td>MAX</td>
<td>Returns the value of the largest element of Array.</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>Returns the median value of Array or applies a median filter.</td>
</tr>
<tr>
<td>MIN</td>
<td>Returns the value of the smallest element of an array.</td>
</tr>
<tr>
<td>PROFILES</td>
<td>Interactively examines image profiles.</td>
</tr>
<tr>
<td>RADON</td>
<td>Returns the Radon transform of a two-dimensional image.</td>
</tr>
<tr>
<td>REGION_GROW</td>
<td>Perform region growing.</td>
</tr>
<tr>
<td>SEARCH2D</td>
<td>Finds “objects” or regions of similar data within a 2D array.</td>
</tr>
<tr>
<td>THIN</td>
<td>Returns the “skeleton” of a bi-level image.</td>
</tr>
<tr>
<td>UNIQ</td>
<td>Returns subscripts of the unique elements in an array.</td>
</tr>
<tr>
<td>WATERSHED</td>
<td>Applies the morphological watershed operator to a grayscale image.</td>
</tr>
<tr>
<td>WHERE</td>
<td>Returns subscripts of nonzero array elements.</td>
</tr>
</tbody>
</table>

#### Image Display

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISSOLVE</td>
<td>Provides a digital “dissolve” effect for images.</td>
</tr>
<tr>
<td>IIMAGE</td>
<td>Creates an iTool and associated user interface (UI) configured to display and manipulate image data.</td>
</tr>
<tr>
<td>RDPIX</td>
<td>Interactively displays image pixel values.</td>
</tr>
<tr>
<td>SLIDE_IMAGE</td>
<td>Creates a scrolling graphics window for examining large images.</td>
</tr>
<tr>
<td>TV</td>
<td>Displays an image.</td>
</tr>
<tr>
<td>TVCRS</td>
<td>Manipulates the image display cursor.</td>
</tr>
<tr>
<td>TVLCT</td>
<td>Loads display color tables.</td>
</tr>
<tr>
<td>TVSCL</td>
<td>Scales and displays an image.</td>
</tr>
<tr>
<td>XOBJVIEW</td>
<td>Displays object viewer widget.</td>
</tr>
<tr>
<td>XOBJVIEW_ROTATE</td>
<td>Programmatically rotate the object currently displayed in XOBJVIEW.</td>
</tr>
<tr>
<td>XOBJVIEW_WRITE_IMAGE</td>
<td>Write the object currently displayed in XOBJVIEW to an image file.</td>
</tr>
<tr>
<td>ZOOM</td>
<td>Zooms portions of the display.</td>
</tr>
<tr>
<td>ZOOM_24</td>
<td>Zooms portions of true-color (24-bit) display.</td>
</tr>
</tbody>
</table>

#### Image Geometry Transformations

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONGRID</td>
<td>Resamples an image to any dimensions.</td>
</tr>
<tr>
<td>EXPAND</td>
<td>Shrinks/expands image using bilinear interpolation.</td>
</tr>
<tr>
<td>EXTRAC</td>
<td>Returns sub-matrix of input array. Array operators (e.g., * and :) should usually be used instead.</td>
</tr>
<tr>
<td>INTERPOLATE</td>
<td>Returns an array of interpolates.</td>
</tr>
<tr>
<td>INVERT</td>
<td>Computes the inverse of a square array.</td>
</tr>
<tr>
<td>POLY_2D</td>
<td>Performs polynomial warping of images.</td>
</tr>
<tr>
<td>POLYWARP</td>
<td>Performs polynomial spatial warping.</td>
</tr>
<tr>
<td>REBIN</td>
<td>Resizes a vector or array by integer multiples.</td>
</tr>
<tr>
<td>REVERSE</td>
<td>Reverses the order of one dimension of an array.</td>
</tr>
<tr>
<td>ROT</td>
<td>Rotates an image by any amount.</td>
</tr>
<tr>
<td>ROTATE</td>
<td>Rotates/transposes an array in multiples of 90 degrees.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>Shifts elements of vectors or arrays by a specified number of elements.</td>
</tr>
<tr>
<td>TRANSPOSE</td>
<td>Transposes an array.</td>
</tr>
<tr>
<td>WARP_TRI</td>
<td>Warps an image using control points.</td>
</tr>
</tbody>
</table>

#### Morphological Image Operators

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DILATE</td>
<td>Implements morphologic dilation operator on binary and grayscale images.</td>
</tr>
<tr>
<td>ERODE</td>
<td>Implements the erosion operator on binary and grayscale images and vectors.</td>
</tr>
<tr>
<td>LABEL_REGION</td>
<td>Labels regions (blobs) of a bi-level image.</td>
</tr>
<tr>
<td>MORPH_CLOSE</td>
<td>Applies closing operator to binary or grayscale image.</td>
</tr>
<tr>
<td>MORPH_DISTANCE</td>
<td>Estimates N-dimensional distance maps, which contain for each foreground pixel the distance to the nearest background pixel, using a given norm.</td>
</tr>
<tr>
<td>MORPH_GRADIENT</td>
<td>Applies the morphological gradient operator to a grayscale image.</td>
</tr>
<tr>
<td>MORPH_HITORMISS</td>
<td>Applies the hit-or-miss operator to a binary image.</td>
</tr>
<tr>
<td>MORPH_OPEN</td>
<td>Applies the opening operator to a binary or grayscale image.</td>
</tr>
<tr>
<td>MORPH_THIN</td>
<td>Performs a thinning operation on binary images.</td>
</tr>
<tr>
<td>MORPH_TOPHAT</td>
<td>Applies top-hat operator to a grayscale image.</td>
</tr>
<tr>
<td>WATERSHED</td>
<td>Applies the morphological watershed operator to a grayscale image.</td>
</tr>
</tbody>
</table>

#### Regions of Interest

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CW_DEFROI</td>
<td>Creates compound widget used to define region of interest.</td>
</tr>
<tr>
<td>DEFROI</td>
<td>Defines an irregular region of interest of an image.</td>
</tr>
<tr>
<td>DRAW_ROI</td>
<td>Draws region or group of regions to current Direct Graphics device.</td>
</tr>
<tr>
<td>IDLanROI</td>
<td>Represents a region of interest.</td>
</tr>
</tbody>
</table>
**Functional List of IDL Routines**

**IDLanROIGroup** - Analytical representation of a group of regions of interest.

**IDLgROI** - Object graphics representation of a region of interest.

**IDLgROIGroup** - Object Graphics representation of a group of regions of interest.

**LABEL_REGION** - Labels regions (blobs) of a bi-level image.

**XROI** - Utility for defining regions of interest, and obtaining geometry and statistical data about these ROIs.

**Input/Output**

**APP_USER_DIR** - Provides access to the application user directory.

**APP_USER_DIR_QUERY** - Locates existing application user directories.

**ASCII_TEMPLATE** - Presents a GUI that generates a template defining an ASCII file format.

**ASSOC** - Associates an array structure with a file.

**BINARY_TEMPLATE** - Presents a GUI for interactively generating a template structure for use with READ_BINARY.

**CDF Routines** - Alphabetical Listing of CDF Routines.

**CLOSE** - Closes the specified files.

**COPY_LUN** - Copies data between two open files.

**DIALOG_READ_IMAGE** - Presents GUI for reading image files.

**DIALOG_WRITE_IMAGE** - Presents GUI for writing image files.

**EOF** - Tests the specified file for the end-of-file condition.

**EOS Routines** - Alphabetical Listing of EOS Routines.

**FILE_COPY** - Copies files or directories to a new location.

**FILE_INFO** - Returns status information about a file.

**FILE_LINES** - Returns the number of lines of text in a file.

**FILE_LINK** - Creates Unix file links.

**FILE_MOVE** - Renames files and directories.

**FILE_READLINK** - Returns the path pointed to by a Unix symbolic link.

**FILE_SEARCH** - Returns a string file names refer to the same underlying file.

**FILE_TEST** - Test a file or directory for existence and other specific attributes.

**FILEPATH** - Returns full path to a file in the IDL distribution.

**FLUSH** - Flushes file unit buffers.

**FREE_LUN** - Frees previously-reserved file units.

**FSTAT** - Returns information about a specified file unit.

**GET_KBRD** - Gets one input IDL character.

**GET_LUN** - Reserves a logical unit number (file unit).

**HDF BROWSER** - Opens a GUI to view contents of HDF, HDF-EOS, or NetCDF file.

**HDF_READ** - Extracts HDF, HDF-EOS, and NetCDF data and metadata into an output structure.

**HDF_WRITE** - Writes HDF, HDF-EOS, and NetCDF data into a file.

**HDF_BROWSER** - Opens a GUI to view contents of HDF, HDF-EOS, or NetCDF file.

**HDF_READ** - Extracts HDF, HDF-EOS, and NetCDF data and metadata into an output structure.

**HDF_WRITE** - Writes HDF, HDF-EOS, and NetCDF data into a file.

**HDF_ROUTINES** - Alphabetical Listing of HDF Routines.

**HDF5_ROUTINES** - Alphabetical Listing of HDF5 Routines.

**IDLHDFICOM** - Contains the data for one or more images embedded in a DCOM part 10 file.

**IDLHDXF** - Object that contains geometry, connectivity, and attributes for graphics primitives.

**IDLHShape** - Contains geometry, connectivity and attributes for graphics primitives accessed from ESRI Shapefiles.

**IOCTL** - Performs special functions on UNIX files.

**MPEG_CLOSE** - Closes an MPEG sequence.

**MPEG_OPEN** - Opens an MPEG sequence.

**MPEG_PUT** - Inserts an image array into an MPEG sequence.

**MPEG_SAVE** - Saves an MPEG sequence to a file.

**NCDF Routines** - Alphabetical Listing of NCDF Routines.

**OPEN** - Opens files for reading, updating, or writing.

**PATH_SEP** - Returns the proper file path segment separator character for the current operating system.

**POINT_LUN** - Sets or gets current position of the file pointer.

**PRINT/PRINTF** - Writes formatted output to screen or file.

**READ/READF** - Reads formatted input from keyboard or file.

**READ_ASCII** - Reads data from an ASCII file.

**READ_BMP** - Reads Microsoft Windows bitmap file (.BMP).

**READ_BINARY** - Reads the contents of a binary file using a passed template or basic command line keywords.

**READ_BMP** - Reads Microsoft Windows bitmap file (.BMP).

**READ_BINARY** - Reads the contents of a binary file using a passed template or basic command line keywords.

**READ_DICOM** - Reads an image from a DICOM file.

**READ_IMAGE** - Reads the image contents of a file and returns the image in an IDL variable.

**READ_INTERFILE** - Reads Interfile (v3.3) file.

**READ_JPEG** - Reads JPEG file.


**READ_MRSID** - Reads MrSID file.

**READ_PICT** - Reads Macintosh PICT (version 2) bitmap file.


**READ_PPM** - Reads PGM (gray scale) or PPM (portable pixmap for color) file.

**READ_SRFL** - Reads Sun Raster Format file.

**READ_SYLK** - Reads Symbolic Link format spreadsheet file.

**READ_TIFF** - Reads TIFF format file.

**READ_WAV** - Reads the audio stream from the named WAV file.

**READ_WAVE** - Reads Wavefront Advanced Visualizer file.


**READ_XWD** - Reads X Windows Dump file.

**READS** - Reads formatted input from a string variable.

**READU** - Reads unformatted binary data from a file.

**SHMMAP** - Maps anonymous shared memory, or local disk files, into the memory address space of the currently executing IDL process.

**SHMMAP** - Maps anonymous shared memory, or local disk files, into the memory address space of the currently executing IDL process.

**SHMMAP** - Maps anonymous shared memory, or local disk files, into the memory address space of the currently executing IDL process.

**SHUNMAP** - Removes a memory segment previously created by SHMMAP from the system.
SHMVAR - Creates an IDL array variable that uses the memory from a current mapped memory segment created by the SHMMAP procedure.

SKIP_LUN - Reads data in an open file and moves the file pointer.

SOCKET - Opens a client-side TCP/IP Internet socket as an IDL file unit.

TRUNCATE_LUN - Truncates an open file at the location of the current file pointer.

TVRD - Reads an image from a window into a variable.

WRITE_BMP - Writes Microsoft Windows Version 3 device independent bitmap file (.BMP).

WRITE_IMAGE - Writes an image and its color table vectors, if any, to a file of a specified type.

WRITE_JPEG - Writes JPEG file.


WRITE_PICT - Writes Macintosh PICT (version 2) bitmap file.


WRITE_PPM - Writes PPM (true-color) or PGM (gray scale) file.

WRITE_SRF - Writes Sun Raster File (SRF).

WRITE_SYLK - Writes SYLK (Symbolic Link) spreadsheet file.

WRITE_TIFF - Writes TIFF file with 1 to 3 channels.

WRITE_WAV - Writes the audio stream to the named .WAV file.

WRITE_WAVE - Writes Wavefront Advanced Visualizer (.WAV) file.

XOBJVIEW_WRITE_IMAGE - Writes the object currently displayed in XOBJVIEW to an image file.

Language Catalogs

IDLffLangCat - Finds and loads an XML language catalog.

LOCALE_GET - Returns the current locale of the operating platform.

MULTI - Returns a catalog object for the given parameters if found.

Mapping

IMAP - Displays georeferenced data in an iTool.

LL_ARC_DISTANCE - Returns the longitude and latitude of a point given arc distance and azimuth.

MAP_CONTINENTS - Draws continental boundaries, filled continents, political boundaries, coastlines, and rivers, over an existing map projection established by the MAP_SET.

MAP_GRID - Draws parallels and meridians over a map projection.

MAP_IMAGE - Returns an image warped to fit the current map projection. (Use when map data is larger than the display).

MAP_PATCH - Returns an image warped to fit the current map projection. (Use when map data is smaller than the display).

MAP_PROJ_FORWARD - Transforms map coordinates from longitude/latitude to Cartesian (X, Y) coordinates.

MAP_PROJ_IMAGE - Warps an image from geographic coordinates to a specified map projection.

MAP_PROJ_INFO - Returns information about current map and/or the available projections.

MAP_PROJ_INIT - Initializes a mapping projection, using either IDL's own map projections or the General Cartographic Transformation Package (GCTP) map projections.

MAP_PROJ_INVERSE - Transforms map coordinates from Cartesian (X, Y) coordinates to longitude/latitude.

MAP_SET - Establishes map projection type and limits.

Mathematics

Complex Numbers

COMPLEX - Converts argument to complex type.

CONJ - Returns the complex conjugate of X.

DCOMPLEX - Converts argument to double-precision complex type.

IMAGINARY - Returns the imaginary part of a complex value.

REAL_PART - Returns the real part of a complex-valued argument.

Correlation Analysis

A_CORRELATE - Computes autocorrelation.

C_CORRELATE - Computes cross correlation.

CORRELATE - Computes the linear Pearson correlation.

M_CORRELATE - Computes multiple correlation coefficient.

P_CORRELATE - Computes partial correlation coefficient.

R_CORRELATE - Computes rank correlation.

Curve and Surface Fitting

COMFIT - Fits paired data using one of six common filtering functions.

CRVLENGTH - Computes the length of a curve.

CURVEFIT - Fits multivariate data with a user-supplied function.

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.

GAUSSFIT - Fits the sum of a Gaussian and a quadratic.

GRID_TPS - Uses thin plate splines to interpolate a set of values over a regular 2D grid, from irregularly sampled data values.

KRIG2D - Interpolates set of points using kriging.

LADF - Fits paired data using least absolute deviation method.

LINFIT - Fits by minimizing the Chi-square error statistic.

LMFIT - Does a non-linear least squares fit.

MIN_CURVE_SURF - Interpolates points with a minimum curvature surface or a thin-plate-spline surface. Useful with CONTOUR.

POLYFIT - Performs a least-square polynomial fit.

REGRESS - Computes fit using multiple linear regression.

SFIT - Performs polynomial fit to a surface.

SVD - Multivariate least squares fit using SVD method.
### Functional List of IDL Routines

<table>
<thead>
<tr>
<th><strong>Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRIGRID</strong></td>
<td>Interpolates irregularly-gridded data to a regular grid from a triangulation.</td>
</tr>
</tbody>
</table>

#### Differentiation and Integration

<table>
<thead>
<tr>
<th><strong>Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CRVLENGTH</strong></td>
<td>Computes the length of a curve.</td>
</tr>
<tr>
<td><strong>DERIV</strong></td>
<td>Performs differentiation using 3-point Lagrangian interpolation.</td>
</tr>
<tr>
<td><strong>DERIVSIG</strong></td>
<td>Computes standard deviation of derivative found by DERIV.</td>
</tr>
<tr>
<td><strong>INT_2D</strong></td>
<td>Computes the double integral of a bivariate function.</td>
</tr>
<tr>
<td><strong>INT_3D</strong></td>
<td>Computes the triple integral of a trivariate function.</td>
</tr>
</tbody>
</table>

#### Gridding and Interpolation

<table>
<thead>
<tr>
<th><strong>Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INTERPOLATE</strong></td>
<td>Returns an array of interpolates.</td>
</tr>
<tr>
<td><strong>KRG2D</strong></td>
<td>Interpolates points using kriging.</td>
</tr>
<tr>
<td><strong>MIN_CURVE_SURF</strong></td>
<td>Interpolates points with a minimum curvature surface or a thin-plate-spline surface. Useful with CONTUR.</td>
</tr>
<tr>
<td><strong>POLAR_SURFACE</strong></td>
<td>Interpolates a surface from polar coordinates to rectangular coordinates.</td>
</tr>
<tr>
<td><strong>SPL_INIT</strong></td>
<td>Establishes the type of interpolating spline.</td>
</tr>
</tbody>
</table>

#### Eigenvalues and Eigenvectors

<table>
<thead>
<tr>
<th><strong>Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EIGENQL</strong></td>
<td>Computes eigenvalues and eigenvectors of a real, symmetric array.</td>
</tr>
<tr>
<td><strong>EIGENVEC</strong></td>
<td>Computes eigenvectors of a real, non-symmetric array.</td>
</tr>
<tr>
<td><strong>ELMHES</strong></td>
<td>Reduces nonsymmetric array to upper Hessenberg form.</td>
</tr>
</tbody>
</table>

#### Hypothesis Testing

<table>
<thead>
<tr>
<th><strong>Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CTI_TEST</strong></td>
<td>Performs chi-square goodness-of-fit test.</td>
</tr>
<tr>
<td><strong>FV_TEST</strong></td>
<td>Performs the F-variance test.</td>
</tr>
<tr>
<td><strong>KW_TEST</strong></td>
<td>Performs Kruskal-Wallis H-test.</td>
</tr>
<tr>
<td><strong>LNP_TEST</strong></td>
<td>Computes the Lomb Normalized Periodogram.</td>
</tr>
<tr>
<td><strong>MD_TEST</strong></td>
<td>Performs the Median Delta test.</td>
</tr>
<tr>
<td><strong>R_TEST</strong></td>
<td>Runs test for randomness.</td>
</tr>
<tr>
<td><strong>RS_TEST</strong></td>
<td>Performs the Wilcoxon Rank-Sum test.</td>
</tr>
<tr>
<td><strong>S_TEST</strong></td>
<td>Performs the Sign test.</td>
</tr>
<tr>
<td><strong>TM_TEST</strong></td>
<td>Performs t-means test.</td>
</tr>
<tr>
<td><strong>XSQ_TEST</strong></td>
<td>Computes Chi-square goodness-of-fit test.</td>
</tr>
</tbody>
</table>

#### LAPACK Routines

<table>
<thead>
<tr>
<th><strong>Function</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LA_CHOLDC</strong></td>
<td>Computes the Cholesky factorization of an $n$-by-$n$ symmetric positive-definite array.</td>
</tr>
<tr>
<td><strong>LA_CHOLMPROVE</strong></td>
<td>Uses Cholesky factorization to improve the solution to a system of linear equations.</td>
</tr>
<tr>
<td><strong>LA_CHOLESOL</strong></td>
<td>Used in conjunction with LA_CHOLDC to solve a set of linear equations.</td>
</tr>
<tr>
<td><strong>LA_DETERM</strong></td>
<td>Uses LU decomposition to compute the determinant of a square array.</td>
</tr>
<tr>
<td><strong>LA_EIGENPROBLEM</strong></td>
<td>Uses the QR algorithm to compute eigenvalues and eigenvectors of an array.</td>
</tr>
<tr>
<td><strong>LA_EIGENQL</strong></td>
<td>Computes selected eigenvalues and eigenvectors.</td>
</tr>
<tr>
<td><strong>LA_EIGENVEC</strong></td>
<td>Uses the QR algorithm to compute all of some eigenvectors of an array.</td>
</tr>
<tr>
<td><strong>LA_ELMHES</strong></td>
<td>Reduces a real nonsymmetric or complex array to upper Hessenberg form.</td>
</tr>
<tr>
<td><strong>LA_GM_LINEAR_MODEL</strong></td>
<td>Used to solve a general Gauss-Markov linear model problem.</td>
</tr>
<tr>
<td><strong>LA_HQR</strong></td>
<td>Uses the multi-shift QR algorithm to compute all eigenvalues of an array.</td>
</tr>
<tr>
<td><strong>LA_INVERT</strong></td>
<td>Uses LU decomposition to compute the inverse of a square array.</td>
</tr>
<tr>
<td><strong>LA_LEAST_SQUARE_EQUALITY</strong></td>
<td>Used to solve linear least-squares problems.</td>
</tr>
<tr>
<td><strong>LA_LEAST_SQUARES</strong></td>
<td>Used to solve linear least-squares problems.</td>
</tr>
<tr>
<td><strong>LA_LINEAR_EQUATION</strong></td>
<td>Uses LU decomposition to solve a system of linear equations.</td>
</tr>
<tr>
<td>Routine</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ABS</td>
<td>Returns the absolute value of X.</td>
</tr>
<tr>
<td>CEIL</td>
<td>Returns the closest integer greater than or equal to X.</td>
</tr>
<tr>
<td>CIR_3PNT</td>
<td>Returns radius and center of circle, given 3 points.</td>
</tr>
<tr>
<td>COMPLEXROUND</td>
<td>Rounds a complex array.</td>
</tr>
<tr>
<td>DIAG_MATRIX</td>
<td>Constructs a diagonal matrix from an input vector, or if given a matrix, then extracts a diagonal vector.</td>
</tr>
<tr>
<td>DIST</td>
<td>Creates array with each element proportional to its frequency.</td>
</tr>
<tr>
<td>EXP</td>
<td>Returns the natural exponential function of given expression.</td>
</tr>
<tr>
<td>FLOOR</td>
<td>Returns closest integer less than or equal to argument.</td>
</tr>
<tr>
<td>IMAGINARY</td>
<td>Returns the imaginary part of a complex value.</td>
</tr>
<tr>
<td>ISHIFT</td>
<td>Performs integer bit shift.</td>
</tr>
<tr>
<td>LEEFILT</td>
<td>Performs the Lee filter algorithm on an image array.</td>
</tr>
<tr>
<td>MATRIX_MULTIPLY</td>
<td>Calculates the IDL matrix-multiply operator (#) of two (possibly transposed) arrays.</td>
</tr>
<tr>
<td>MATRIX_POWER</td>
<td>Computes the product of a matrix with itself.</td>
</tr>
<tr>
<td>PNT_LINE</td>
<td>Returns the perpendicular distance between a point and a line.</td>
</tr>
<tr>
<td>POLY_AREA</td>
<td>Returns the area of a polygon given the coordinates of its vertices.</td>
</tr>
<tr>
<td>PRIMES</td>
<td>Computes the first K prime numbers.</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>Returns the product of elements within an array.</td>
</tr>
<tr>
<td>ROUND</td>
<td>Returns the integer closest to its argument.</td>
</tr>
<tr>
<td>SPH_4PNT</td>
<td>Returns center and radius of a sphere given 4 points.</td>
</tr>
<tr>
<td>SQRT</td>
<td>Returns the square root of X.</td>
</tr>
<tr>
<td>TOTAL</td>
<td>Sums of the elements of an array.</td>
</tr>
<tr>
<td>VOIGT</td>
<td>Calculates intensity of atomic absorption line (Voight) profile.</td>
</tr>
<tr>
<td>CLUST_WTS</td>
<td>Computes cluster weights of array for cluster analysis.</td>
</tr>
<tr>
<td>CLUSTER</td>
<td>Performs cluster analysis.</td>
</tr>
<tr>
<td>CLUSTER_TREE</td>
<td>Computes the hierarchical clustering for a set of m items in an n-dimensional space.</td>
</tr>
<tr>
<td>CTI_TEST</td>
<td>Performs chi-square goodness-of-fit test.</td>
</tr>
<tr>
<td>DENDRO_PLOT</td>
<td>Draws a two-dimensional dendrite plot on the current direct graphics device if given a hierarchical tree cluster, as created by CLUSTER_TREE.</td>
</tr>
<tr>
<td>DENDROGRAM</td>
<td>Constructs a dendrogram and returns a set of vertices and connectivity that can be used to visualize the dendrite plot if given a hierarchical tree cluster, as created by CLUSTER_TREE.</td>
</tr>
<tr>
<td>DISTANCE_MEASURE</td>
<td>Computes the pairwise distance between a set of items or observations.</td>
</tr>
<tr>
<td>KW_TEST</td>
<td>Performs Kruskal-Wallis H-test.</td>
</tr>
<tr>
<td>M_CORRELATE</td>
<td>Computes multiple correlation coefficient.</td>
</tr>
<tr>
<td>P_CORRELATE</td>
<td>Computes partial correlation coefficient.</td>
</tr>
<tr>
<td>PCOMP</td>
<td>Computes principal components/derived variables.</td>
</tr>
<tr>
<td>STANDARDIZE</td>
<td>Computes standardized variables.</td>
</tr>
<tr>
<td>BROYDEN</td>
<td>Solves nonlinear equations using Broyden’s method.</td>
</tr>
</tbody>
</table>

**Mathematics**

- **Mathematical Error Assessment**
  - **CHECK_MATH** - Returns and clears accumulated math error status.
  - **FINITE** - Returns True if its argument is finite.
  - **MACHAR** - Determines and returns machine-specific parameters affecting floating-point arithmetic.

- **Miscellaneous Math Routines**
  - **ABS** - Returns the absolute value of X.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FX_ROOT</strong></td>
<td>Computes real and complex roots of a univariate nonlinear function using an optimal Müller's method.</td>
</tr>
<tr>
<td><strong>FZ_ROOTS</strong></td>
<td>Finds the roots of a complex polynomial using Laguerre's method.</td>
</tr>
<tr>
<td><strong>NEWTON</strong></td>
<td>Solves nonlinear equations using Newton's method.</td>
</tr>
<tr>
<td><strong>AMOEBA</strong></td>
<td>Minimizes a function using downhill simplex method.</td>
</tr>
<tr>
<td><strong>CONSTRAINED_MIN</strong></td>
<td>Minimizes a function using Generalized Reduced Gradient Method.</td>
</tr>
<tr>
<td><strong>DFPMIN</strong></td>
<td>Minimizes a function using Davidon-Fletcher-Powell method.</td>
</tr>
<tr>
<td><strong>POWELL</strong></td>
<td>Minimizes a function using the Powell method.</td>
</tr>
<tr>
<td><strong>SIMPLEX</strong></td>
<td>Use the simplex method to solve linear programming problems.</td>
</tr>
<tr>
<td><strong>BINOMIAL</strong></td>
<td>Computes binomial distribution function.</td>
</tr>
<tr>
<td><strong>CHISQR_CVF</strong></td>
<td>Computes cutoff value in a Chi-square distribution.</td>
</tr>
<tr>
<td><strong>CHISQR_PDF</strong></td>
<td>Computes Chi-square distribution function.</td>
</tr>
<tr>
<td><strong>F_CVF</strong></td>
<td>Computes the cutoff value in an F distribution.</td>
</tr>
<tr>
<td><strong>F_PDF</strong></td>
<td>Computes the F distribution function.</td>
</tr>
<tr>
<td><strong>GAUSS_CVF</strong></td>
<td>Computes cutoff value in Gaussian distribution.</td>
</tr>
<tr>
<td><strong>GAUSS_PDF</strong></td>
<td>Computes Gaussian distribution function.</td>
</tr>
<tr>
<td><strong>GAUSSINT</strong></td>
<td>Returns integral of Gaussian probability function.</td>
</tr>
<tr>
<td><strong>T_CVF</strong></td>
<td>Computes the cutoff value in a Student's t distribution.</td>
</tr>
<tr>
<td><strong>T_PDF</strong></td>
<td>Computes Student's t distribution.</td>
</tr>
<tr>
<td><strong>FULSTR</strong></td>
<td>Restores a sparse matrix to full storage mode.</td>
</tr>
<tr>
<td><strong>LINBCG</strong></td>
<td>Solves a set of sparse linear equations using the iterative biconjugate gradient method.</td>
</tr>
<tr>
<td><strong>READ_SPR</strong></td>
<td>Reads a row-indexed sparse matrix from a file.</td>
</tr>
<tr>
<td><strong>SPRSAB</strong></td>
<td>Performs matrix multiplication on sparse matrices.</td>
</tr>
<tr>
<td><strong>SPRSAX</strong></td>
<td>Multiplies sparse matrix by a vector.</td>
</tr>
<tr>
<td><strong>SPRSIN</strong></td>
<td>Converts matrix to row-index sparse matrix.</td>
</tr>
<tr>
<td><strong>SPRSTP</strong></td>
<td>Constructs the transpose of a sparse matrix.</td>
</tr>
<tr>
<td><strong>WRITE_SPR</strong></td>
<td>Writes row-indexed sparse array structure to a file.</td>
</tr>
<tr>
<td><strong>BESELI</strong></td>
<td>Returns the I Bessel function of order N for X.</td>
</tr>
<tr>
<td><strong>BESELJ</strong></td>
<td>Returns the J Bessel function of order N for X.</td>
</tr>
<tr>
<td><strong>BESELK</strong></td>
<td>Returns the K Bessel function of order N for X.</td>
</tr>
<tr>
<td><strong>BESELY</strong></td>
<td>Returns the Y Bessel function of order N for X.</td>
</tr>
<tr>
<td><strong>BETA</strong></td>
<td>Returns the value of the beta function.</td>
</tr>
<tr>
<td><strong>ERF</strong></td>
<td>Returns the value of an error function.</td>
</tr>
<tr>
<td><strong>ERFC</strong></td>
<td>Returns the value of a complementary error function.</td>
</tr>
<tr>
<td><strong>ERFCX</strong></td>
<td>Returns the value of a scaled complementary error function.</td>
</tr>
<tr>
<td><strong>EXPINT</strong></td>
<td>Returns the value of the exponential integral.</td>
</tr>
<tr>
<td><strong>GAMMA</strong></td>
<td>Returns the gamma function of X.</td>
</tr>
<tr>
<td><strong>IBETA</strong></td>
<td>Computes the incomplete beta function.</td>
</tr>
<tr>
<td><strong>IGAMMA</strong></td>
<td>Computes the incomplete gamma function.</td>
</tr>
<tr>
<td><strong>LAGUERRE</strong></td>
<td>Returns value of the associated Laguerre polynomial.</td>
</tr>
<tr>
<td><strong>LEGENDRE</strong></td>
<td>Returns value of the associated Legendre polynomial.</td>
</tr>
<tr>
<td><strong>LNGAMMA</strong></td>
<td>Returns logarithm of the gamma function of X.</td>
</tr>
<tr>
<td><strong>POLY</strong></td>
<td>Evaluates polynomial function of a variable.</td>
</tr>
<tr>
<td><strong>SPHER_HARM</strong></td>
<td>Returns value of the spherical harmonic function.</td>
</tr>
<tr>
<td><strong>COMFIT</strong></td>
<td>Fits paired data using one of six common filtering functions.</td>
</tr>
<tr>
<td><strong>CURVEFIT</strong></td>
<td>Fits multivariate data with a user-supplied function.</td>
</tr>
<tr>
<td><strong>FUNCT</strong></td>
<td>Evaluates the sum of a Gaussian and a 2nd-order polynomial and optionally returns the value of its partial derivatives.</td>
</tr>
<tr>
<td><strong>LADFIT</strong></td>
<td>Fits paired data using least absolute deviation method.</td>
</tr>
<tr>
<td><strong>LINFIT</strong></td>
<td>Fits by minimizing the Chi-square error statistic.</td>
</tr>
<tr>
<td><strong>REGRESS</strong></td>
<td>Multiple linear regression.</td>
</tr>
<tr>
<td><strong>SVDFIT</strong></td>
<td>Multivariate least squares fit using SVD method.</td>
</tr>
<tr>
<td><strong>FACTORIAL</strong></td>
<td>Computes the factorial N!.</td>
</tr>
<tr>
<td><strong>HIST_2D</strong></td>
<td>Returns histogram of two variables.</td>
</tr>
<tr>
<td><strong>HISTOGRAM</strong></td>
<td>Computes the density function of an array.</td>
</tr>
<tr>
<td><strong>KURTOSIS</strong></td>
<td>Computes statistical kurtosis of n-element vector.</td>
</tr>
<tr>
<td><strong>MAX</strong></td>
<td>Returns the value of the largest element of an array.</td>
</tr>
<tr>
<td><strong>MEAN</strong></td>
<td>Computes the mean of a numeric vector.</td>
</tr>
<tr>
<td><strong>MEANABSDEV</strong></td>
<td>Computes the mean absolute deviation of a vector.</td>
</tr>
<tr>
<td><strong>MEDIAN</strong></td>
<td>Returns the median value of Array or applies a median filter.</td>
</tr>
<tr>
<td><strong>MIN</strong></td>
<td>Returns the value of the smallest element of an array.</td>
</tr>
<tr>
<td><strong>MOMENT</strong></td>
<td>Computes mean, variance, skewness, and kurtosis.</td>
</tr>
<tr>
<td><strong>RANDOMN</strong></td>
<td>Returns normally-distributed pseudo-random numbers.</td>
</tr>
<tr>
<td><strong>RANDOMU</strong></td>
<td>Returns uniformly-distributed pseudo-random numbers.</td>
</tr>
<tr>
<td><strong>RANKS</strong></td>
<td>Computes magnitude-based ranks.</td>
</tr>
<tr>
<td><strong>SKEWNESS</strong></td>
<td>Computes statistical skewness of an n-element vector.</td>
</tr>
<tr>
<td><strong>SORT</strong></td>
<td>Returns the indices of an array sorted in ascending order.</td>
</tr>
<tr>
<td><strong>STDDEV</strong></td>
<td>Computes statistical skewness of an n-element vector.</td>
</tr>
<tr>
<td><strong>VARIANCE</strong></td>
<td>Computes the statistical variance of an n-element vector.</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>Sums of the elements of an array.</td>
</tr>
<tr>
<td><strong>A_CORRELATE</strong></td>
<td>Computes autocorrelation.</td>
</tr>
<tr>
<td><strong>C_CORRELATE</strong></td>
<td>Computes cross correlation.</td>
</tr>
<tr>
<td><strong>SMOOTH</strong></td>
<td>Smooths with a boxcar average.</td>
</tr>
</tbody>
</table>
Functional List of IDL Routines

Transforms

TS_COEF - Computes the coefficients for autoregressive time-series.
TS_DIFF - Computes the forward differences of a time-series.
TS_FCAST - Computes future or past values of stationary time-series.
TS_SMOOTH - Computes moving averages of a time-series.

Transcendental Functions

ACOS - Returns the arc-cosine of X.
ALOG - Returns the natural logarithm of X.
ALOG10 - Returns the logarthihm to the base 10 of X.
ASIN - Returns the arc-sine of X.
ATAN - Returns the arc-tangent of X.
COS - Returns the cosine of X.
COSH - Returns the hyperbolic cosine of X.
EXP - Returns the natural exponential function of a given expression.
SIN - Returns the trigonometric sine of X.
SINH - Returns the hyperbolic sine of X.
TAN - Returns the tangent of X.
TANH - Returns the hyperbolic tangent of X.

Transforms

BLK_CON - Convolves input signal with impulse-response sequence.
CHEBYSHEV - Returns the forward or reverse Chebyshev polynomial expansion.
CONVOL - Convolves two vectors or arrays.
FFT - Returns the Fast Fourier Transform of an array.
HILBERT - Constructs a Hilbert transform.
HOUGH - Returns the Hough transform of a two-dimensional image.
RADON - Returns the Radon transform of a two-dimensional image.
WTN - Returns wavelet transform of the input array.

See Also - Wavelet Toolkit

Object Class Library

IDL_Container - Object that holds other objects.
IDLROI - Represents a region of interest.
IDLROIGroup - Analytical representation of a group of regions of interest.
IDLIDICOM - Contains the data for one or more images embedded in a DICOM Part 10 file.
IDLIDXF - Contains geometry, connectivity and attributes for graphics primitives.
IDLJPEG2000 - Contains the data for one or more images embedded in a JPEG-2000 file as well as functionality for reading and writing JPEG-2000 files.
IDLIMrsID - Used to query information about and load image data from a MeSID (.sid) image file.
IDLShape - Contains geometry, connectivity and attributes for graphics primitives.
IDLXMLSAX - Represents an XML SAX level 2 parser.
IDLXMLDOMAttr - Represents an attribute that is part of an element object in an XML document.
IDLXMLDOMCDATASection - Used to escape blocks of text in an XML document containing characters that would otherwise be regarded as markup.
IDLXMLDOMCharacterData - Extension of the IDLXMLDOMNode class that supplies a set of methods for accessing character data in the XML DOM tree.
IDLXMLDOMComment - Represents the content of an XML comment (characters between “<!--” and “-->”).
IDLXMLDOMNode - Represents the entire XML document as the root of the XML document tree and by providing the primary access to the document’s data.
IDLXMLDOMDocumentFragment - Represents a document fragment in an XML document.
IDLXMLDOMNodeType - References a DocumentType node in an XML document.
IDLXMLDOMElement - References an element node in an XML document.
IDLXMLDOMEntity - References an entity, either parsed or unparsed, in an XML document.
IDLXMLDOMEntityReference - References an entity reference node in an XML document.
IDLXMLDOMNamedNodeMap - Container for IDLXMLDOMNode nodes that uses node names to access the nodes.
IDLXMLDOMNode - Abstract class used as a superclass for other IDLXMLDOMNode node classes.
IDLXMLDOMNodeList - Container for IDLXMLDOMNode nodes.
IDLXMLDOMProcessingInstruction - Represents a processing instruction in an XML document.
IDLXMLDOMText - References a text node in the XML document.
IDLgrAxis - Represents a single vector that may include a set of tick marks, tick labels, and a title.
IDLgrBuffer - An in-memory, off-screen destination object.
IDLgrClipboard - A destination object representing the native clipboard.
IDLgrColorbar - Consists of a color-ramp with an optional framing box and annotation axis.
IDLgrContour - Draws a contour plot from data stored in a rectangular array or from a set of unstructured points.
IDLgrFont - Represents a typeface, style, weight, and point size that may be associated with text objects.
IDLgrImage - Creates an MPEG movie file from an array of image frames.
IDLgrPalette - Represents a color lookup table that maps indices to red, green, and blue values.
Functional List of IDL Routines

IDLrPattern - Describes which pixels are filled and which are left blank when an area is filled.

IDLrPlot - Creates set of polylines connecting data points in 2D space.

IDLrPolygon - Represents one or more polygons that share a set of vertices and rendering attributes.

IDLrPolyline - Represents one or more polylines that share a set of vertices and rendering attributes.

IDLrPrinter - Represents a hardcopy graphics destination.

IDLrROI - Object graphics representation of a region of interest.

IDLrROIGroup - Object graphics representation of a group of regions of interest.

IDLrScene - Represents the entire scene to be drawn and serves as a container of IDLrView or IDLrViewgroup objects.

IDLrSurface - A shaded or vector representation of a mesh grid. No superclasses.

IDLrSymbol - Represents a graphical element that is plotted relative to a particular position.

IDLrTessellator - Converts a simple concave polygon (or a simple polygon with "holes") into a number of simple convex polygons (general triangles).

IDLrText - Represents one or more text strings that share common rendering attributes.

IDLrView - Represents a rectangular area in which graphics objects are drawn. It is a container for objects of the IDLrModel class.

IDLrViewgroup - A simple container object that contains one or more IDLrView objects. An IDLrScene can contain one or more of these objects.

IDLrVolume - Represents mapping from a 3D array of data to a 3D array of voxel colors, which, when drawn, are projected to two dimensions.

IDLrVRML - Saves the contents of an Object Graphics hierarchy into a VRML 2.0 format file.

IDLrWindow - Represents an on-screen area on a display device that serves as a graphics destination.

IDLManipulator - The base functionality of the iTools manipulator system.

IDLManipulatorContainer - A container for IDLManipulator objects, which allows for the construction of manipulator hierarchies. This container implements the concept of a current manipulator for the items it contains.

IDLManipulatorManager - A specialization of the manipulator container (IDLManipulatorContainer), which acts as the root of the manipulator hierarchy.

IDLManipulatorVisual - The means for iTool developers to create visual elements associated with an interactive manipulator.

IDLOperation - The basis for all iTool operations. It defines how an operation is executed and how information about the operation is recorded for the command transaction (undo-redo) system.

IDLParameter - An interface providing parameter management methods to associate parameter names with IDLData objects.

IDLParameterSet - A specialized subclass of the IDLDataContainer class. This class provides the ability to associate names with contained IDLData objects.

IDLReader - The definition of the interface and the process used to construct file readers for the iTools framework. When a new file reader is constructed for the iTools system, a new class is subclassed from this IDLReader class.

IDLTool - All the functionality provided by a particular instance of an IDL Intelligent Tool (iTool). This object provides the management systems for the underlying tool functionality.

IDLUI - A link between the underlying functionality of an iTool and the IDL widget interface.

IDLVisualization - The basis for all iTool visualizations. All visualization components subclass from this class.

IDLWindow - The basis for all iTool windows. All iTool windows subclass from this class.

IDLWriter - The definition of the interface and the process used to construct file writers for the iTools framework. When a new file writer is constructed for the iTools system, a new class is subclassed from this IDLWriter class.

IDLJavaObject - An IDL object encapsulating a Java object. IDL provides data type and other translation services, allowing IDL programs to access the Java object's methods and properties using standard IDL syntax.

TrackBall - Translates widget events from a draw widget into transformations that emulate a virtual trackball (for transforming object graphics in three dimensions).

Operating System Access

APP_USER_DIR - Provides access to the application user directory.

APP_USER_DIR_QUERY - Locates existing application user directories.

CALL_EXTERNAL - Calls a function in an external sharable object.

FILE_BASENAME - Returns the basename of a file path.

FILE_CHMOD - Changes file access permissions.

FILE_DELETE - Deletes files and empty directories.

FILE_DIRNAME - Returns the dirname of a file path.

FILE_EXPAND_PATH - Fully qualifies file and directory paths.
**Functional List of IDL Routines**

**Performance Testing**

- `FILE_INFO` - Returns status information about a file.
- `FILE_MKDIR` - Creates directories.
- `FILESAME` - Determines whether two different file names refer to the same underlying file.
- `FILE_SEARCH` - Returns a string array containing the names of all files matching the input path specification.
- `FILE_TEST` - Tests a file or directory for existence and other specific attributes.
- `FILE_WHICH` - Searches for a specified file in a directory search path.
- `GET_DRIVE_LIST` (Windows only) - Returns string array of the names of valid drives/volumes for the file system.
- `GET_SCREEN_SIZE` - Returns dimensions of the screen.
- `GETENV` - Returns the value of an environment variable.
- `LINKIMAGE` - Merges routines written in other languages with IDL at run-time.
- `PATHSEP` - Returns the proper file path segment separator character for the current operating system.
- `POPD` - Removes the top directory on the working directory stack maintained by `PUSHD/POPD`.
- `PRINTD` - Prints contents of the directory stack maintained by `PUSHD/POPD`.
- `PUSHD` - Pushes a directory to top of directory stack maintained by `PUSHD/POPD`.
- `SETENV` - Adds or changes an environment variable.
- `SETUP_KEYS` - Sets function keys for UNIX versions of IDL.
- `SPAWN` - Spawns child process for access to operating system.

**Plotting**

- `AXIS` - Draws an axis of the specified type and scale.
- `BAR_PLOT` - Creates a bar graph.
- `ERRPLOT` - Plots error bars over a previously drawn plot.
- `IPLOT` - Creates an interactive tool and associated user interface (UI) configured to display and manipulate plot data.
- `OLOGIN` - Plots vector data over a previously-drawn plot.
- `OPLOTERR` - Draws error bars over a previously drawn plot.
- `PLOT` - Plots vector arguments as X versus Y graphs.
- `PLOT3DBOX` - Plots function of two variables inside a 3D box.
- `PLOT_FIELD` - Plots a 2D field using arrows.
- `PLOTER` - Plots individual data points with error bars.

**Programming and IDL Control**

- `APP_USER_DIR` - Provides access to the application user directory.
- `APP_USER_DIR_QUERY` - Locates existing application user directories.
- `ARG_PRESENT` - Returns TRUE if the value of the specified variable can be passed back to the caller.
- `BREAKPOINT` - Sets and clears breakpoints for debugging.
- `BYTEORDER` - Converts between host and network byte ordering.
- `CALL_FUNCTION` - Calls an IDL function.
- `CALL_METHOD` - Calls an IDL object method.
- `CALL_PROCEDURE` - Calls an IDL procedure.
- `CATCH` - Declares and clears exception handlers.
- `CPU` - Changes the values stored in the read-only !CPU system variable.
- `CREATE_STRUCT` - Creates and concatenates structures.
- `COMPILE_OPT` - Change default rules for compiling routines.
- `DEFINE_KEY` - Programs keyboard function keys.
- `DEFINE_MGBLK` - Defines and loads a new message block into the current IDL session.
- `DEFINE_MGBLK_FROM_FILE` - Reads the definition of a message block from a file, and loads it into the current IDL session.
- `DEF_SYSV` - Creates a new system variable.
- `EXECUTE` - Compiles, executes IDL statements contained in a string.
- `EXIT` - Exits IDL and exits back to the operating system.
- `EXPAND_PATH` - Expands path-definition string into full path name for use with the !PATH system variable.
- `HEAP_FREE` - Recursively frees all heap variables referenced by its input argument.
- `HEAP_GC` - Performs "garbage collection" on heap variables.
- `IDL_VALIDNAME` - Determines whether a string may be used as a valid IDL variable name or structure tag name.
**Functional List of IDL Routines**  

<table>
<thead>
<tr>
<th>Routine Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLITSYS_CREATETOOL</td>
<td>Creates an instance of the specified tool registered within the iTools system.</td>
</tr>
<tr>
<td>ITCURRENT</td>
<td>Set the current tool in the iTools system.</td>
</tr>
<tr>
<td>ITDELETE</td>
<td>Deletes a tool in the iTools system.</td>
</tr>
<tr>
<td>ITGETCURRENT</td>
<td>Gets the identifier of the current tool in the iTools system.</td>
</tr>
<tr>
<td>ITREGISTER</td>
<td>Registers tool object classes with the iTools system.</td>
</tr>
<tr>
<td>ITRESET</td>
<td>Resets the iTools session.</td>
</tr>
<tr>
<td>ITRESOLVE</td>
<td>Resolves all IDL code within the iTools directory, as well as all other IDL code required for the iTools framework.</td>
</tr>
<tr>
<td>KEYWORD_SET</td>
<td>Returns True if given expression is defined and nonzero or an array.</td>
</tr>
<tr>
<td>LMGR</td>
<td>Determines the type of license used by the current IDL session.</td>
</tr>
<tr>
<td>LOGICAL_AND</td>
<td>Performs a logical AND operation on its arguments.</td>
</tr>
<tr>
<td>LOGICAL_OR</td>
<td>Performs a logical OR operation on its arguments.</td>
</tr>
<tr>
<td>LOGICAL_TRUE</td>
<td>Determines whether its arguments are non-zero (or non-NULL).</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Issues error and informational messages.</td>
</tr>
<tr>
<td>N_ELEMENTS</td>
<td>Returns the number of elements contained in an expression or variable.</td>
</tr>
<tr>
<td>N_PARAMS</td>
<td>Returns the number of non-keyword parameters used in calling an IDL procedure or function.</td>
</tr>
<tr>
<td>N_TAGS</td>
<td>Returns the number of tags in a structure.</td>
</tr>
<tr>
<td>OBJ_CLASS</td>
<td>Determines the class name of an object.</td>
</tr>
<tr>
<td>OBJ_DESTROY</td>
<td>Destroys an object reference.</td>
</tr>
<tr>
<td>OBJ_ISA</td>
<td>Determines inheritance relationship of an object.</td>
</tr>
<tr>
<td>OBJ_NEW</td>
<td>Creates an object reference.</td>
</tr>
<tr>
<td>OBJ_VALID</td>
<td>Verifies validity of object references.</td>
</tr>
<tr>
<td>ON_ERROR</td>
<td>Designates the error recovery method.</td>
</tr>
<tr>
<td>ON_IOERROR</td>
<td>Declares I/O error exception handler.</td>
</tr>
<tr>
<td>PATH_CACHE</td>
<td>Controls IDL's path caching mechanism.</td>
</tr>
<tr>
<td>PTR_FREE</td>
<td>Destroys a pointer.</td>
</tr>
<tr>
<td>PTR_NEW</td>
<td>Creates a pointer.</td>
</tr>
<tr>
<td>PTR_VALID</td>
<td>Verifies the validity of pointers.</td>
</tr>
<tr>
<td>PTRARR</td>
<td>Creates an array of pointers.</td>
</tr>
<tr>
<td>RECALL_COMMANDS</td>
<td>Returns entries in IDL's command recall buffer.</td>
</tr>
<tr>
<td>REGISTER_CURSOR</td>
<td>Associates a given name with cursor information, used in conjunction with IDLgrWindow::SetCurrentCursor.</td>
</tr>
<tr>
<td>RESOLVE_ALL</td>
<td>Compiles any uncompiled routines.</td>
</tr>
<tr>
<td>RESOLVE_ROUTINE</td>
<td>Compiles a routine.</td>
</tr>
<tr>
<td>RETALL</td>
<td>Returns control to the main program level.</td>
</tr>
<tr>
<td>RETURN</td>
<td>Returns control to the next-higher program level.</td>
</tr>
<tr>
<td>ROUTINE_INFO</td>
<td>Provides information about compiled procedures and functions.</td>
</tr>
<tr>
<td>SETUP_KEYS</td>
<td>Sets function keys for UNIX versions of IDL.</td>
</tr>
<tr>
<td>STOP</td>
<td>Stops the execution of a running program or batch file.</td>
</tr>
<tr>
<td>STRMESSAGE</td>
<td>Returns the text of a given error number.</td>
</tr>
<tr>
<td>STRUCT_ASSIGN</td>
<td>Uses “Relaxed Structure Assignment” to copy structures.</td>
</tr>
<tr>
<td>STRUCT_HIDE</td>
<td>Prevents the IDL HELP procedure from displaying information about structures or objects.</td>
</tr>
<tr>
<td>SWAP_ENDIAN</td>
<td>Reverses the byte ordering of scalars, arrays or structures.</td>
</tr>
<tr>
<td>SWAP_ENDIAN_INPLACE</td>
<td>Reverses the byte ordering of scalars, arrays or structures. Differs from the SWAP_ENDIAN function in that it alters the input data in place rather than making a copy.</td>
</tr>
<tr>
<td>TAG_NAMES</td>
<td>Returns the names of tags in a structure.</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>Returns a temporary copy of a variable, and sets the original variable to “undefined”.</td>
</tr>
<tr>
<td>WAIT</td>
<td>Suspends execution of an IDL program for a specified period.</td>
</tr>
</tbody>
</table>

### Query Routines

<table>
<thead>
<tr>
<th>Routine Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY_BMP</td>
<td>Obtains information about a BMP image file.</td>
</tr>
<tr>
<td>QUERY_DICOM</td>
<td>Tests file for compatibility with READ_DICOM.</td>
</tr>
<tr>
<td>QUERY_IMAGE</td>
<td>Determines if a file is recognized as an image file.</td>
</tr>
<tr>
<td>QUERY_JPEG</td>
<td>Obtains information about a JPEG image file.</td>
</tr>
<tr>
<td>QUERY_MRSID</td>
<td>Obtains information about a MrSID image file.</td>
</tr>
<tr>
<td>QUERY_PICT</td>
<td>Obtains information about a PICT image file.</td>
</tr>
<tr>
<td>QUERY_PNG</td>
<td>Obtains information about a PNG image file.</td>
</tr>
<tr>
<td>QUERY_PPM</td>
<td>Obtains information about a PPM image file.</td>
</tr>
<tr>
<td>QUERY_SRF</td>
<td>Obtains information about an SRF image file.</td>
</tr>
<tr>
<td>QUERY_TIFF</td>
<td>Obtains information about a TIFF image file.</td>
</tr>
<tr>
<td>QUERY_WAV</td>
<td>Checks that the file is actually a .WAV file and that the READ_WAV function can read the data in the file.</td>
</tr>
</tbody>
</table>

### Saving/Restoring a Session

<table>
<thead>
<tr>
<th>Routine Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDL_Savefile</td>
<td>Object that provides complete query and restore capabilities for IDL SAVE files.</td>
</tr>
<tr>
<td>JOURNAL</td>
<td>Logs IDL commands to a file.IDL.</td>
</tr>
<tr>
<td>RESTORE</td>
<td>Restores IDL variables and routines in an IDL SAVE file.</td>
</tr>
<tr>
<td>SAVE</td>
<td>Saves variables, system variables, and IDL routines in a file for later use.</td>
</tr>
</tbody>
</table>
## Functional List of IDL Routines

### Scientific Data Formats

CDF Routines - Common Data Format routines.
EOS Routines - HDF-EOS (Hierarchical Data Format-Earth Observing System) routines.
HDF5 Routines - Hierarchical Data Format routines (version 5).
HDF Routines - Hierarchical Data Format routines.
NCDF Routines - Network Common Data Format routines.

### Scope Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOPE_LEVEL</td>
<td>Returns the current routine’s scope level.</td>
</tr>
<tr>
<td>SCOPE_VARFETCH</td>
<td>Returns variables outside the current routine’s local scope.</td>
</tr>
<tr>
<td>SCOPE_VARNAME</td>
<td>Returns the names of variables outside current routine’s local scope.</td>
</tr>
</tbody>
</table>

### Signal Processing

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_CORRELATE</td>
<td>Computes autocorrelation.</td>
</tr>
<tr>
<td>BLK_CON</td>
<td>Convolves input signal with impulse-response sequence.</td>
</tr>
<tr>
<td>C_CORRELATE</td>
<td>Computes cross correlation.</td>
</tr>
<tr>
<td>CONVOL</td>
<td>Convolves two vectors or arrays.</td>
</tr>
<tr>
<td>CORRELATE</td>
<td>Computes the linear Pearson correlation.</td>
</tr>
<tr>
<td>DIGITAL_FILTER</td>
<td>Calculates coefficients of a non-recursive, digital filter.</td>
</tr>
<tr>
<td>FFT</td>
<td>Returns the Fast Fourier Transform of an array.</td>
</tr>
<tr>
<td>HANNING</td>
<td>Creates Hanning and Hamming windows.</td>
</tr>
<tr>
<td>HILBERT</td>
<td>Constructs a Hilbert transform.</td>
</tr>
<tr>
<td>INTERPOL</td>
<td>Performs linear interpolation on vectors.</td>
</tr>
<tr>
<td>LEEFILT</td>
<td>Performs the Lee filter algorithm on an image array.</td>
</tr>
<tr>
<td>M_CORRELATE</td>
<td>Computes multiple correlation coefficient.</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>Returns median value of an array or applies a median filter.</td>
</tr>
<tr>
<td>P_CORRELATE</td>
<td>Computes partial correlation coefficient.</td>
</tr>
<tr>
<td>R_CORRELATE</td>
<td>Computes rank correlation.</td>
</tr>
<tr>
<td>SAVGOL</td>
<td>Returns coefficients of Savitzky-Golay smoothing filter.</td>
</tr>
<tr>
<td>SMOOTH</td>
<td>Smooths with a boxcar average.</td>
</tr>
<tr>
<td>TS_COEF</td>
<td>Computes the coefficients for autoregressive time-series.</td>
</tr>
<tr>
<td>TS_DIFF</td>
<td>Computes the forward differences of a time-series.</td>
</tr>
<tr>
<td>TS_FCAST</td>
<td>Computes future or past values of stationary time-series.</td>
</tr>
<tr>
<td>TS_SMOOTH</td>
<td>Computes moving averages of a time-series.</td>
</tr>
</tbody>
</table>

### Statements

### Compound Statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN...END</td>
<td>Defines a block of statements.</td>
</tr>
</tbody>
</table>

### Conditional Statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF...THEN...ELSE</td>
<td>Conditionally executes a statement or block of statements.</td>
</tr>
<tr>
<td>CASE</td>
<td>Selects one statement for execution, depending on the value of an expression.</td>
</tr>
<tr>
<td>SWITCH</td>
<td>Selects one statement for execution, depending upon the value of an expression.</td>
</tr>
</tbody>
</table>

### Loop Statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOR</td>
<td>Executes one or more statements repeatedly, incrementing or decrementing a variable with each repetition, until a condition is met.</td>
</tr>
<tr>
<td>REPEAT...UNTIL</td>
<td>Repeats statement(s) until expression evaluates to true. Subject is always executed at least once.</td>
</tr>
<tr>
<td>WHILE...DO</td>
<td>Performs statement(s) as long as expression evaluates to true. Subject is never executed if condition is initially false.</td>
</tr>
</tbody>
</table>

### Jump Statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREAK</td>
<td>Exits from a loop (FOR, WHILE, REPEAT), CASE, or SWITCH statement.</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>Starts the next iteration of the enclosing FOR, WHILE, or REPEAT loop.</td>
</tr>
<tr>
<td>GOTO</td>
<td>Transfers program control to point specified by label.</td>
</tr>
</tbody>
</table>

### Functions and Procedures

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPILE_OPT</td>
<td>Gives IDL compiler information that changes the default rules for compiling functions or procedures.</td>
</tr>
<tr>
<td>FORWARD_FUNCTION</td>
<td>Causes argument(s) to be interpreted as functions rather than variables (versions of IDL prior to 5.0 used parentheses to declare arrays).</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>Defines a function.</td>
</tr>
<tr>
<td>PRO</td>
<td>Defines a procedure.</td>
</tr>
</tbody>
</table>

### Variable Scope

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMON</td>
<td>Creates a common block.</td>
</tr>
</tbody>
</table>
String Processing

**FILE_BASENAME** - Returns the basename of a file path.

**FILE_DIRNAME** - Returns the dirname of a file path.

**STRCMP** - Compares two strings.

**STRCOMPRESS** - Removes whitespace from a string.

**STREGEX** - Performs regular expression matching.

**STRING** - Converts arguments to string type.

**STRJOIN** - Collapses a string scalar or array into merged strings.

**STRENS** - Returns the length of a string.

**STRLOWCASE** - Converts a string to lower case.

**STRMATCH** - Compares search string against input string expression.

**STRMID** - Extracts a substring from a string.

**STRPOS** - Finds first occurrence of a substring within a string.

**STRPUT** - Inserts the contents of one string into another.

**STRSPLIT** - Splits its input string argument into separate substrings, according to the specified pattern.

**STRTRIM** - Removes leading and/or trailing blanks from string.

**STRUPCASE** - Converts a string to upper case.

**Structures**

**IDL_VALIDNAME** - Determines whether a string may be used as a valid IDL variable name or structure tag name.

**REPLICATE** - Creates an array of given dimensions, filled with specified value.

**STRUCT_ASSIGN** - Uses “Relaxed Structure Assignment” to copy structures.

**STRUCT_HIDE** - Prevents the IDL HELP procedure from displaying information about structures or objects.

**Type Conversion**

**BYTE** - Converts argument to byte type.

**COMPLEX** - Converts argument to complex type.

**DCOMPLEX** - Converts argument to double-precision complex type.

**DOUBLE** - Converts argument to double-precision type.

**FIX** - Converts argument to integer type, or type specified by TYPE keyword.

**FLOAT** - Converts argument to single-precision floating-point.

**LONG** - Converts argument to longword integer type.

**LONG64** - Converts argument to 64-bit integer type.

**STRING** - Converts argument to string type.

**UINT** - Converts argument to unsigned integer type.

**ULONG** - Converts argument to unsigned longword integer type.

**ULONG64** - Converts argument to unsigned 64-bit integer type.

Utilities

**EFONT** - Interactive vector font editor and display tool.

**SLIDE_IMAGE** - Creates a scrolling graphics window for examining large images.

**XBM_EDIT** - Creates, edits bitmap icons for IDL widget button labels.

**XDISPLAYFILE** - Displays ASCII text file in scrolling text widget.

**XDFX** - Utility to display and interactively manipulate DXF objects.

**XFONT** - Creates modal widget to select and view an X Windows font.

**XINTERANIMATE** - Displays animated sequence of images.

**XDISPLAYFILE** - Displays object viewer widget.

**XOBJVIEW_ROTATE** - Programmatically rotate the object currently displayed in XOBJVIEW.

**XOBJVIEW_WRITE_IMAGE** - Write the object currently displayed in XOBJVIEW to an image file.

**XPCOLOR** - Adjusts the value of the current foreground plotting color, !P.

**XPLT3D** - Utility for creating and interactively manipulating 3D plots.

**XROI** - Utility for interactively defining and obtaining information about regions of interest.

**XVOLUME** - Utility for viewing and interactively manipulating volumes and isosurfaces.

Wavelet Toolkit

**WV_APPLET** - Runs the IDL Wavelet Toolkit GUI.

**WV_CW_WAVELET** - Compound widget used to select and display wavelet functions.

**WV_IMPORT_DATA** - Allows user to add a variable to the currently active WV_APPLET widget from the IDL> command prompt.

**WV_IMPORT_WAVELET** - Allows user to add wavelet functions to the IDL Wavelet Toolkit.

**WV_PLOT3D_WPS** - Runs the GUI for 3D visualization of the wavelet power spectrum.

**WV_PLOT_MULTIRES** - Runs the GUI for multiresolution analysis.

**WV_TOOL_DENOISE** - Runs the GUI for wavelet filtering and denoising.

Wavelet Transform

**WV_CWT** - Returns the one-dimensional continuous wavelet transform of the input array.

**WV_DENOISE** - Uses the wavelet transform to filter (or de-noise) a multi-dimensional array.

**WV_DWT** - Returns the multi-dimensional discrete wavelet transform of the input array.

**WV_PWT** - Returns the partial wavelet transform of the input vector.
Wavelet Functions

- **WV_FN_COIFLET** - Constructs wavelet coefficients for the coiflet wavelet function.
- **WV_FN_DAUBECHIES** - Constructs wavelet coefficients for the Daubechies wavelet function.
- **WV_FN_GAUSSIAN** - Constructs wavelet coefficients for the Gaussian wavelet function.
- **WV_FN_HAAR** - Constructs wavelet coefficients for the Haar wavelet function.
- **WV_FN_MORLET** - Constructs wavelet coefficients for the Morlet wavelet function.
- **WV_FN_PAUL** - Constructs wavelet coefficients for the Paul wavelet function.
- **WV_FN_SYMLET** - Constructs wavelet coefficients for the symlet wavelet function.

Widget Routines

- **WIDGET_ACTIVEX** - Create an ActiveX control and place it into an IDL widget hierarchy.
- **WIDGET_BASE** - Creates base widget (containers for other widgets).
- **WIDGET_BUTTON** - Creates button widgets.
- **WIDGET_COMBOBOX** - Creates editable droplist widgets.
- **WIDGET_CONTROL** - Realizes, manages, and destroys widgets.
- **WIDGET_DISPLAYCONTEXTMENU** - Displays a context-sensitive menu.
- **WIDGET_DRAW** - Creates drawable widgets.
- **WIDGET_DROPLIST** - Creates droplist widgets.
- **WIDGET_EVENT** - Returns events for the widget hierarchy.
- **WIDGET_INFO** - Obtains information about widgets.
- **WIDGET_LABEL** - Creates label widgets.
- **WIDGET_LIST** - Creates list widgets.
- **WIDGET_PROPERTYSMALL** - Creates a property small widget, which exposes the properties of an IDL object in a graphical interface. This widget transparently handles property value changes.
- **WIDGET_SLIDER** - Creates slider widgets.
- **WIDGET_TAB** - Creates tab widgets.
- **WIDGET_TABLE** - Creates table widgets.
- **WIDGET_TEXT** - Creates text widgets.
- **WIDGET_TREE** - Creates tree widgets.
- **XMANAGER** - Provides event loop manager for IDL widgets.
- **XMG_TMPL** - Template for creating widgets.
- **XMTool** - Displays tool for viewing XMANAGER widgets.
- **XREGISTERED** - Returns registration status of a given widget.

Widget Routines, Compound

- **CW_ANIMATE** - Creates a compound widget for animation.
- **CW_ANIMATE_GETP** - Gets pixmap window IDs used by **CW_ANIMATE**.
- **CW_ANIMATE_LOAD** - Loads images into **CW_ANIMATE**.
- **CW_ANIMATE_RUN** - Displays images loaded into **CW_ANIMATE**.
- **CW_ARCBALL** - Creates compound widget for intuitively specifying 3D orientations.
- **CW_BGROUP** - Creates button group for use as a menu.
- **CW_CLR_INDEX** - Creates compound widget for the selection of a color index.
- **CW_COLORSEL** - Creates compound widget that displays all colors in current colormap.
- **CW_DEFROI** - Creates compound widget used to define region of interest.
- **CW_FIELD** - Creates a widget data entry field.
- **CW_FILESEL** - Creates compound widget for file selection.
- **CW_FORM** - Creates compound widget for creating forms.
- **CW_FSLIDER** - Creates slider that selects floating-point values.
- **CW_LIGHT_EDITOR** - Creates compound widget to edit properties of existing IDLgrLight objects in a view.
- **CW_LIGHT_EDITOR_GET** - Gets the **CW_LIGHT_EDITOR** properties.
- **CW_PALETTE_EDITOR_SET** - Sets the **CW_PALETTE_EDITOR** properties.
- **CW_PALETTE_EDITOR** - Creates compound widget to display and edit color palettes.
- **CW_PALETTE_EDITOR_GET** - Gets the **CW_PALETTE_EDITOR** properties.
- **CW_PALETTE_EDITOR_SET** - Sets the **CW_PALETTE_EDITOR** properties.
- **CW_PDMENU** - Creates pulldown menus.
- **CW_RGBSLIDER** - Creates compound widget with sliders for adjusting RGB color values.
- **CW_ZOOM** - Creates widget for displaying zoomed images.

Window Routines

- **WDELETE** - Deletes IDL graphics windows.
- **WINDOW** - Creates window for the display of graphics or text.
- **WSET** - Selects the current window.
- **WSHOW** - Exposes or hides the designated window.
This quick reference guide contains an alphabetical listing of all IDL routines. The alphabetical listing contains all functions, procedures, statements, and objects, including the syntax of each.
IDL Syntax Conventions

Function:  
Result = FUNCTION ( Argument1 [, Argument2] [, KEYWORD1=value] [, /KEYWORD2] )

Procedure: 
PROCEDURE, Argument1 [, Argument2] [, KEYWORD1=[value1 | value2]] [, /KEYWORD2]

Statement:  
IF expression THEN statement [ ELSE statement ]

Elements of Syntax

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] (Square brackets)</td>
<td>Indicates that the contents are optional.</td>
</tr>
<tr>
<td>/ (Italicized square brackets)</td>
<td>Indicates that the square brackets are part of the statement (used to define an array).</td>
</tr>
<tr>
<td>Argument</td>
<td>Arguments are shown in italics, and must be specified in the order listed.</td>
</tr>
<tr>
<td>KEYWORD</td>
<td>Keywords are all caps, and can be specified in any order. For functions, all arguments and keywords must be contained within parentheses.</td>
</tr>
<tr>
<td>/KEYWORD</td>
<td>Indicates a boolean keyword.</td>
</tr>
<tr>
<td>Italics</td>
<td>Indicates arguments, expressions, or statements for which you must provide values.</td>
</tr>
</tbody>
</table>
| [ ] (Braces)       | - Indicates that you must choose one of the values they contain  
|                    | - Encloses a list of possible values, separated by vertical lines ( | ) 
|                    | - Encloses useful information about a keyword  
|                    | - Defines an IDL structure (this is the only case in which the braces are included in the statement).                                      |
| | (Vertical lines)  | Separates multiple values or keywords.                                                                                                       |
| [, Value1, ... , Value_n] | Indicates that any number of values can be specified.                                                                                        |
| [, Value1, ... , Value_8] | Indicates the maximum number of values that can be specified.                                                                               |

Square Brackets ([ ]) 

- Content between square brackets is optional. Pay close attention to the grouping of square brackets. Consider the following examples:

  ROUTINE_NAME, Value1 [, Value2 [, Value3]] : You must include Value1. You do not have to include Value2 or Value3. Value2 and Value3 can be specified independently.
  ROUTINE_NAME, Value1 [, Value2, Value3] : You must include Value1. You do not have to include Value2 or Value3, but you must include both Value2 and Value3, or neither.
  ROUTINE_NAME [, Value1 [, Value2]] : You can specify Value1 without specifying Value2, but if you specify Value2, you must also specify Value1.
- Do not include square brackets in your statement unless the brackets are italicized. Consider the following syntax:

```
Result = KRIG2D( Z [, X, Y] [, BOUNDS=[xmin, ymin, xmax, ymax]) )
```

An example of a valid statement is:
```
R = KRIG2D( Z, X, Y, BOUNDS=[0,0,1,1] )
```

- Note that when `[ Value1, ..., Value_n ]` is listed, you can specify any number of arguments. When an explicit number is listed, as in `[ Value1, ..., Value_8 ]`, you can specify only as many arguments as are listed.

**Braces ( { } )**

- For certain keywords, a list of the possible values is provided. This list is enclosed in braces, and the choices are separated by a vertical line ( | ). Do not include the braces in your statement. For example, consider the following syntax:

```
READ_JPEG [, TRUE={1 | 2 | 3}]
```

In this example, you must choose either 1, 2, or 3. An example of a valid statement is:
```
READ_JPEG, TRUE=1
```

- Braces are used to enclose the allowable range for a keyword value. Unless otherwise noted, ranges provided are inclusive. Consider the following syntax:

```
Result = CVTTOBM( Array [, THRESHOLD=value{0 to 255}] )
```

An example of a valid statement is:
```
Result = CVTTOBM( A, THRESHOLD=150 )
```

- Braces are also used to provide useful information about a keyword. For example:

```
[, LABEL=n{label every nth gridline}] 
```

Do not include the braces or their content in your statement.

- Certain keywords are prefaced by X, Y, or Z. Braces are used for these keywords to indicate that you must choose one of the values it contains. For example, `[(X | Y) RANGE=array]` indicates that you can specify either `XRANGE=array` or `YRANGE=array`.

- Note that in IDL, braces are used to define structures. When defining a structure, you do want to include the braces in your statement.

**Italics**

- Italicized words are arguments, expressions, or statements for which you must provide values. The value you provide can be a numerical value, such as 10, an expression, such as `DIST(100)`, or a named variable. For keywords that expect a string value, the syntax is listed as `KEYWORD=string`. The value you provide can be a string, such as ‘Hello’ (enclosed in single quotation marks), or a variable that holds a string value.

- The italicized values that must be provided for keywords are listed in the most helpful terms possible. For example, `[ XSIZE=px ]` indicates that the `XSIZE` keyword expects a value in pixels, while `[ ORIENTATION=ccw_degrees_from_horiz ]` indicates that you must provide a value in degrees, measured counter-clockwise from horizontal.
Alphabetical List of IDL Routines

Specifying Keywords

- Certain keywords are boolean, meaning they can be set to either 0 or 1. These keywords are switches used to turn an option on and off. Usually, setting such keywords equal to 1 causes the option to be turned on. Explicitly setting the keyword to 0 (or not including the keyword) turns the option off. All keywords in this reference that are preceded by a slash can be set by prefacing them by the slash. For example, SURFACE, DIST(10), /SKIRT is a shortcut for SURFACE, DIST(10), SKIRT=1. To turn the option back off, you must set the keyword equal to 0, as in SURFACE, DIST(10), SKIRT=0.

  In rare cases, a keyword’s default value is 1. In these cases, the syntax is listed as KEYWORD=0, as in SLIDE_IMAGE [, Image] [, CONGRID=0]. In this example, CONGRID is set to 1 by default. If you specify CONGRID=0, you can turn it back on by specifying either /CONGRID or CONGRID=1.

- Some keywords are used to obtain values that can be used upon return from the function or procedure. These keywords are listed as KEYWORD=variable. Any valid variable name can be used for these keywords, and the variable does not need to be defined first. Note, however that when a keyword calls for a named variable, only a named variable can be used—sending an expression causes an error.

  For example, the WIDGET_CONTROL procedure can return the user values of widgets in a named variable using the GET_UVALUE keyword. To return the user value for a widget ID (contained in the variable mywidget) in the variable userval, you would use the command:

  WIDGET_CONTROL, mywidget, GET_UVALUE = userval

  Upon return from the procedure, userval contains the user value. Note that userval did not have to be defined before the call to WIDGET_CONTROL.

- Some routines have keywords that are mutually exclusive, meaning only one of the keywords can be present in a given statement. These keywords are grouped together, and separated by a vertical line. For example, consider the following syntax:

  PLOT, [X], Y [, /DATA | /DEVICE | /NORMAL]

  In this example, you can choose either DATA, DEVICE, or NORMAL, but not more than one. An example of a valid statement is:

  PLOT, SIN(A), /DEVICE

- Keywords can be abbreviated to their shortest unique length. For example, the XSTYLE keyword can be abbreviated to XST because there are no other keywords in IDL that begin with XST. You cannot shorten XSTYLE to XS, however, because there are other keywords that begin with XS, such as XSIZE.
Alphabetical Listing

The following alphabetical listing contains all IDL functions, procedures, and statements included in IDL version 6.1.

A

A_CORRELATE - Computes autocorrelation.
*Result = A_CORRELATE(X, Lag [, /COVARIANCE] [, /DOUBLE] )*

ABS - Returns the absolute value of X.
*Result = ABS(X [, Thread pool keywords])*

ACOS - Returns the arc-cosine of X.
*Result = ACOS(X [, Thread pool keywords])*

ADAPT_HIST_EQUAL - Performs adaptive histogram equalization.

ALOG - Returns the natural logarithm of X.
*Result = ALOG(X [, Thread pool keywords])*

ALOG10 - Returns the logarithm to the base 10 of X.
*Result = ALOG10(X [, Thread pool keywords])*

AMOEBA - Minimizes a function using downhill simplex method.
*Result = AMOEBA(Ftol [, FUNCTION_NAME=string] [, FUNCTION_VALUE=variable] [, NCALLS=value] [, NMAX=value] [, P0=vector, SCALE=vector [, SIMPLEX=array] )*

ANNOTATE - Starts IDL widget used to interactively annotate images and plots with text and drawings.

APP_USER_DIR - Provides access to the application user directory.

APP_USER_DIR_QUERY - Allows searches for application user directories.
*Result = APP_USER_DIR_QUERY(AuthorDirname, AppDirname [, COUNT=variable] [, /EXCLUDE_CURRENT] [RESTRICT keywords] [QUERY keywords] )*

ARG_PRESENT - Returns TRUE if the value of the specified variable can be passed back to the caller.
*Result = ARG_PRESENT(Variable)*

ARRAY_EQUAL - Provides a fast way to compare data for equality in situations where the index of the elements that differ are not of interest.
*Result = ARRAY_EQUAL(Op1 , Op2 [, /NO_TYPECONV ] )*

ARRAY_INDICES - Converts one-dimensional subscripts of an array into corresponding multi-dimensional subscripts.
*Result = ARRAY_INDICES(Array, Index)*

ARROW - Draws line with an arrow head.
*ARROW, X0, Y0, X1, Y1 [, /DATA [, /NORMALIZED] [, HSIZE=length] [, COLOR=index] [, THICK=value] [, /SOLID] [, THICK=value] *

ASCII_TEMPLATE - Presents a GUI that generates a template defining an ASCII file format.

ASIN - Returns the arc-sine of X.
*Result = ASIN(X [, Thread pool keywords])*

ASSOC - Associates an array structure with a file.
*Result = ASSOC(Unit, Array Structure [, Offset] [, /PACKED] )*

ATAN - Returns the arc-tangent of X.
*Result = ATAN(X [, /PHASE] [, Thread pool keywords])*
or
*Result = ATAN(Y, X) [, Thread pool keywords]*)
Alphabetical List of IDL Routines

**AXIS** - Draws an axis of the specified type and scale.

```
AXIS, X=[Y; Z]; ZSAVE=[XAXIS=[0; 1]; YAXIS=[0; 1]; ZAXIS=[0; 1; 2; 3]]; XLOG=[/YNOZERO]; /YLOG; /ZLOG
```

**Graphics Keywords**:
- [CHARSIZE=value
  - CHARTHICK=[integer
  - COLOR=value
  - /DATA
  - /DEVICE
  - /NORMAL
  - FONT=integer
  - /NODATA
  - /NOERASE
  - SUBTITLE=string
  - /T3D
  - /OVERPLOT
  - /ROTA TE
  - TITLE=string
  - /T3D
  - /NORMAL
  - FONT=integer
  - BACKGROUND=Va lue
  - BAR_OFFSET=integer
  - BAR_WIDTH=integer
  - BASELINES=vector
  - BASE_RANGE=scalar
  - /OVERRIDE
  - /OVERRIDE
  - /ROTA TE
  - TITLE=string
  - /TITLE=string

**BEGIN...END** - Defines a block of statements.

```
BEGIN statements END | ENDS | ENDELSE | ENDFOR | ENDRP | ENDWHILE
```

**BESELI** - Returns the I Bessel function of order $N$ for $X$.

```
Result = BESELI(X, N [, /DOUBLE] [, ITER=variable])
```

**BESELY** - Returns the Y Bessel function of order $N$ for $X$.

```
Result = BESELY(X, N [, /DOUBLE] [, ITER=variable])
```

**BESELY** - Returns the Y Bessel function of order $N$ for $X$.

```
Result = BESELY(Y, N [, /DOUBLE] [, ITER=variable])
```

**BETA** - Returns the value of the beta function.

```
Result = BETA( Z, W [, /DOUBLE] )
```

**BILINEAR** - Computes array using bilinear interpolation.

```
Result = BILINEAR(P, IX, JY [, MISSING=value] )
```

**BIN_DATE** - Converts ASCII date/time string to binary string.

```
Result = BIN_DATE(ASCII_Time)
```

**BINARY_TEMPLATE** - Presents a GUI for interactively generating a template structure for use with READ_BINARY.

```
Template = BINARY TEMPLATE ((Filename)
[, CANCEL=variable] [, GROUP=widget_id]
[, N_ROWS=rows] [, TEMPLATE=value])
```

**BINDGEN** - Returns byte array with each element set to its subscript.

```
Result = BINDGEN(D1 [, ...D8] [, Thread pool keywords])
```

**BINOMIAL** - Computes binomial distribution function.

```
Result = BINOMIAL(V, N, P [, /DOUBLE] [, /GAUSSIAN] )
```

**BLAS_AXPY** - Updates existing array by adding a multiple of another array.

```
BLAS_AXPY, Y, A, X [, D1, Loc1, D2, Range]]
```

**BLK_CON** - Convolves input signal with impulse-response sequence.

```
Result = BLK_CON( Filter, Signal [, B_LENGTH=scalar]
[, /DOUBLE] )
```

**BOX_CURSOR** - Emulates the operation of a variable-sized box cursor.

```
BOX_CURSOR, [ X0, Y0, NX, NY ] [, INIT]
[, /FIXED_SIZE] [ , /MESSAGE]
```

**BREAK** - Immediately exits from a loop (FOR, WHILE, REPEAT), CASE, or SWI TCH statement.

```
BREAK
```

**BREAKPOINT** - Sets and clears breakpoints for debugging.

```
BREAKPOINT [, File, Index [, AFTER=integer]
[, CLEAR=variable] [, CONDITION=expression]
[, /DISABLE]
```

**BROYDEN** - Solves nonlinear equations using Broyden's method.

```
Result = BROYDEN( X, Vecfunc [, CHECK=variable]
[, /DOUBLE] [, EPS=value] [, ITPMAX=value]
[, STEPMAX=value] [, TOL=value]
[, TOLMIN=value] [, TOLX=value])
```

**BYTE** - Creates a byte vector or array.

```
Result = BYTE( Expression [, Offset [, D1 [, ...D8] ]] [, /NOZERO] )
```

**BYTE** - Converts argument to byte type.

```
Result = BYTE(Expression [, Offset [, D1 [, ...D8] ]]
[, Thread pool keywords])
```
Alphabetical List of IDL Routines

BYTORDER - Converts between host and network byte ordering.
BYTORDER, Variable_1, ..., Variable_n [, /DUTOVAX]
[, /NTOSH] [, /SSWAP] [, /SWAP_IF_BIG_ENDIAN]
[, /SWAP_IF_LITTLE_ENDIAN] [, /VAXTOD]
keywords]

BYTSCl - Scales all values of an array into range of bytes.
Result = BYTSCl( Array [, MAX=value] [, MIN=value]
[, /NAN] [, TOP=value] [, Thread pool
keywords])

CASE - Selects one statement for execution, depending on the value of
an expression.
CASE expression OF
expression: statement
... expression: statement
[ ELSE: statement ]
ENDCASE

CATCH - Declares and clears exception handlers.
CATCH, [Variable] [, /CANCEL]

CD - Sets and/or changes the current working directory.
CD [, Directory] [, CURRENT=variable]

CDF_* Routines - See “CDF Routines” on page 124.

CEIL - Returns the closest integer greater than or equal to X.
Result = CEIL( X [, /L64] [, Thread pool
keywords])

CHEBYSHEV - Returns the forward or reverse Chebyshev polynomial
expansion.
Result = CHEBYSHEV(D, N)

CHECK_MATH - Returns and clears accumulated math error status.
Result = CHECK_MATH( [, MASK=bitmask]
[ , /NOCLEAR] [, PRINT])

CHISQR_CVF - Computes cutoff value in a Chi-square distribution.
Result = CHISQR_CVF(P, Df)

CHISQR_PDF - Computes Chi-square distribution function.
Result = CHISQR_PDF(V, Df)

CHOLDC - Constructs Cholesky decomposition of a matrix.
CHOLDC, A, P [, /DOUBLE]

CHOLSOL - Solves set of linear equations (use with CHOLDC).
Result = CHOLSOL(A, P, B [, /DOUBLE])

CINDGEN - Returns a complex array with each element set to its sub-
script.
Result = CINDGEN(D_1 [, ..., D_n] [, Thread pool
keywords])

CIR_3PNT - Returns radius and center of circle, given 3 points.
CIR_3PNT, X, Y, R, X0, Y0

CLOSE - Closes the specified files.
CLOSE, Unit_1, ..., Unit_n [, IALL]
[, EXIT_STATUS=variable] [, FILE] [, /FORCE]

CLUST_WTS - Computes the cluster weights of an array for cluster
analysis.
Result = CLUST_WTS( Array [, /DOUBLE]
[ , N_CLUSTERS=value] [, N_ITERATIONS=integer]
[, VARIABLE_WTS=vector] )

CLUSTER - Performs cluster analysis.
Result = CLUSTER( Array, Weights [, /DOUBLE]
[ , N_CLUSTERS=value] )
CLUSTER_TREE - Computes the hierarchical clustering for a set of m items in an n-dimensional space.
Result = CLUSTER_TREE( Pairdistance, Linkdistance [, LINKAGE = value] )
or for LINKAGE = 3 (centroid):
Result = CLUSTER_TREE( Pairdistance, Linkdistance, LINKAGE = 3, DATA = array[, MEASURE=value] [, POWER_MEASURE=value] )

COLOR_CONVERT - Converts color triples to and from RGB, HLS, and HSV.
COLOR_CONVERT, I0, I1, I2, O0, O1, O2 [, /HLS_RGB | /RGB_HLS | /RGB_HSV]

COLOR_QUAN - Computes the hierarchical clustering for a set of m items in an n-dimensional space.
Result = COLOR_QUAN( Pairdistance, Linkdistance [, LINKAGE = value] )
or for LINKAGE = 3 (centroid):
Result = COLOR_QUAN( Pairdistance, Linkdistance, LINKAGE = 3, DATA = array[, MEASURE=value] [, POWER_MEASURE=value] )

COMPLEX - Converts argument to complex type.
Result = COMPLEX( Expression )
Result = COMPLEX( Offset, D1 [, ... , Dn] [, /DOUBLE] [, Thread pool keywords] )

COMPLEXARR - Creates a complex, single-precision, floating-point vector or array.
Result = COMPLEXARR( D1 [, ... , Dn] [, /NOZERO] )

COMPLEXROUND - Rounds a complex array.
Result = COMPLEXROUND( Input )

COLOR_CONVERT - Converts color triples to and from RGB, HLS, and HSV.
COLOR_CONVERT, I0, I1, I2, O0, O1, O2 [, /HLS_RGB | /RGB_HLS | /RGB_HSV]

COMFIT - Fits paired data using one of six common filtering functions.
Result = COMFIT( X, Y [, A[, /EXPONENTIAL | /GEOMETRIC | /GOMPERTZ | /HYPERBOLIC | /LOGISTIC | /LOGSQUARE | /POWER_MEASURE=value] [, /XLOG | /YLOG] )

CONJ - Returns the complex conjugate of X.
Result = CONJ( X [, Thread pool keywords] )

CONVOL - Converts argument to complex type.
Result = CONVOL( Expression )
Result = CONVOL( Offset, D1 [, ... , Dn] [, /DOUBLE] [, /TO_DATA | /TO_DEVICE | /TO_NORMAL] )

CONSTRAINED_MIN - Minimizes a function using Generalized Reduced Gradient Method.

CONTINUE - Immediately starts the next iteration of the enclosing FOR, WHILE, or REPEAT loop.
CONTINUE

CONTOUR - Draws a contour plot.

Graphics Keywords: Accepts all graphics keywords accepted by PLOT except for: LINESTYLE, PSYM, SYMSIZE.
CTI_TEST - Performs chi-square goodness-of-fit test.

Result = CTI_TEST( Obsfreq [, COEFF=variable] [/CORRECTED] [ , CRAMN=variable] [ , DF=variable] [ , EXPFREQ=variable] [ , RESIDUAL=variable] )

CURSOR - Reads position of the interactive graphics cursor.


CURVEFIT - Fits multivariate data with a user-supplied function.


CV_COORD - Converts 2D and 3D coordinates between coordinate systems.


CVTTOBM - Creates a bitmap byte array for a button label.

Result = CVTTOBM( Array [, THRESHOLD=value] )

CW_ANIMATE - Creates a compound widget for animation.


CW_ANIMATE_GETP - Gets pixmap window IDs used by CW_ANIMATE.

Result = CW_ANIMATE_GETP, Widget, Pixmaps [, /KILL_ANYWAY]
CW_BGROUP - Creates button group for use as a menu.

\[ Result = CW_BGROUP( Parent, Names \) \]

\[-\] Creates a widget data entry field.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]

\[-\] Creates compound widget for file selection.

\[ Result = CW_FILESEL( Parent [, FILENAME \) \]
Alphabetical List of IDL Routines

**CW_LIGHT_EDITOR_SET** - Sets the CW_LIGHT_EDITOR properties.

```idl
Result = CW_LIGHT_EDITOR_SET(WidgetID[, /DIRECTION_DISABLED], [ /DRAG_EVENTS], [ /HIDE_DISABLED], [ /LOCATION_DISABLED], [ /TYPE_DISABLED], [ XSIZE=vector], [YSIZE=vector], [XOFFSET=vector], [YOFFSET=vector], [SCALE=value], [TAB_MODE=value], [/RETURN_INDEX], [/RETURN_ID], [/TRACK], [UNAME=string], [UNAME=string], [tüm=ocale|STRING], [ /VERTICAL])
```

**CW_Palette_EDITOR** - Creates compound widget used to interactively adjust the 3D drawing transformation.

```idl
Result = CW_PALETTE_EDITOR(WidgetID[, AX=degrees], [AX=degrees], [ /FRAME], [ /HORIZONTAL], [TITLE=string], [UNAME=string], [UVALUE=value], [XSIZE=width], [YSIZE=height])
```

**CW_Palette_EDITOR_GET** - Gets the CW_PALETTE_EDITOR properties.

```idl
Result = CW_PALETTE_EDITOR_GET(WidgetID[, ALPHA=variable], [ /HISTOGRAM=variable])
```

**CW_Palette_EDITOR_SET** - Sets the CW_PALETTE_EDITOR properties.

```idl
Result = CW_PALETTE_EDITOR_SET(WidgetID[, ALPHA=byte_vector], [ /HISTOGRAM=byte_vector])
```

**CW_RGBSLIDER** - Creates compound widget with sliders for adjusting RGB color values.

```idl
Result = CW_RGBSLIDER(WidgetID[, /CMY], [ /HSV], [ /HLS], [ /RGB], [ /COLOR_INDEX], [GRAPHICS_LEVEL=1|2], [ /DRAG], [ /FRAME], [LENGTH=value], [ /RGB], [TAB_MODE=value], [UNAME=string], [UVALUE=value], [VALUE=r, g, b])
```

**CW_TMPL** - Template for compound widgets that use XMANAGER.

```idl
Result = CW_TMPL(WidgetID[, TAB_MODE=value], [UNAME=string], [UVALUE=value])
```

**CW_ZOOM** - Creates widget for displaying zoomed images.

```idl
Result = CW_ZOOM(WidgetID[, /FRAME], [MAX=value], [MIN=value], [RETAIHE=0], [2], [SAMPLE=value], [SCALE=value], [TAB_MODE=value], [TRACK], [UNAME=string], [UVALUE=value], [XSIZE=width], [X_SCROLL_SIZE=width], [YSIZE=height], [Y_SCROLL_SIZE=height], [Y_SIZE=zoom_height])
```

**D**

**DBLARR** - Creates a double-precision array.

```idl
Result = DBLARR(D [, /NOZERO])
```

**DCINDGEN** - Returns a double-precision, complex array with each element set to its subscript.

```idl
Result = DCINDGEN(D [, /NOZERO])
```

**DCOMPLEX** - Converts argument to double-precision complex type.

```idl
Result = DCOMPLEX(Real [, Imaginary], [ /NOZERO])
```

**DCOMPLEXARR** - Creates a complex, double-precision vector or array.

```idl
Result = DCOMPLEXARR(D [, /NOZERO])
```

**DEFINE_KEY** - Programs keyboard function keys.

```idl
DEFINE_KEY(Key [, Value], [ /MATCH_PREVIOUS], [ /NOECHO], [ /TERMINATE])
```

**UNIX Keywords**:

- `/BACK_CHARACTER`
- `/BACKWORD`
- `/CONTROL`
- `/DELETE_CHARACTER`
- `/DELETE_CURRENT`
- `/DELETE_EOL`
- `/DELETE_LINE`
- `/END_FILE`
- `/END_OF_LINE`
- `/ENTER_LINE`
- `/FORWARD_CHARACTER`
- `/FORWARD_WORD`
- `/INSERT_OVERSTRIKE_TOGGLE`
- `/NEXT_LINE`
- `/PREVIOUS_LINE`
- `/RECALL`
- `/REDRAW`
- `/START_OF_LINE`

**DEFINE_MSGBLK** - Defines and loads a new message block into the current IDL session.

```idl
DEFINE_MSGBLK(BlockName, [ErrorNames, ErrorFormats[, /IGNORE_DUPLICATE]], [PREFIX=PrefixStr])
```

**DEFINE_MSGBLK_FROM_FILE** - Reads the definition of a message block from a file, and loads it into the current IDL session.

```idl
DEFINE_MSGBLK_FROM_FILE(Filename[, BlockName, [ /IGNORE_DUPLICATE]], [PREFIX=PrefixStr[, /VERBOUSE]])
```
DEFROI - Defines an irregular region of interest of an image.

RESULT = DEFROI(Xs, Ys [, Xverts, Yverts] [, /NOREGION] [, /NOFILL] [, /RESTORE] [, XD=device_coord, YD=device_coord] [, /ZOOM=factor])

DEFSYSV - Creates a new system variable.

DEFSYSV , Name , Value [, Read_Only]

[ , EXITS=variable]

DELVAR - Deletes variables from the main IDL program level.

DELVAR, V1, ... , Vn

DENDRO_PLOT - Draws a two-dimensional dendrite plot on the current direct graphics device if given a hierarchical tree cluster, as created by CLUSTER_TREE.

DENDRO_PLOT, Cluster, Linkdistance

[ , LABEL_CHARSIZE=value] [ , LABEL_COLOR=value] [ , LABEL_COLOR=value] [ , LABEL_COLOR=value] [ , LABEL_ORIENTATION=value]

[ , LINECOLOR=value] [ , LINESTYLE=value] [ , ORIENTATION=value] [ , /OVERPLOT]

DENDROGRAM - Constructs a dendrogram and returns a set of vertices and connectivity that can be used to visualize the dendrite plot if given a hierarchical tree cluster, as created by CLUSTER_TREE.

DENDROGRAM, Cluster, Linkdistance, Outverts, Outconn [, LEAFNODULES=variable]

DERIV - Performs differentiation using 3-point, Lagrangian interpolation and returns the derivative.

RESULT = DERIV([X, Y])

DERIVSIG - Computes standard deviation of derivative found by DERIV.

RESULT = DERIVSIG([X, Y, SIGx, SIGy])

DETERM - Computes the determinant of a square matrix.

RESULT = DETERM(A [, /CHECK] [, /DOUBLE] [, ZERO=value])

DEVICE - Sets to plot in device coordinates.

Note: Each keyword to DEVICE is followed by the device(s) to which it applies.


[ , /BYPASS_TRANSLATION/[WIN, X]] [, /CLOSE/[Z]] [, /CLOSE_DOCUMENT/[PRINTER]] [, /CLOSE_FILE/[CGM, HP, METAFILE, PCL, PS, REGIS, TEK]] [, /CMYK [PS]] [, /COLOR[PCL, PS]] [, /COLORS=value[CGM, TEK]] [, /COPY=/Xsource, Ysource, cols, rows, Xdest, Ydest, /WINDOW_INDEX] [WIN, X]] [, /CROSSHAIR/[WIN, X]] [, /CURSOR_IMAGE=value[16-element short int vector] [WIN, X]] [, /CURSOR_MASK=value[WIN, X]] [, /CURSOR_ORIGINAL/[WIN, X]] [, /CURSOR_STANDARD=value[WIN, arrow=32512, I-beam=32513, hourglass=32514, black cross=32515, up arrow=32516, size(NT)=32640, icon(NT)=32641, size NW-SE=32642, size NE-SW=32643, size E-W=32644, size N-S=32645] [X: one of the values in file cursorfonts.h]] [, /CURSOR_XY=[x,y][WIN, X]]

[ , /DECOMPOSED/[WIN, X]] [, /DIRECT_COLOR/[X]] [, /EJECT=[0 1 2][HP]] [, /ENCAPSULATED=[0 1][PS]] [, /ENCODING=1 (binary) 2 (text) 3 (NCAR binary)] [CMGM]

[, FILENAME=filename[CGM, HP, METAFILE, PCL, PS, REGIS, TEK]] [, /FLOYD/[PCL, X]]

[, /FONT_INDEX=integer[PS]] [, /FONT_SIZE=points[PS]]

[, /GET_CURRENT_FONT=variable[METAFILE, PRINTER, WIN, X]]

[, /GET_DECOMPOSED=variable[WIN, X]]

[, /GET_FONTNAMES=variable[PRINTER]]; [, /INDEX_COLOR=variable[PRINTER, X], [, /ITALIC], [, /LANDSCAPE], [, /LIGHT], [, /MEDIUM], [, /NARROW], [, /OBLIQUE], [, /PSEUDO_COLOR], [, /POLYFILL], [, /PREVIEW], [, /PRINT_FILE=filename], [, /PRE_XSIZE], [, /PRE_DEPTH], [, /PRINTSTYLE], [, /PRINTFILE], [, /PRINTSTYLE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [, /PRINTFILE], [. }
**DEVICE, continued**

`[ , SET_FONT=string [METAFILE, PRINTER, PS, WIN, Z] [ , SET_GRAPHICS_FUNCTION=code [0 to 15] [WIN, X, Z] [ , SET_RESOLUTION=length [width, height] [Z] ] [ , SET_STRING=string [TEK] ] [ , SET_TRANSLATION=variable [X] ] [ , SET_WRITE_MASK=value [0 to 2^n-1 for n-bit system] [X, Z] ] [ , STATIC_COLOR=byte [bits per pixel] [X] ] [ , STATIC_GRAY=byte [bits per pixel] [X] ] [ , Threshold=value [byte [bits per pixel] [X] ] [ , TRANSLATION=variable [WIN, X] ] [ , TRUE_COLOR=byte [bits per pixel] [METAFILE, PRINTER, X] [ , TT_FONT=variable [METAFILE, PRINTER, WIN, X, Z] [ , TTY=variable [REGIS, TEK] ] [ , VT240] [ , /VT340] [ , /VT341] [ , WINDOW_STATE=variable [WIN, X] ] [ , XOFFSET=value [pixels] ] [ , XSIZE=value [pixels] ] [ , XY=value ] [ , YOFFSET=value [pixels] ] [ , ZBUFFERING={0 | 1 (default)} [Z] ] [ , WINDOW_STATE=variable [WIN, X] ] [ , XOFFSET=value [pixels] ] [ , XSIZE=value [pixels] ] [ , XY=value ] [ , YOFFSET=value [pixels] ] [ , ZBUFFERING={0 | 1 (default)} [Z] ]

**DFPMIN**  
Minimizes a function using Davidon-Fletcher-Powell method.

`DFPMIN, X, Gtol, Fmin, Fhigh, Func, Dfunc`

`[ , EPS=value ] [ , ITER=variable ] [ , ITMAX=value ] [ , TOLX=value ] [ , EPS=value ] [ , ITER=variable ] [ , ITMAX=value ] [ , TOLX=value ]`

**DIAG_MATRIX**  
Constructs a diagonal matrix from an input vector, or if given a matrix, extracts a diagonal vector.

`Result = DIAG_MATRIX(A [, Diag])`

**DIALOG_MESSAGE**  
Creates modal message dialog.


**DIALOG_PICKFILE**  
Creates native file-selection dialog.


**DIALOG_PRINTERSETUP**  
Opens native dialog used to set properties for a printer.


**DIALOG_PRINTJOB**  
Opens native dialog used to set parameters for a print job.


**DIALOG_READ_IMAGE**  
Presents GUI for reading image files.


**DIALOG_WRITE_IMAGE**  
Presents GUI for writing image files.


**DIGITAL_FILTER**  
Calculates coefficients of a non-recursive, digital filter.

`Result = DIGITAL_FILTER( Flow, Fhigh, A, Nterms [, /DOUBLE] )`

**DILATE**  
Implements morphologic dilation operator on binary and grayscale images.


**DINDGEN**  
Returns a double-precision array with each element set to its subscript.

`Result = DINDGEN(D1 [, ..., DN] [ , Thread_pool_keywords] )`

**DISSOLVE**  
Provides a digital “dissolve” effect for images.


**DIST**  
Creates array with each element proportional to its frequency.

`Result = DIST(N [, M] )`

**DISTANCE_MEASURE**  
Computes the pairwise distance between a set of items or observations.

DLM_LOAD - Explicitly causes a DLM to be loaded.
DLM_LOAD, DLMNameStr
[ , DLMNameStr,..., DLMNameStr_n]

DOC_LIBRARY - Extracts documentation headers from IDL programs.
DOC_LIBRARY [ , Name] [ , /PRINT]
UNIX keywords: [ , DIRECTORY=string] [ , /MULTI]

DOUBLE - Converts argument to double-precision type.
Result = DOUBLE( Expression [ , Offset [, D1 [, ... D9]]]
[ , Thread pool keywords])

DRAW_ROI - Draws region or group of regions to current Direct Graphics device.
DRAW_ROI, oROI [ , LINE_FILL] [ , SPACING=value]
Graphics Keywords: [ , CLIP=Xp, Yp, Xf, Yf]
[ , COLOR=value] [ , /DATA] [ , /DEVICE] [ , /NORMAL]
[ , LINESTYLE={0 | 1 | 2 | 3 | 4 | 5}] [ , /NOCLIP]
[ , ORIENTATIoN=nccw_degrees_from_horiz]
[ , PSYM={integer{0 to 10}}] [ , SYMSIZE=value] [ , /T3D]
[ , THICK=value]

ERODE - Implements the erosion operator on binary and grayscale images and vectors.
Result = ERODE( Image, Structure [ , X0 [, Y0 [, Z0]]]
[ , /GRAY] [ , /RESERVE_TYPE_] [ , /UINT] [ , /ULONG]
[ , VALUES=varray] )

ERF - Returns the value of an error function.
Result = ERF(Z [ , Thread pool keywords])

ERFC - Returns the value of a complementary error function.
Result = ERFC(Z [ , Thread pool keywords])

ERFCX - Returns the value of a scaled complementary error function.
Result = ERFCX(Z [ , Thread pool keywords])

ERRPLOT - Plots error bars over a previously drawn plot.
ERRPLOT, [ X [, Low, High [, WIDTH=value]]]

EXECUTE - Compiles and executes IDL statements contained in a string.
Result = EXECUTE(String [ , QuietCompile]
[ , QuietExecution])

EXIT - Quits IDL and exits back to the operating system.
EXIT [ , /NO_CONFIRM] [ , STATUS=code]

EXP - Returns the natural exponential function of Expression.
Result = EXP(Expression [ , Thread pool keywords])

EXPAND - Shrinks/expands images using bilinear interpolation.
EXPAND, A, Nx, Ny, Result [ , FILLVAL=value]
[ , MAXVAL=value]

EXPAND_PATH - Expands path-definition string into full path name for use with the !PATH system variable.
Result = EXPAND_PATH(String [ , /ALL_DIRS]
[ , /ARRAY] [ , COUNT=variable] [ , /DLIM] [ , /HELP])

EXPINT - Returns the value of the exponential integral.
Result = EXPINT( N, X [, /DOUBLE] [ , EPS=value]
[ , /ITER=variable] [ , ITMAX=value] [ , Thread pool keywords])

EXTRAC - Returns sub-matrix of input array. Array operators (e.g., *
and :) should usually be used instead.
Result = EXTRAC(Array, C1, C2, ..., Cn, S1, S2, ..., Sn)

EXTRACT_SLICE - Returns 2D planar slice extracted from volume.
Result = EXTRACT_SLICE(Vol, Xsize, Ysize, Zsize, Xcenter,
Ycenter, Zcenter, Xrot, Yrot, Zrot
[ , ANISOTROPY={xspacing, yspacing, zspacing]}
[ , /CUBIC] [ , OUT_VAL=value] [ , /RADIANS]
[ , /SAMPLE] [ , VERTICES=variable] )
or
Result = EXTRACT_SLICE(Vol, Xsize, Ysize, Zsize, Xcenter,
Ycenter, Zcenter, PlaneNormal, Xvec
[ , ANISOTROPY={xspacing, yspacing, zspacing]}
[ , /CUBIC] [ , OUT_VAL=value] [ , /RADIANS]
[ , /SAMPLE] [ , VERTICES=variable] )
### Alphabetical List of IDL Routines

#### F

**F_CVF** - Computes the cutoff value in an F distribution.

\[
\text{Result} = \text{F}_\text{CVF}(P, \text{Dfnum}, \text{Dfd})
\]

**F_PDF** - Computes the F distribution function.

\[
\text{Result} = \text{F}_\text{PDF}(V, \text{Dfnum}, \text{Dfd})
\]

**FACTORIAL** - Computes the factorial \( N \! \).  

\[
\text{Result} = \text{FACTORIAL}(N, [\text{/STIRLING} [/, \text{UL64}])
\]

**FFT** - Returns the Fast Fourier Transform of Array.

\[
\text{Result} = \text{FFT}(\text{Array}, [\text{Direction}], [\text{DIMENSION}=\text{vector}]
\]

**FILE_BASENAME** - Returns the basename of a file path.

\[
\text{Result} = \text{FILE_BASENAME}([\text{Path}], [\text{RemoveSuffix}])
\]

**FILE_CHMOD** - Changes the current access permissions (or modes) associated with a file or directory.

\[
\text{Result} = \text{FILE_CHMOD}([\text{File}], [\text{Mode}]
\]

UNIX-Only Keywords: [/, \text{FILE_EXECUTE} | /\text{FILE_READ} | /\text{FILE_WRITE}]

**FILE_COPY** - Copies files or directories to a new location.

\[
\text{Result} = \text{FILE_COPY}([\text{SourcePath}], [\text{DestPath}], [\text{/ALLOW SAME}],
\]

\[
[\text{/NOEXPAND_PATH}], [\text{/OVERWRITE}], [\text{/RECURSIVE}]
\]

UNIX-Only Keywords: [/, \text{COPY_NAMED_PIPE}], [/, \text{COPY_SYMLINK}], [/, \text{FORCE}]

**FILE_DELETE** - Deletes a file or empty directory, if the process has the necessary permissions to remove the file as defined by the current operating system.

\[
\text{Result} = \text{FILE_DELETE}([\text{File}])
\]

UNIX-Only Keywords: [/, \text{ALLOW NONEXISTENT}], [/, \text{ALLOW SAME}]

**FILE_DIRNAME** - Returns the dirname of a file path.

\[
\text{Result} = \text{FILE_DIRNAME}([\text{Path}]
\]

**FILE_EXPAND_PATH** - Expands a given file or partial directory name to its fully qualified name regardless of the current working directory.

\[
\text{Result} = \text{FILE_EXPAND_PATH}([\text{Path}]
\]

**FILE_INFO** - Returns status information about a file.

\[
\text{Result} = \text{FILE_INFO}([\text{Path}], [\text{/NOEXPAND_PATH}])
\]

**FILE_LINES** - Returns the number of lines of text in a file.

\[
\text{Result} = \text{FILE_LINES}([\text{Path}], [\text{/COMPRESS}], [\text{/NOEXPAND_PATH}])
\]

**FILE_LINK** - Creates UNIX file links.

\[
\text{Result} = \text{FILE_LINK}([\text{SourcePath}], [\text{DestPath}], [\text{/ALLOW SAME}],
\]

\[
[\text{/HARDLINK}], [\text{/NOEXPAND_PATH}], [\text{/VERBOSE}]
\]

**FILE_MKDIR** - Creates a new directory, or directories, with default access permissions for the current process.

\[
\text{Result} = \text{FILE_MKDIR}([\text{File1}], [\text{...}], [\text{FileN}], [\text{/NOEXPAND_PATH}])
\]

**FILE_MOVE** - Renames files and directories.

\[
\text{Result} = \text{FILE_MOVE}([\text{SourcePath}], [\text{DestPath}], [\text{/ALLOW SAME}],
\]

\[
[\text{/NOEXPAND_PATH}], [\text{/OVERWRITE}], [\text{/RECURSIVE}]
\]

UNIX-Only Keywords: [/, \text{MOVE_NAMED_PIPE}]

**FILE_READLINK** - Returns the path pointed to by a UNIX symbolic link.

\[
\text{Result} = \text{FILE_READLINK}([\text{Path}], [\text{/ALLOW NONSYMLINK}],
\]

\[
[\text{/NOEXPAND_PATH}]
\]

**FILESAME** - Determines whether two different file names refer to the same underlying file.

\[
\text{Result} = \text{FILESAME}([\text{Path1}], [\text{Path2}],
\]

\[
[\text{/NOEXPAND_PATH}]
\]

**FILE_SEARCH** - Returns a string array containing the names of all files matching the input path specification.

\[
\text{Result} = \text{FILE_SEARCH}([\text{Path Specification}],
\]

\[
[\text{Recursion}]
\]

UNIX-Only Keywords: [/, \text{MATCH_ALL_INITIAL_DOT} | /\text{MATCH_INITIAL_DOT}],

[/, \text{NOSORT}], [/, \text{QUOTE}]

**FILE_LINK** - Checks files for existence and other file attributes without first having to open the file.

\[
\text{Result} = \text{FILE_LINK}([\text{File}], [\text{/DIRECTORY}],
\]

\[
[\text{/EXECUTABLE}], [\text{/READ}], [\text{/REGULAR}], [\text{/WRITE}],
\]

[/, \text{ZERO_LENGTH}], [\text{GET_MODE}=\text{variable}]

[\text{/NOEXPAND_PATH}]

**FILE_READLINK** - Returns the path pointed to by a UNIX symbolic link.

\[
\text{Result} = \text{FILE_READLINK}([\text{Path}], [\text{/ALLOW NONSYMLINK}],
\]

\[
[\text{/NOEXPAND_PATH}]
\]

**FILE_LINES** - Returns the number of lines of text in a file.

\[
\text{Result} = \text{FILE_LINES}([\text{Path}], [\text{/COMPRESS}],
\]

\[
[\text{/NOEXPAND_PATH}]
\]

**FILE_LINK** - Creates UNIX file links.

\[
\text{Result} = \text{FILE_LINK}([\text{SourcePath}], [\text{DestPath}], [\text{/ALLOW SAME}],
\]

\[
[\text{/HARDLINK}], [\text{/NOEXPAND_PATH}], [\text{/VERBOSE}]
\]

**FILE_MKDIR** - Creates a new directory, or directories, with default access permissions for the current process.

\[
\text{Result} = \text{FILE_MKDIR}([\text{File1}], [\text{...}], [\text{FileN}], [\text{/NOEXPAND_PATH}])
\]

**FILE_MOVE** - Renames files and directories.

\[
\text{Result} = \text{FILE_MOVE}([\text{SourcePath}], [\text{DestPath}], [\text{/ALLOW SAME}],
\]

\[
[\text{/NOEXPAND_PATH}], [\text{/OVERWRITE}], [\text{/RECURSIVE}]
\]

UNIX-Only Keywords: [/, \text{MOVE_NAMED_PIPE}]

**FILE_READLINK** - Returns the path pointed to by a UNIX symbolic link.

\[
\text{Result} = \text{FILE_READLINK}([\text{Path}], [\text{/ALLOW NONSYMLINK}],
\]

\[
[\text{/NOEXPAND_PATH}]
\]

**FILESAME** - Determines whether two different file names refer to the same underlying file.

\[
\text{Result} = \text{FILESAME}([\text{Path1}], [\text{Path2}],
\]

\[
[\text{/NOEXPAND_PATH}]
\]

**FILE_SEARCH** - Returns a string array containing the names of all files matching the input path specification.

\[
\text{Result} = \text{FILE_SEARCH}([\text{Path Specification}],
\]

\[
[\text{Recursion}]
\]

UNIX-Only Keywords: [/, \text{MATCH_ALL_INITIAL_DOT} | /\text{MATCH_INITIAL_DOT}],

[/, \text{NOSORT}], [/, \text{QUOTE}]

**FILE_LINK** - Checks files for existence and other file attributes without first having to open the file.

\[
\text{Result} = \text{FILE_LINK}([\text{File}], [\text{/DIRECTORY}],
\]

\[
[\text{/EXECUTABLE}], [\text{/READ}], [\text{/REGULAR}], [\text{/WRITE}],
\]

[/, \text{ZERO_LENGTH}], [\text{GET_MODE}=\text{variable}]

[\text{/NOEXPAND_PATH}]

**FILE_READLINK** - Returns the path pointed to by a UNIX symbolic link.

\[
\text{Result} = \text{FILE_READLINK}([\text{Path}], [\text{/ALLOW NONSYMLINK}],
\]

\[
[\text{/NOEXPAND_PATH}]
\]
FILE_TEST - continued
UNIX-Only Keywords: [. /BLOCK_SPECIAL | . /CHARACTER_SPECIAL | . /DANGLING_SYMLINK | . /GROUP | . /NAMED_PIPE | . /SETGUID | . /SETUID | . /SOCKET | . /STICKY_BIT | . /SYMLINK | . /USER]

FILE WHICH - Separates a specified file path into its component
directories, and searches each directory in turn for a specific file.
Result = FILE WHICH( Path ) File
[ , /INCLUDE_CURRENT_DIR ]

FILEPATH - Returns full path to a file in the IDL distribution.
Result = FILEPATH( Filename [, Root_DIR=string ]
[ , SubDIRECTORY=string/string_array ]
[ , /TERMINAL ] [ , /TMP ])

FINDGEN - Returns a floating-point array with each element set to its
subscript.
Result = FINDGEN( D1 [ , ... , D8 ] [ , Thread pool keywords ])

FIX - Converts argument to integer type, or type specified by TYPE key-
word.
Result = FIX( Expression [ , Offset [ , D1 [ , ... , D8 ] ]
[ , /PRINT ] [ , TYPE=type code (0 to 15) ] [ , Thread pool
keywords ])

FINITE - Returns True if its argument is finite.
Result =FINITE( X [ , /INFINITY ] [ , /NAN ]
[ , SIGN=value ] [ , /Thread pool keywords ])

FLUSH - Draws lines representing a 3D flow/velocity field.
FLOW3, Vx, Vy, Vz [] , ARROWSIZE=value [] , /BLOB
[ , LEN=value ] [ , NSTEPS=value ] [ , NVECS=value ]
[ , SX=vector, SY=vector, SZ=vector ]

FLICK - Causes the display to flicker between two images.
Flick, A, B [ , Rate ]

FLOAT - Converts argument to single-precision floating-point.
Result = FLOAT( Expression [ , Offset [ , D1 [ , ... , D8 ] ]
[ , Thread pool keywords ])

FLOW - Returns closest integer less than or equal to argument.
Result = FLOOR( X [ , /L64 ] [ , Thread pool keywords ])

FOR - Executes statements repeatedly, incrementing or decrementing a
variable with each repetition, until a condition is met.
FOR variable = init, limit [ , Increment ] DO statement
or
FOR variable = init, limit [ , Increment ] DO BEGIN
statements
ENDFOR

FORMAT_AXIS_VALUES - Formats numbers as strings for use as
axis values.
Result = FORMAT_AXIS VALUES( Values )

FORWARD_FUNCTION - Causes argument(s) to be interpreted as
functions rather than variables (versions of IDL prior to 5.0 used
parentheses to declare arrays).
FORWARD_FUNCTION Name1 [, Name2, ... , NameN]

FREE_LUN - Frees previously-reserved file units.
FREE_LUN [ , Unit1 , ... , UnitN ]
[ , EXIT_STATUS=variable ] [ , /FORCE ]

FSTAT - Returns information about a specified file unit.
Result = FSTAT( Unit )

FULSTR - Restores a sparse matrix to full storage mode.
Result = FULSTR( A )

FUNCTION - Defines a function.
FUNCTION Function_Name, parameter1, ..., parameterN

FV_TEST - Performs the F-variance test.
Result = FV_TEST( X, Y )

FX_ROOT - Computes real and complex roots of a univariate nonlinear
function using an optimal Müller’s method.
Result = FX_ROOT( X, Func [ , /DOUBLE ]
[ , ITMAX=value ] [ , /STOP ] [ , TOL=value ]
)

FZ_ROOTS - Finds the roots of a complex polynomial using
Laguerre’s method.
Result = FZ_ROOTS( C [ , /DOUBLE ] [ , EPS=value ]
[ , /NO_POLISH ] )

GAMMA - Returns the gamma function of Z.
Result = GAMMA( Z [ , Thread pool keywords ]
)

GAMMA_CVF - Applies gamma correction to a color table.
GAMMA_CVF, Gamma [, /CURRENT ] [, /INTENSITY]

GAUSS_CVF - Computes cutoff value in Gaussian distribution.
Result = GAUSS_CVF( F )

GAUSS_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( V )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly
gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [ , /NEGATIVE ]
[ , /TILT ] )
Alphabetical List of IDL Routines

GAUSSFIT - Fits the sum of a Gaussian and a quadratic.

GAUSSINT - Returns integral of Gaussian probability function.
   Result = GAUSSINT(X [, /ENVIRONMENT] )

GRID_INPUT - Preprocesses and sorts two-dimensional scattered data points, and removes duplicate values.
   GRID_INPUT, X, Y, F, X1, Y1, F1
   or
   GRID_INPUT, Lon, Lat, F, Xyc, F1, /SPHERE
   [ , /DEGREES] [, DUPLICATES=string] [, EPSILON=value ] [, EXCLUDE=vector ]
   or
   GRID_INPUT, R, Theta, F, X1, Y1, F1, /POLAR
   [, /DEGREES] [, DUPLICATES=string ] [, EPSILON=value ] [, EXCLUDE=vector ]
   or
   GRID_INPUT, R, Theta, F, X1, Y1, F1, /POLAR
   [, /DEGREES] [, DUPLICATES=string ] [, EPSILON=value ] [, EXCLUDE=vector ]

GRID_TPS - Uses thin plate splines to interpolate a set of values over a regular 2D grid, from irregularly sampled data values.
   Interp = GRID_TPS(Xp, Yp, Values
   [ , COEFFICIENTS=variable ] [, NGRID=[nx, ny] ]
   [, START=[x0, y0]] [, DELTA=[dx, dy]] )

GRID3 - Creates a regularly-gridded 3D dataset from a set of scattered 3D nodes.
   Result = GRID3(X, Y, Z, F, Gx, Gy, Gz
   [, DELTA=scalar/vector] [, DTOL=value ]
   [, GRID=variable ] [, NGRID=value ] [, START=[x, y, z]] )

GRIDDATA - Interpolates scattered data values and locations sampled on a plane or a sphere to a regular grid.
   Result = GRIDDATA(X, F )
   or
   Result = GRIDDATA(X, Y, F )
   or
   Result = GRIDDATA(X, Y, Z, F, /SPHERE )
   or
   Result = GRIDDATA(Lon, Lat, F, /SPHERE )

GS_ITER - Solves linear system using Gauss-Seidel iteration.
   Result = GS_ITER(A, B [, , /CHECK] [, , /DOUBLE]
   [, LAMBDA=value{0.0 to 2.0}] [, MAX_ITER=value ]
   [, TOL=value ] [, X_0=vector ]

H

H_EQ_CT - Histogram-equalizes the color tables for an image or a region of the display.
   Result = H_EQ_CT(Filename
   [, TEMPLATE=string] [, IMAGE=variable ]
   [, /CHECK] [, /DOUBLE] )

H_EQ_INT - Interactively histogram-equalizes the color tables of an image or a region of the display.
   Result = H_EQ_INT(Filename
   [, TEMPLATE=string] [, IMAGE=variable ]
   [, /CHECK] [, /DOUBLE] )

HANNING - Creates Hanning and Hamming windows.
   Result = HANNING(N1 [, N2] [, ALPHA=value{0.5 to 1.0}] [, /DOUBLE] )

HDF_* Routines - See “HDF Routines” on page 130.

HDF5_* Routines - See “HDF5 Routines” on page 135.

HDF_BROWSER - Opens GUI to view contents of HDF, HDF-EOS, or NetCDF file.
   Template = HDF_BROWSER(Filename
   [, CANCEL=variable ] [, GROUP=widget_id
   [, PREFIX=string] ]

HDF_READ - Extracts HDF, HDF-EOS, and NetCDF data and metadata into an output structure.
   Result = HDF_READ(Filename
   [, DF24=variable ] [, PREFIX=string
   [, TEMPLATE=value ]

HEAP_FREE - Recursively frees all heap variables referenced by its input argument.
HELP - Provides information about the current IDL session.

HELP: Expression, ..., Expression [, /ALL_KEYS]
[ , /BREAKPOINTS] [, /BRIEF] [, CALLS=variable]
[ , /FUNCTIONS] [, /HEAP_VARIABLES] [, /KEYS]
[, /LAST_MESSAGE] [, /LEVEL=value] [, /MEMORY]
[ , /MESSAGES] [, NAMES=string_of_variable_names]
[, /OBJECTS] [, OUTPUT=variable] [, /PATH_CACHE]
[, /PROCEDURES] [, /RECALL_COMMANDS]
[, /Routines] [, /SHARED_MEMORY]
[, /SOURCE_FILES] [, /STRUCTURES]
[, /SYSTEM_VARIABLES] [, /TRACEBACK]

HILBERT - Constructs a Hilbert transform.

Result = HILBERT(X [, D])

HIST_2D - Returns histogram of two variables.

Result = HIST_2D( V1, V2, [BIN1=width] [, BIN2=height] [, MAX1=value] [, MAX2=value] [, MIN1=value] [, MIN2=value])

HIST_EQUAL - Histogram-equalizes an image.


HISTOGRAM - Computes the density function of an array.


HLS - Creates color table in Hue, Lightness, Saturation color system.

HLS, Lilith, Satilo, Sathì, Hue, Loops [, Colr]

HOUGH - Returns the Hough transform of a two-dimensional image.


HQR - Returns all eigenvalues of an upper Hessenberg array.

Result = HQR( A [, /COLUMN] [, /DOUBLE])

HSV - Creates color table based on Hue/Saturation Value color system.

HSV, Vlo, Vhi, Satlo, Sathi, Hue, Loops [, Colr]

IBETA - Computes the incomplete beta function.

Result = IBETA( A, B, Z [, /DOUBLE] [, EPS=value] [, ITER=variable] [, ITMAX=value])

ICONTOUR - Creates an iTool and associated user interface (UI) con-
figured to display and manipulate contour data.

ICONTOUR[, Z[, X[, Y]]]

iTool Common Keywords: [, BACKGROUND_COLOR=value]
[, DIMENSIONS=[x, y]] [, IDENTIFIER=variable]
[, LOCATION=[x, y]] [, MACRO_NAMES=string or string array]
[, NAME=string] [, NO_SAVEPROMPT] [, OVERPLOT=[toolID]] [, STYLE_NAME=string]
[, TITLE=string] [, VIEW_GRID=[columns, rows]] [, VIEW_NEXT] [, VIEW_NUMBER=integer]

iTool Contour Keywords: [, RGB_INDICES=vector of indices] [, RGB_TABLE=byte array of 256 by 3 or 3 by 256 elements] [, ZVALUE=value]

Contour Object Keywords: [, AM_PM=vector of two strings] [, ANISOTROPY=[x, y, z]] [, C_COLOR=color array] [, C_FILL_PATTERN=any of IDLgrPattern objects] [, C_LABEL_INTERVAL=vector]
[, C_LABEL_NOGPS=vector] [, C_LABEL_OBJECTS=array of object references]
[, C_LABEL_ORIENTATION=vector of values] [, C_LABEL_NOGAPS=vector] [, C_LABEL_INTERVAl=vector]
[, C_FILL_P ATTERN=any of values] [, C_VALUE=scalar or vector] [, CLIP_PLANES=vector]
[, COLOR=RGB vector] [, DAYS_OF_WEEK=vector of 7 strings] [, DEPTH_OFFSET=value]
[, /DOWNHILL] [, /FILL] [, GRID_UNITS=value]
[, /FORMAT=vector] [, /LABEL_FONT=vector] [, /LABEL_FORMAT=vector]
[, /LABEL_FORMAT=string] [, /LABEL_FLOAT=vector] [, /LABEL_FLOAT=vector]
[, /LABEL_UNITS=vector of 12 values] [, /LEVELS=value] [, /PLANAR]
[, SHEADING=[min, max]] [, /SHADING=[0 | 1]] [, /TICKINTERVAL=value] [, /TICKLEN=value]
[, USE_TEXT_ALIGNMENTS=value]

Axis Object Keywords: [, [X | Y | Z]GRIDSTYLE=[0 | 1] [, 2 | 3 | 4 | 5 | 6]]
[, [X | Y | Z][MAJOR=integer]] [, [X | Y | Z][MINOR=integer]]
[, [X | Y | Z][RANGE=[min, max]]] [, [X | Y | Z][SUBTICKLEN=boolean]]
[, [X | Y | Z][TEXT_COLOR=RGB vector]]
[, [X | Y | Z][TICKFONT_INDEX= [0 | 1] 2 | 3 | 4]]
[, [X | Y | Z][TICKFONT_SIZE=integer]]

ICOUNTOUR Axis Object Keywords — continued
[TICKFORMAT_STYLE=[0 | 2 | 3], [X | Y | Z]
TICKLABEL=value [X | Y | Z], TICKLAYOUT= [0 | 1 | 2], [X | Y | Z], TICKNAMES=vector X | Y | Z], TICKVALUES=vector

IDENTITY - Returns an identity array.
Result = IDENTITY(N \[, /DOUBLE\])

IDL_Container Object - See "IDL_Container" on page 73.

IDL_VALIDNAME - Determines whether a string may be used as a valid IDL variable name or structure tag name.
Result = IDL_VALIDNAME(String \[, /CONVERT_ALL\] \[, CONVERT_SPACES\])

IDLanROI Object - See "IDLanROI" on page 74.

IDLanROIGroup Object - See "IDLanROIGroup" on page 74.

IDLffDICOM Object - See "IDLffDICOM" on page 75.

IDLffDXF Object - See "IDLffDXF" on page 76.

IDLffJPEG2000 Object - See "IDLffJPEG2000" on page 76.

IDLffLangCat Object - See "IDLffLangCat" on page 77.

IDLffMrSID Object - See "IDLffMrSID" on page 77.

IDLffShape Object - See "IDLffShape" on page 77.

IDLffXMLDOM - See "IDLffXMLDOM Classes" on page 78.

IDLffXMLSAX Object - See "IDLffXMLSAX" on page 82.

IDLgr* Objects - IDLgr* objects and their methods are described starting with "IDLgrAxis" on page 83.

IDL* Objects - IDL* objects and their methods are described starting with "IDL_" on page 97.

IDLSYS_CREATETOOL - Creates an instance of the specified tool registered within the iTools system.
Result = IDLSYS_CREATETOOL(StrTool \[, DIMENSIONS=[width, height]\] \[, IDENTIFIER=variable\] \[, INITIAL_DATA=data\] \[, LOCATION=[x, y]\] \[, MACRO_NAMES=vector or string array\] \[, NO_SAVEPROMPT\] \[, OVERPLOT=iToolID\] \[, STYLE_NAME=string\] \[, VIEW_GRID=vector\] \[, VIEW_NEXT\] \[, VIEW_NUMBER=number\] \[, VISUALIZATION_TYPE=visistyle\])

IF...THEN...ELSE - Conditionally executes a statement or block of statements.
IF expression THEN statement [ ELSE statement ]
or
IF expression THEN BEGIN
  statements
ENDIF [ ELSE BEGIN
  statements
ENDELSE ]
**IMAGE_STATISTICS** - Computes sample statistics for a given array of values.

```
IMAGE_STATISTICS, Data [, /LABELED
| [, /WEIGHTED | [ WEIGHT_SUM=value ]
| [, /VECTOR | [, LUT=array ] | | , MASK=array ]
| [, COUNT=value ] | | | | , MEAN=value ]
| , STDERR=value ] | | | | , DATA_SUM=value ]
| , SUM_OF_SQUARES=value ]
| , MINIMUM=value ] | | | | , MAXIMUM=value ]
| | | | , VARIANCE=value ]
```

**IMAGINARY** - Returns the imaginary part of a complex value.

```
Result = IMAGINARY(Complex_Expression [, Thread pool keywords])
```

**IMAP**

- This procedure accepts all ITool Image Keywords:
  - iTool Contour Keywords: If the CONTOUR keyword is

```
IMAP[, MAP_PROJECTION=value
| | , MAP_PROJECTION=string
| |
```

- IMAP[, X, Y][] | GRID_UNITS=value
- IMAP[, Z, X, Y][] | GRID_UNITS=value
- IMAP[, MAP_PROJECTION=string

**iTool Common Keywords:**

```
| \[ BACKGROUND_COLOR=value
| | | \[ DIMENSIONS=x, y ] | | | IDENTIFIER=value
| | | \[ LOCATION=x, y ]
| | | \[ MACRO_NAME=value string or string array
| | | \[ NAME=string | \[ NO_SAVEPROMPT
| | | \[ OVERPLOT=toolID | | | STYLE_NAME=string
| | | \[ VIEW_GRID=columns, rows ]
| | | \[ VIEW_NEXT ] | | | VIEW_NUMBER=value
```

**iTool Image Keywords:** This procedure accepts all IMAGE keywords.

**iTool Contour Keywords:** If the CONTOUR keyword is set, this procedure accepts all ICONTOUR keywords.

**Map Projection Keywords:**

```
| \[ CENTER_LATITUDE=value
| | | \[ CENTER_LONGITUDE=value | | | DATUM=string
| | | \[ FALSE_EASTING=value
| | | \[ FALSE_NORTHING=value | | | HEIGHT=value
| | | \[ HOM_AZIM_LATITUDE=value
| | | \[ HOM_AZIM_ANGLE=value
| | | \[ HOM_LATITUDE=value
| | | \[ HOM_LATITUDE2=value
| | | \[ HOM_LONGITUDE=value
| | | \[ HOM_LONGITUDE2=value
| | | \[ IS_ZONES=value
| | | \[ IS_ORTHODROMIC=value
| | | \[ LIMITS=latmin, lonmin, latmax, lonmax ]
| | | \[ MERCATOR_SCALE=value | | | OEA_ANGLE=value
| | | \[ OEA_SHAPE=value | | | OEA_SHAPEN=value
| | | \[ SEMIMAJOR_AXIS=value
| | | \[ SEMIMINOR_AXIS=value
| | | \[ SOM_INCLINATION=value
| | | \[ SOM_LONGITUDE=value
| | | \[ SOM_LANDSAT_NUMBER=value
| | | \[ SOM_LANDSAT_PATH=value
| | | \[ SPHERE_RADIUS=value
| | | \[ TRUE_SCALE_LATITUDE=value
| | | \[ ZONE=value
```

**Axis Keywords:**

```
| \[ XY_GRIDSTYLE=0|1|2|3|4|5|6 ]
| | | \[ XY_MAJOR=value | \[ XY_MINOR=value
| | | \[ XY RANGE=min, max ]
| | | \[ XY_TICKFORM=0|1|2|3 ]
| | | \[ XY_TICKFONT_INDEX=0|1|2|3|4 ]
| | | \[ XY_TICK_INTERV=integer
| | | \[ XY_TICKINTERVAL=value
| | | \[ XY_TICKLABELS=value
| | | \[ XY_TICKNAME=value string
| | | \[ XY_TICKUNITS=vector
| | | \[ XY_TICKVALUES=vector
| | | \[ XY_TITLE=string
```

**INDGEN**

- Return an integer array with each element set to its sub-script.

```
Result = INDGEN(D1 [, ... , D8 ] [, /BYTE | /COMPLEX | /DCOMPLEX | /DOUBLE | /FLOAT | /L64 | /LONG | /STRING | \[ /UINT | /UL64 | /ULONG ]
| \[ TYPE=value | \[ Thread pool keywords ]
```

**INT_2D**

- Computes the double integral of a bivariate function.

```
Result = INT_2D( Fxy, AB_Limits, PQ_Limits, Pts [, /DOUBLE ] [, /ORDER ]
```

**INT_3D**

- Computes the triple integral of a trivariate function.

```
Result = INT_3D( Fxyz, AB_Limits, PQ_Limits, UV_Limits, Pts [, /DOUBLE ]
| , /SORT )
```

**INT_TABULATED**

- Integrates a tabulated set of data.

```
Result = INT_TABULATED( X, F [, /DOUBLE ]
| [, /SORT ]
```

**INTARR**

- Creates an integer vector or array.

```
Result = INTARR( D1 [, ... , D8 ] [, /NOZERO ]
```

**INTERPOL**

- Performs linear interpolation on vectors.

For regular grids: `Result = INTERPOL( V, N` | /LQUADRATIC | /QUADRATIC | /SPLINE )

For irregular grids: `Result = INTERPOL( V, X, U` | /LQUADRATIC | /QUADRATIC | /SPLINE )

**INTERPOLATE**

- Returns an array of interpolates.

```
Result = INTERPOLATE( P, X [, Y, Z] ]
| \[ CUBICS=1 to 0 ] | [, /GRID ] | [, MISS=value
| \[ Thread pool keywords ]
```

**IMAGE_STATISTICS**

**IDL Quick Reference**
**INVERT** - Computes the inverse of a square array.

Result = INVERT(Array [, Status] [, /DOUBLE] )

Return Status:

0, Value 0, Outverts, Outconn [ , AUXDATA_IN=array, AUXDATA_OUT=variable]

[ , GEOM_XYZ=array, TETRAHEDRA=array]

[ , PROGRESS_CALLBACK=string]

[ , PROGRESS_METHOD=string]

[ , PROGRESS_OBJECT=object]

[ , PROGRESS_PERCENT=percent(0 to 100)]

[ , PROGRESS_USERDATA=value]

**INTERVAL_VOLUME** - Generates a tetrahedral mesh from volumetric data.

INTERVAL_VOLUME, Data, Value0, Value1, Outverts, Outconn [ , AUXDATA_IN=array, AUXDATA_OUT=variable]

[ , GEOM_XYZ=array, TETRAHEDRA=array]

[ , PROGRESS_CALLBACK=string]

[ , PROGRESS_METHOD=string]

[ , PROGRESS_OBJECT=object]

[ , PROGRESS_PERCENT=percent(0 to 100)]

[ , PROGRESS_USERDATA=value]

**ISOSURFACE** - Returns topologically consistent triangles by using oriented tetrahedral decomposition.

ISOSURFACE, Data, Value, Outverts, Outconn [ , AUXDATA_IN=array, AUXDATA_OUT=variable]

[ , GEOM_XYZ=array, TETRAHEDRA=array]

[ , PROGRESS_CALLBACK=string]

**ITALISC** - Performs special functions on UNIX files.

Result = IOCTL(File_Unit, Request, Arg)

[ , BY_VALUE] [, /MT_OFFLINE] [, /MT_REWIND]

[ , MT_SKIP_FILE=-number_of_files]

[ , MT_SKIP_RECORD=-number_of_records]

[ , /MT_WEOF]

[ , /SUPRESS_ERROR] )

**IPLOT** - Creates an iTool and associated user interface (UI) configured to display and manipulate plot data.

IPLOT, [X,] Y or IPLOT, X, Y, Z

or

IPLOT[, R], Theta [, P] POLAR

iTool Common Keywords:

[ , BACKGROUND_COLOR=value]

[ , DIMENSIONS=[x, y]] [ , IDENTIFIER=variable]

[ , LOCATION=[x, y]]

[ , MACRO_NAME=string or string array]

[ , NAME=string] [ , /NO_SAVEDPROMPT]

[ , OVERPLOT=ToolID] [ , STYLE_NAME=string]

[ , TITLE=string] [ , VIEW_GRID=[columns, rows]]

[ , VIEW_NEXT] [ , VIEW_NUMBER=integer]

iTool Plot Keywords:

[ , ERRORBAR_COLOR=RGB vector] [ , ERROR_CAP_SIZE=points(0.0 to 1.0)]

[ , FILL_BACKGROUND] [ , FILL_COLOR=RGB vector] [ , FILL_LEVEL=value] [ , RGB_TABLE=byte array of 256 by 3 or 3 by 256 elements] [ , /SCATTER]

[ , SYM_COLOR=RGB color]

[ , SYM_INCREMENT=integer]

[ , SYM_INDEX=integer] [ , SYM_SIZE=point(0.0 to 1.0)]

[ , SYM_THICK=points(1.0 to 10.0)]

[ , TRANSPARENCY=percent(0.0 to 100.0)]

[ , /USE_DEFAULT_COLOR] [ , /XY_SHADOW]

[ , /X [ Y | Z] ERRORBARS] [ , /X [ Y | Z] LOG]

[ , X [ Y | Z] ERROR=vector or array]

[ , /XZ_SHADOW] [ , /YZ_SHADOW]

Plot Object Keywords:

[ , CLIP_PLANES=array]

[ , COLOR somebody] [ , /HIDE] [ , /HISTOGRAM]

[ , LINES_STYLE=integer] [ , MAX_VALUE=value]

[ , MIN_VALUE=value] [ , /HIDE] [ , /HISTOGRAM]

[ , THICK=points(1.0 to 10.0)]

[ , VERT_COLORS=vector]

Axis Object Keywords:

[ , X | Y | Z] GRIDSTYLE=[0 | 1]

[ , X | Y | Z] MAJOR=integer

[ , X | Y | Z] MINOR=integer

[ , X | Y | Z] SUBTICKLEN=ratio

[ , X | Y | Z] TEXT_COLOR=RGB vector

[ , X | Y | Z] TICKFONT_INDEX=[0 | 1 | 2 | 3 | 4]

[ , X | Y | Z] TICKFONT_SIZE=integer] [ , X | Y | Z]

TICKFONT_STYLE=[0 | 1 | 2 | 3] [ , X | Y | Z]

TICKFORMAT=string or string array] [ , X | Y | Z]

TICKINTERVAL=value]

[ , X | Y | Z] TICKLAYOUT=[0 | 1 | 2]

[ , X | Y | Z] TICKLEN=value] [ , X | Y | Z]

TICKNAME=string array] [ , X | Y | Z]

TICKUNITs=string] [ , X | Y | Z]

TICKVALUES=vector]

[ , X | Y | Z] TITLE=string

**ISHFT** - Performs integer bit shift.

Result = ISHFT(P1, P2 [, Thread pool keywords])

ISOCONTOUR - Interprets the contouring algorithm found in the IDL CreateContour object.

ISOCONTOUR, Values, Outverts, Outconn [ , AUXDATA_IN=array, AUXDATA_OUT=variable]

[ , C_LABEL_INTERVAL=vector of values]

[ , C_LABEL_SHOW=vector of integers]

[ , C_VALUE=scalar or vector] [, /DOUBLE] [ , /FILL]

[ , GEOMX=vector] [ , GEOMY=vector]

[ , GEOMZ=vector] [ , LEVEL_VALUES=variable]

[ , N_LEVELS=levels]

[ , OUT_LABEL_OFFSETS=variable]

[ , OUT_LABEL_PREFERENCES=variable]

[ , OUT_LABEL_STRINGS=variable]

[ , OUT_CONN_INDICES=variable]

[ , POLYGONS=array of polygon descriptions]
ISURFACE - Creates an iTool and associated user interface (UI) configured to display and manipulate surface data.

ISURFACE, [Z], [X, Y] iTool Common Keywords:
- [BACKGROUND_COLOR=value]
- [DIMENSIONS=[x, y]] [IDENTIFIER=variable]
- [LOCATION=[x, y]]
- [MACRO_NAMES=string or string array]
- [NAME=string] [NO_SAVEPROMPT]
- [OVERPLOT=ToolID] [STYLE_NAME=string]
- [TITLE=string] [VIEW_GRID=[columns, rows]]
- [VIEW_NEXT] [VIEW_NUMBER=integer]

iTool Surface Keywords:
- [RGB_TABLE=] array of 256 by 3 or 256 elements
- [TEXTURE_ALPHA=] 2-D array
- [TEXTURE_GREEN=] 2-D array
- [TEXTURE_IMAGE=] 2-D array
- [TEXTURE_RED=] 2-D array

Surface Object Keywords:
- [BOTTOM=] index or RGB vector
- [CLIP_PLANES=] array of RGB vector
- [COLOR=] RGB vector
- [DEPTH_OFFSET=value] [DOUBLE]
- [EXTENDED_LEGO] [HIDDEN_LINES] [HIDE]
- [LINESTYLE=] value [SHADING=[0 | 1]]
- [SHOW_SKIRT] [SKIRT=Z value]
- [STYLE=] [0 | 1 | 2 | 3 | 4 | 5 | 6]
- [TEXTURE_Highb repercussion=] [0 | 1 | 2]
- [TEXTURE_INTERP] [THICK=points [1.0 to 10.0]]
- [USE_TRIANGLES] [VERT_COLORS=vector or 2-D array] [ZERO_OPACITY_SKIP=0 | 1]

Axis Object Keywords:
- [X | Y | Z] GRIDSTYLE=[0 | 1 | 2 | 3 | 4 | 5 | 6]
- [X | Y | Z] MAJOR=integer
- [X | Y | Z] MINOR=integer
- [X | Y | Z] RANGE=[min, max]
- [X | Y | Z] SUBTICKLEN=abi
- [X | Y | Z] TEXT_COLOR=RGB vector
- [X | Y | Z] TICKFONT_INDEX= [0 | 1 | 2 | 3 | 4]
- [X | Y | Z] TICKFONT_SIZE=integer
- [X | Y | Z] TICKINTERVAL=value
- [X | Y | Z] TICKLAYOUT= [0 | 1 | 2]
- [X | Y | Z] TICKLEN=value
- [X | Y | Z] TICKNAME=string array
- [X | Y | Z] TICKVALUES=vector
- [X | Y | Z] TITLE=string

ITCURRENT - Sets the current tool in the iTools system.

ITCURRENT, iToolID

ITDELETE - Deletes a tool in the iTools system.

ITDELETE, [iToolID]

ITGETCURRENT - Gets the identifier of the current tool in the iTools system.

Result = ITGETCURRENT([ TOOL=variable])

ITREGISTER - Registers tool classes with the iTools system.

ITREGISTER, Name, ItemName [, /ANNOTATION]
- [DEFAULT] [FILE_READER] [FILE_WRITER]
- [TYPE=string] [, /UI_PANEL], [/UI_SERVICE]
- [/USER_INTERFACE] [/VISUALIZATION]

ITRESET - Resets the iTools session.

ITRESET [, /NO_PROMPT]

ITRESOLVE - Resolves all IDL code within the iTools directory, as well as all other IDL code required for the iTools framework.

ITRESOLVE [, PATH=string]

IVOLUME - Creates an iTool and associated user interface (UI) configured to display and manipulate volume data.

IVOLUME, Vol0, Vol1, Vol2, Vol3 iTool Common Keywords:
- [BACKGROUND_COLOR=value]
- [DIMENSIONS=[x, y]] [IDENTIFIER=variable]
- [LOCATION=[x, y]]
- [MACRO_NAMES=string or string array]
- [NAME=string] [NO_SAVEPROMPT]
- [OVERPLOT=ToolID] [STYLE_NAME=string]
- [TITLE=string] [VIEW_GRID=[columns, rows]]
- [VIEW_NEXT] [VIEW_NUMBER=integer]

iTool Volume Keywords:
- [AUTO_RENDER]
- [BOUNDEXTS=[0 | 1 | 2]]
- [BOUNDQUALITY=[0 | 1 | 2]]
- [SUBVOLUME=[xmin, ymin, zmin, xmax, ymax, zmax]]
- [VOLUME_DIMENSIONS=[width, height, depth]]
- [VOLUME_LOCATION=[x, y, z]]

Volume Object Keywords:
- [AMBIENT=RGB vector]
- [BRIGHTENE=[0 | 1 | 2 | 3]]
- [CLIP_PLANES=] array
- [COMPOSITE_FUNCTION=] [0 | 1 | 2 | 3]
- [DEPTH_CUE=[0 | 1 | 2 | 3]]
- [HINTS=[0 | 1 | 2 | 3]] [ANTERPOLATE]
- [LIGHTING_MODEL]
- [OPACITY_TABLE0=byte array of 256 elements]
- [OPACITY_TABLE1=byte array of 256 elements]
- [RENDER_STEP=] [x, y, z]
- [RGB_TABLE0=byte array of 256 by 3 or 256 elements]
- [RGB_TABLE1=byte array of 256 by 3 or 256 elements]
- [TWO_SIDED] [ZBUFFER]
- [ZERO_OPACITY_SKIP=[0 | 1]]

Axis Object Keywords:
- [X | Y | Z] GRIDSTYLE=[0 | 1 | 2 | 3 | 4 | 5 | 6]
- [X | Y | Z] MAJOR=integer
- [X | Y | Z] MINOR=integer
- [X | Y | Z] RANGE=[min, max]
- [X | Y | Z] SUBTICKLEN=abi
- [X | Y | Z] TEXT_COLOR=RGB vector
- [X | Y | Z] TICKFONT_INDEX= [0 | 1 | 2 | 3 | 4]
- [X | Y | Z] TICKFONT_SIZE=integer
- [X | Y | Z] TICKINTERVAL=value
- [X | Y | Z] TICKLAYOUT= [0 | 1 | 2 | 3 | 4]
- [X | Y | Z] TICKLEN=value
- [X | Y | Z] TICKNAME=string array
- [X | Y | Z] TICKVALUES=vector
- [X | Y | Z] TITLE=string
LABEL_REGION - Labels regions (blobs) of a bi-level image.
Result = LABEL_REGION( Array [, /ALL_NEIGHBORS] [, /LONG] )

LADFIT - Fits paired data using least absolute deviation method.
Result = LADFIT( X, Y [, ABSDEV=variable] [, /DOUBLE] )

LAGUERRE - Returns value of the associated Laguerre polynomial.
Result = LAGUERRE( X, N [, K] [, COEFFICIENTS=variable] [, /DOUBLE] )

LA_CHOLDC - Computes the Cholesky factorization of an n-by-n symmetric (or Hermitian) positive-definite array.
LA_CHOLDC, Array [, /DOUBLE] [, STATUS=variable] [, /UPPER] 

LA_CHOLMPROVE - Uses Cholesky factorization to improve the solution to a system of linear equations.
Result = LA_CHOLMPROVE( A, B [, /DOUBLE] [, /UPPER] )

LA_DETERM - Uses LU Decomposition to compute the determinant of a square array.

LA_EIGENPROBLEM - Uses the QR algorithm to compute all eigenvalues and eigenvectors of an array.

LA_EIGENQL - Computes eigenvalues and eigenvectors of an array.

LA_EIGENVEC - Uses the QR algorithm to compute all or some of the eigenvectors of an array.
**Alphabetical List of IDL Routines**

**LA_ELMHES** - Reduces a real nonsymmetric or complex non-Hermitian array to upper Hessenberg form H.

```idl
```

**LA_LINEAR_EQUATION** - Used to solve a set of sparse linear equations using the iterative biconjugate gradient method.

```idl
```

**LA_LUDC** - Computes the LU decomposition of an n-column by m-row array.

```idl
LA_LUDC, Array, Index [, /DOUBLE] [, STATUS=variable]
```

**LA_LUMPROVE** - Uses LU decomposition to improve the solution to a system of linear equations.

```idl
Result = LA_LUMPROVE( Array , Aludc , Index , B , X [, BACKWARD_ERROR=variable] [, /DOUBLE] [, FORWARD_ERROR=variable] )
```

**LA_LUSOL** - Used in conjunction with the LA_LUDC procedure to solve a set of n linear equations in m unknowns, AX = B.

```idl
Result = LA_LUSOL( A , Index , B [, /DOUBLE] )
```

**LA_SVD** - procedure computes the singular value decomposition of an n-columns by m-row array.

```idl
```

**LA_TRIDC** - Computes the LU decomposition of a tridiagonal array as array:

```idl
LA_TRIDC, AL, A, AU, U2, Index [, /DOUBLE] [, STATUS=variable]
```

**LA_TRIMPROVE** - Improves the solution to a system of linear equations with a tridiagonal array.

```idl
```

**LA_TRIQL** - Uses the QR and QL variants of the implicitly-shifted QR algorithm to compute the eigenvalues and eigenvectors of a symmetric tridiagonal array.

```idl
LA_TRIQL, D, E [, A] [, /DOUBLE] [, STATUS=variable]
```

**LA_TRISOL** - Used in conjunction with the LA_TRIDC procedure to solve a set of linear equations.

```idl
Result = LA_TRISOL( AL, A, AU, U2, Index, B [, /DOUBLE] )
```

**LEEFILT** - Performs the Lee filter algorithm on an image array.

```idl
Result = LEEFILT( A [, N [, Sig]] [, /DOUBLE] [, /EXACT] )
```

**LEGENDRE** - Returns value of the associated Legendre polynomial.

```idl
Result = LEGENDRE( X, L [, M] [, /DOUBLE] )
```

**LINBCG** - Solves a set of sparse linear equations using the iterative biconjugate gradient method.

```idl
```

**LINDGEN** - Returns a longword integer array with each element set to its subscript.

```idl
Result = LINDGEN(D [, ..., D8] [, Thread pool keywords] )
```

**LINFIT** - Fits by minimizing the Chi-square error statistic.

```idl
```

**LINKIMAGE** - Merges routines written in other languages with IDL at run-time.

```idl
LINKIMAGE, Name, Image [, Type [, Entry]] [, /DEVICE] [, /FUNCTION] [, KEYWORDS] [, MAX_ARGS=value] [, MIN_ARGS=value]
```

**LL_ARC_DISTANCE** - Returns the longitude and latitude of a point given arc distance and azimuth.

```idl
Result = LL_ARC_DISTANCE( Lon_lat0, Arc_Dist, Az [, /DEGREES] )
```
LMGR - Determines the type of license used by the current IDL session.

LOCALE_GET - Returns the current locale of the operating platform.
Result = LOCALE_GET()

LOGICAL_AND - Performs a logical AND operation on its arguments.
Result = LOGICAL_AND(Arg[1], Arg[2])

LOGICAL_TRUE - Determines whether its arguments are non-zero (or non-NULL).
Result = LOGICAL_TRUE(Arg)

LON64ARR - Returns a 64-bit integer vector or array.
Result = LON64ARR(D[1], ..., D[8]) [, /NOZERO])

LONARR - Returns a longword integer vector or array.
Result = LONARR(D[1], ..., D[8]) [, /NOZERO])

LONG - Converts argument to longword integer type.
Result = LONG(Expression [, Offset [, D[1], ..., D[8]]] [, Thread pool keywords])

LONG64 - Converts argument to 64-bit integer type.
Result = LONG64(Expression [, Offset [, D[1], ..., D[8]]] [, Thread pool keywords])

LSODE - Advances a solution to a system of ordinary differential equations one time-step H.
Result = LSODE(Y, X, H, Derivs[, Status] [, ATOL=value] [, /QUiet] [, RTOL=value])

LU_COMPLEX - Solves complex linear system using LU decomposition.
Result = LU_COMPLEX(A, B [, /DOUBLE] [, /INVERSE] [, /SPARSE])

LUDC - Replaces array with the LU decomposition.
LUDC, A, Index [, /COLUMN] [, /DOUBLE] [, INTERCHANGES=variable]

LUMPROVE - Uses LU decomposition to iteratively improve an approximate solution.
Result = LUMPROVE(A, Alad, Index, B, X [, /COLUMN] [, /DOUBLE])

LUSOL - Solves a set of linear equations. Use with LUDC.
Result = LUSOL(A, Index, B [, /COLUMN] [, /DOUBLE])

M_CORRELATE - Computes multiple correlation coefficient.
Result = M_CORRELATE(X, Y [, /DOUBLE])

MACHAR - Determines and returns machine-specific parameters affecting floating-point arithmetic.
Result = MACHAR([, /DOUBLE])

MAKE_ARRAY - Returns an array of the specified type, dimensions, and initialization.

MAKE_DLL - Builds a shareable library compatible for use with IDL's dynamic linking.

MAP_2POINTS - Returns distance, azimuth, and path relating to the great circle or rhumb line connecting two points on a sphere.
Result = MAP_2POINTS(lon0, lat0, lon1, lat1 [, DPATH=value] [, /METERS] [, /MILES] [, NPATH=integer (2 or greater)] [, /PARAMETERS] [, RADIUS=value] [, /RADlANS] [, /RHUMB])
MAP_CONTINENTS - Draws continental boundaries, filled continents, political boundaries, coastlines, and/or rivers, over an existing map projection established by MAP_SET.


Graphics Keywords: [, /T3D]
[, ZVALUE=value] [0 to 1]]

MAP_GRID - Draws parallels and meridians over a map projection.

MAP_GRID [, /BOX_AXES [, /CLIP_TEXT=0]] [, LATALIGN=value] [0.0 to 1.0] [, LONALIGN=value] [0.0 to 1.0] [, LATLAB=latitude] [, LONLAB=longitude] [, ORIENTATION=degrees] [, /HORIZONTAL] [, /VERTICAL]

MAP_STRUCTURE=structure] [, /NO_GRID]

Graphics Keywords: [, /T3D]
[, ZVALUE=value] [0 to 1]]

MAP_IMAGE - Returns an image warped to fit the current map projection. (Use when map data is smaller than the display).


MAP_PATCH - Returns an image warped to fit the current map projection. (Use when map data is smaller than the display).

[, XSIZE=variable] [, XSTART=variable] [, YSIZE=variable] [, YSTART=variable]}

MAP_PROJ_FORWARD - Transforms map coordinates from longitude/latitude to Cartesian (X, Y) coordinates.


MAP_PROJ_IMAGE - Warps an image from geographic coordinates to a specified map projection.


MAP_PROJ_INFO - Returns information about current map and/or the available projections.


Keywords—Projection Parameters:
MAP_PROJ_INVERSE - Transforms map coordinates from Cartesian (X, Y) to geographic (Lat, Long).

Result = MAP_PROJ_INVERSE( X [, Y] [, MAP_STRUCTURE=variable] [, /RADIANS] )

MAP_SET - Establishes map projection type and limits.

MAP_SET [ , Polat, Polon, Rot ]

Keywords—Projection Types: [ , /AITOFF | , /ALBERS [ , /AZIMUTHAL | , / CONIC | , / CYLINDRICAL | , / GNOMIC | , / GOOD_S_HOMOLOSINE | , / HAMMER | , / LAMBERT | , / MERCATOR | , / MILLER_CYLINDRICAL | , / MOLLWEIDE | , / ORTHOGRAPHIC | , / ROBINSON | , / SATELLITE | , / SINUOSIDAL | , / STEREOPHGRAPHIC | , / TRANSVERSE_MERCATOR | NAME=string ]

Keywords—Map Characteristics: [ , /ADVANCE | , / CENTRAL_AZIMUTH=degrees_east_of_north ] [ , / DEGREES | , / SAT_P=value ] [ , / ENHANCED | , / USASTANDARD | , / USA[unit] ] [ , / IRRADIAN ] [ , / ISOTROPIC | , / LIMIT=value ] [ , / NUM_ALLOC | , / NUM_FREE | , / STRUCTURE | , / L64 | ]

Graphics Keywords: [ , POSITION=X0 [, Y0 , X1 , Y1 ] ] [ , / FTD | , ZVALUE=value [0 to 1] ]

MESH_CLIP - Clips a polygonal mesh to an arbitrary plane in space and returns a polygonal mesh of the remaining portion.


MESH_DECIMATE - Reduces the density of geometry while preserving as much of the original data as possible.

Result = MESH_DECIMATE( Verts [, Conn [, ConnOUT [, VERTICES=variable ] ] ] , %=percent | , / PERCENT_VERTICES=percent ] [ , / PERCENT_POLYGONS=percent ] [ , / PROGRESS_CALLBACK=string ] [ , / PROGRESS_METHOD=string ] [ , / PROGRESS_OBJECT=object ] [ , / PROGRESS_PERCENT=percent [0 to 100] ] [ , / PROGRESS_USERDATA=variable ] )

MESH_ISSOLID - Computes various mesh properties and enables IDL to determine if a mesh encloses space (is a solid).

Result = MESH_ISSOLID ( Conn )

MESH_MERGE - Merges two polygonal meshes.

Result = MESH_MERGE ( Verts [, Conn [, Conn1 [, /COMBINE_VERTICES ] ] ] , / JOIN_POLYMESH | , / DOUBLE ]

MESH_NUMTRIANGLES - Computes the number of triangles in a polygonal mesh.

Result = MESH_NUMTRIANGLES ( Conn )

MESH_OBJ - Generates a polygon mesh for various simple objects.


MESH_SMOOTH - Performs spatial smoothing on a polygon mesh.

Result = MESH_SMOOTH ( Verts [, Conn [, / ITERATIONS=value | , FIXED_VERTICES=variable ] ] , / FIXED_EDGE_VERTICES | , / LAMBDA=value ] )
MESH_SURFACEAREA - Computes various mesh properties to determine the mesh surface area, including integration of other properties interpolated on the surface of the mesh.

\[
\text{Result} = \text{MESH\_SURFACEAREA} ( \text{Verts, Conn} \\
[\text{, AUXDATA=Array} ] [\text{, MOMENT=variable} ] )
\]

MESH_VALIDATE - Checks for NaN values in vertices, removes unused vertices, and combines close vertices.

\[
\text{Result} = \text{MESH\_VALIDATE} ( \text{Verts, Conn} \\
[\text{, /REMOVE_NAN} ] [\text{, /PACK_VERTICES} ] \\
[\text{, /COMBINE_VERTICES} ] [\text{, TOLERANCE=value} ] )
\]

MESH_VOLUME - Computes the volume that the mesh encloses.

\[
\text{Result} = \text{MESH\_VOLUME} ( \text{Verts, Conn} \\
[\text{, /DOUBLE} ] [\text{, /SIGNED} ] )
\]

MESSAGE - Issues error and informational messages.

\[
\text{MESSAGE} ( [\text{[Text]}] [\text{[Arg, ... Arg]}] \\
[\text{, /REISSUE\_LAST} ] [\text{, BLOCK=BlockName} ] \\
[\text{, /CONTINUE} ] [\text{, /INFORMATIONAL} ] [\text{, /ERROR}] \\
[\text{, LEVEL=CallLevel} ] [\text{, NAME=ErrorName} ] \\
[\text{, /NONAME} ] [\text{, /NOPREFIX} ] [\text{, /NOPRINT} ] [\text{, /RESET}]
\]

MIN - Returns the value of the smallest element of an array.

\[
\text{Result} = \text{MIN\_Array} ( \text{[Min\_Subscript]} ) [\text{, /ABSOLUTE}] \\
[\text{, DIMENSION=value} ] [\text{, MAX=value} ] [\text{, /NAN}] \\
[\text{, SUBSCRIPT\_MAX} ] [\text{[Thread\_pool\_keywords]}]
\]

MIN\_CURVE\_SURF - Interpolates over either a plane or a sphere with a minimum curvature surface or a thin-plate-spline surface.

\[
\text{Result} = \text{MIN\_CURVE\_SURF} ( Z, X, Y ) [\text{, /DOUBLE}] \\
[\text{, /TPS} ] [\text{, /REGULAR} ] [\text{, /SHEFRE} ] [\text{, /CONST}] \\
[\text{, XGRID={xstart, xspacing}} ] [\text{, YVALUES=Array} ] \\
[\text{, YGRID={ystart, yspacing}} ] [\text{, YVALUES=Array} ] \\
[\text{, GS=[xspace, yspace]} ] [\text{, BOUNDS=[xmin, ymin, xmax, ymax]} ] \\
[\text{, NX=value} ] [\text{, NY=value} ] [\text{, XOUT=vector} ] \\
[\text{, YOUT=vector} ] [\text{, XPOUT=Array} ] [\text{, YPOUT=Array}] \\
\]

MK\_HTTP\_HELP - Converts text documentation headers to HTML files.

\[
\text{MK\_HTTP\_HELP} ( \text{Sources, Filename} ) [\text{, /STRITC}] \\
[\text{, TITLE=string} ] [\text{, /VERBOSE}]
\]

MODIFYCT - Saves modified color tables in the IDL color table file.

\[
\text{MODIFYCT} ( \text{Itab, Name, R, G, B} ) [\text{, FILE=filename}] \\
\]

MOMENT - Computes mean, variance, skewness, and kurtosis.

\[
\text{Result} = \text{MOMENT} ( \text{X} ) [\text{, /DOUBLE}] [\text{, MDEV=variable}] \\
[\text{, /NAN}] [\text{, SDEV=variable}] \\
\]

MORPH\_CLOSE - Applies closing operator to binary or grayscale image.

\[
\text{Result} = \text{MORPH\_CLOSE} ( \text{Image, Structure} [\text{, /GRAY}] \\
[\text{, PRESERVE\_TYPE=bytearray} ] [\text{, /UINT} ] [\text{, /ULONG}] \\
[\text{, VALUES=Array}] )
\]

MORPH\_DISTANCE - Estimates N-dimensional distance maps, which contain for each foreground pixel the distance to the nearest background pixel, using a given norm.

\[
\text{Result} = \text{MORPH\_DISTANCE} ( \text{Data} [\text{, /BACKGROUND}] [\text{, NEIGHBOR\_SAMPLING=\{1 | 2 | 3\}} ] [\text{, /NO\_COPY}] )
\]

MORPH\_GRADIENT - Applies the morphological gradient operator to a grayscale image.

\[
\text{Result} = \text{MORPH\_GRADIENT} ( \text{Image, Structure} [\text{, /GRAY}] \\
[\text{, PRESERVE\_TYPE=bytearray} ] [\text{, /UINT} ] [\text{, /ULONG}] \\
[\text{, VALUES=Array}] )
\]

MORPH\_HITORMISS - Applies the hit-or-miss operator to a binary image.

\[
\text{Result} = \text{MORPH\_HITORMISS} ( \text{Image, HitStructure, MissStructure} )
\]

MORPH\_OPEN - Applies the opening operator to a binary or grayscale image.

\[
\text{Result} = \text{MORPH\_OPEN} ( \text{Image, Structure} [\text{, /GRAY}] \\
[\text{, PRESERVE\_TYPE=bytearray} ] [\text{, /UINT} ] [\text{, /ULONG}] \\
[\text{, VALUES=Array}] )
\]

MORPH\_THIN - Performs a thinning operation on binary images.

\[
\text{Result} = \text{MORPH\_THIN} ( \text{Image, HitStructure, MissStructure} )
\]

MORPH\_TOPHAT - Applies top-hat operator to a grayscale image.

\[
\text{Result} = \text{MORPH\_TOPHAT} ( \text{Image, Structure} [\text{, /GRAY}] \\
[\text{, PRESERVE\_TYPE=bytearray} ] [\text{, /UINT} ] [\text{, /ULONG}] \\
[\text{, VALUES=Array}] )
\]

MPEG\_CLOSE - Closes an MPEG sequence.

\[
\text{MPEG\_CLOSE} ( \text{mpegID} )
\]

MPEG\_OPEN - Opens an MPEG sequence.

\[
\text{mpegID} = \text{MPEG\_OPEN} ( \text{Dimensions} \\
[\text{, BITRATE=value} ] [\text{, FILENAME=string}] \\
[\text{, IFRAME\_GAP=integer\_value}] \\
[\text{, MOTION\_VEC\_LENGTH=\{1 | 2 | 3\}}] \\
[\text{, QUALITY=value\{0 to 100\} }])
\]

MPEG\_PUT - Inserts an image array into an MPEG sequence.

\[
\text{MPEG\_PUT} ( \text{mpegID} [\text{, /COLOR}] [\text{, IMAGE=Array}] ) \\
[\text{, FRAME=frame\_number}] [\text{, WINDOW=index} ] [\text{, /ORDER}]
\]

MPEG\_SAVE - Encodes and saves an open MPEG sequence.

\[
\text{MPEG\_SAVE} ( \text{mpegID} [\text{, FILENAME=string}] )
\]

MULTI - Replicates current color table to enhance contrast.

\[
\text{MULTI} ( \text{N} )
\]

N\_ELEMENTS - Returns the number of elements contained in an expression or variable.

\[
\text{Result} = \text{N\_ELEMENTS(Expression)}
\]

N\_PARAMS - Returns the number of non-keyword parameters used in calling an IDL procedure or function.

\[
\text{Result} = \text{N\_PARAMS}()
\]

N\_TAGS - Returns the number of tags in a structure.

\[
\text{Result} = \text{N\_TAGS( Expression } [\text{, /LENGTH} ] )
\]
NCDF_* Routines - See “NetCDF Routines” on page 138.

NEwTON - Solves nonlinear equations using Newton’s method.
Result = NEwTON( X, Vecfunc [, CHECK=variable] 
[, /DOUBLE] [, ITMAX=value] [, STEPMAX=value] 
[, TOLF=value] [, TOLMIN=value] [, TOLX=value] )

NORM - Computes Euclidean norm of vector or Infinity norm of array.
Result = NORM( A [, /DOUBLE] 
[, L_NORM={0 | 1 | 2 | n} )

OBJ_CLASS - Determines the class name of an object.
Result = OBJ_CLASS( [Arg] [, COUNT=variable] 
[, /SUPERCLASS{must specify Arg}] )

OBJ_DESTROY - Destroys an object reference.
OBJ_DESTROY, ObjRef [, Arg 1, …, Arg n]

OBJ_ISA - Determines inheritance relationship of an object.
Result = OBJ_ISA( ObjectInstance, ClassName)

OBJ_NEW - Creates an object reference.
Result = OBJ_NEW( ObjectClassName [, Arg 1 …...Arg n] )

OBJ_VALID - Verifies validity of object references.
Result = OBJ_VALID( [Arg] [, /CAST] 
[, COUNT=variable] )

OBJJARR - Creates an array of object references.
Result = OBJJARR(D_1 [, ….D_n] [, /NOZERO] )

ON_ERROR - Designates the error recovery method.
ON_ERROR, N

ON_IOERROR - Declares an I/O error exception handler.
ON_IOERROR, Label
. . .
Label: Statement to perform upon I/O error

ONLINE_HELP - Invokes online help viewer from programs.
ONLINE_HELP [, Value] [, BOOK=filename] 
[, /FULL_PATH] [, /QUIT]
UNIX-Only Keywords: [, /FOLD_CASE]
[ , PAGE=pageno] [, /SUPPRESS_PLUGIN_ERRORS]
Windows-Only Keywords: [, /CONTEXT] [, /TOPICS]

OPEN - Opens files for reading, updating, or writing.
OPENR, Unit, File
OPENW, Unit, File
OPENU, Unit, File

Keywords (all platforms): [, /APPEND] [, /COMPRESS] 
[, BUFSIZE={0 | 1 | value>512}] [, /DELETE] 
[, ERROR=variable] [, /F77_UNDEFINED] 
[, /GET_LUN] [, /MORE] [, /NOEXPAND_PATH] 
[, /STDIO] [, /SWAP_ENDIAN]
[, SWAP_IF_BIG_ENDIAN] 
[, SWAP_IF_LITTLE_ENDIAN] [, /VAX_FLOAT] 
[, WIDTH=value] [, /XDR]
UNIX-Only Keywords: [, /RAWIO]

OPLOT - Plots vector data over a previously-drawn plot.
OPLOT, [X] Y [, MAX_VALUE=value] 
[, MIN_VALUE=value] [, NSUM=value] [, /POLAR] 
[, THICK=value]
Graphics Keywords: [, CLIP=[X_0, Y_0, X_1, Y_1] ]
[, COLOR=value] [, LINESTYLE={0 | 1 | 2 | 3 | 4 | 5}] 
[, /NOCLIP] [, PSYM=integer{0 to 10}] 
[, SYMSIZE=value] [, /T3D] [, ZVALUE=value {0 to 1}]

OPLOTErr - Draws error bars over a previously drawn plot.
OPLOTErr, [ X,] Y, Err [, PSym ]

P_CORRELATE - Computes partial correlation coefficient.
Result = P_CORRELATE(X, Y, C [, /DOUBLE])

PARTICLE_TRACE - Traces the path of a massless particle through a vector field.
PARTICLE_TRACE, Data, Seeds, Verts, Conn
[, Normals] [, MAX_ITERATIONS=value] 
[, ANISOTROPY= array] [, INTEGRATION={0 | 1}] 
[, SEED_NORMAL=vector] [, TOLERANCE=value] 
[, MAX_STEPSIZE=value] [, /UNIFORM]

PATH_CACHE - Used to control IDL’s path cache mechanism.
PATH_CACHE [, /CLEAR] [, /ENABLE] [, /REBUILD]

PATH_SEP - Returns the proper file path segment separator character for the current operating system.
Result = PATH_SEP( [, /PARENT_DIRECTORY] 
[, /SEARCH_PATH] )

PCOMP - Computes principal components/derived variables.
Result = PCOMP( A [, COEFFICIENTS=variable] 
[, /COVARIANCE] [, /DOUBLE] 
[, EIGENVALUES=variable] [, N_VARIABLES=value] 
[, /STANDARDIZE] [, VARIANCES=variable] )
PLOT - Plots vector arguments as X versus Y graphs.

PLOT, [X], Y [, MAX_VALUE=value] [, MIN_VALUE=value] [, NSUM=value] [, /POLAR]

Graphics Keywords: [ , BACKGROUND=color_index] [, CHARHEIGHT=integer]
[, CLIP=[X0, Y0, X1, Y1]] [, COLOR=value] [, /DATA] [, /DEVICE] [, /NORMAL] [, FONT=integer]
[, LINESSTYLE=(0 | 1 | 2 | 3 | 4 | 5)] [, /NOCLIP]
[, /NODATA] [, /NOERASE] [, POSITION=[X0, Y0, X1, Y1]] [, PSYM=integer[0 to 10]] [, SUBTITLE=string]
[, SYMSIZE=value] [, /T3D] [, THICK=value]
[, TICKLEN=value] [, TITLE=string]
[, X | Y | Z] [CHARSIZE=value]
[, X | Y | Z] [GRIDSTYLE=integer[0 to 5]]
[, X | Y | Z] [MARGIN=left, right]
[, X | Y | Z] [MINOR=integer]
[, X | Y | Z] [RANGE=[min, max]]
[, X | Y | Z] [STYLE=value] [, X | Y | Z] [THICK=value]
[, X | Y | Z] [TICK_GET=variable]
[, X | Y | Z] [TICKFORMAT=string]
[, X | Y | Z] [TICKINTERVAL=value]
[, X | Y | Z] [TICKLAYOUT=scalar]
[, X | Y | Z] [TICKLEN=value]
[, X | Y | Z] [TICKNAME=string_array]
[, X | Y | Z] [TICKS=integer]
[, X | Y | Z] [TICKUNITS=string]
[, X | Y | Z] [TICKV=vector]
[, X | Y | Z] [TICK_UNITS=string]
[, X | Y | Z] [TITLE=string]
[, ZVALUE=value[0 to 1]]

PLOT_3DBOX - Plots function of two variables inside 3D box.

PLOT_3DBOX, X, Y, Z [, GRIDSTYLE=(0 | 1 | 2 | 3 | 4 | 5)] [, PSYM=integer[1 to 10]] [, /SOLID_WALLS]
[, /XY_PLANE] [, XSTYLE=(0 | 1 | 2 | 3 | 4 | 5)]
[, /YZ_PLANE] [, YSTYLE=(0 | 1 | 2 | 3 | 4 | 5)]
[, /ZVOLUME] [, ZSTYLE=(0 | 1 | 2 | 3 | 4 | 5)]
[, AX=degrees] [, AZ=degrees] [, ZAXIS=(1 | 2 | 3 | 4)]

Graphics Keywords: Accepts all graphics keywords accepted by PLOT except for: FONT, PSYM, SYMSIZE, X/Y/Z TICK_GET, and ZVALUE.

PLOT_FIELD - Plots a 2D field using arrows.

PLOT_FIELD, U, V [, ASPECT=ratio]
[, LENGTH=value] [, N=number_arrows] [, TITLE=string]

PLOTERR - Plots individual data points with error bars.

PLOTERR, [X,] Y [, Err ] [, TYPE=[1 | 2 | 3 | 4]]
[, PSYM=integer[1 to 10]]

PLOTS - Plots vectors and points.

PLOTS, X, Y, Z [, (CONTINUE)]

Graphics Keywords: [ , CLIP=[X0, Y0, X1, Y1]]
[, COLOR=value] [, /DATA] [, /DEVICE] [, /NORMAL]
[, LINESSTYLE=(0 | 1 | 2 | 3 | 4 | 5)] [, /NOCLIP]
[, PSYM=integer[0 to 10]] [, SYMSIZE=value] [, /T3D]
[, THICK=value] [, Z=value]

PNT_LINE - Returns the perpendicular distance between a point and a line.

Result = PNT_LINE( P0, L0, LI [, Pl ] [, /INTERVAL ] )

POINT_LUN - Sets or gets current position of the file pointer.

POINT_LUN, Unit, Position

POLAR_CONTOUR - Draws a contour plot from data in polar coordinates.

POLAR_CONTOUR, Z, Theta, R
[, C_ANNOTATION=vector_of_strings]
[, C_CHARSIZE=value] [, C_CHARTHICK=integer]
[, C_COLORS=vector] [, C_LINESSTYLE=vector]
[, /FILL] [, CELL_FILL] [, C_ORIENTATION=degrees]
[, C_SPACING=value] [, C_THICK=vector]
[, /CLOSED] [, /IRREGULAR] [, LEVELS=vector]
[, NLEVELS=integer[1 to 29]] [, MAX_VALUE=value]
[, MIN_VALUE=value] [, /OVERPLOT]
[, /PATH_DATA_COORDS]
[, _TRIANGULATION=variable] [, /XLOG] [, /YLOG]
[, /XAXIS] [, /SHOW_TRIANGULATION=color_index]

POLAR_SURFACE - Interpolates a surface from polar coordinates to rectangular coordinates.

Result = POLAR_SURFACE( Z, Theta, R [, /GRID]
[, SPACING=xspacing, yspacing] [, BOUNDS=[x 0, x 1, y 0, y 1]] [, /QUINTIC]
[, MISSING=value] )

POLY - Evaluates polynomial function of a variable.

Result = POLY(X, C)

POLY_2D - Performs polynomial warping of images.

Result = POLY_2D( Array, P, Q [, Interp [, Dimx, Dimy]]
[, CUBIC=-1 to 0]] [, MISSING=value] [, Thread pool keywords])

POLY_AREA - Returns the area of a polygon given the coordinates of its vertices.

Result = POLY_AREA( X, Y [, /DOUBLE] [, /SIGNED] )

POLY_FIT - Performs a least-square polynomial fit.

Result = POLY_FIT( X, Y, Degree [, CHISQ=variable]
[, COVAR=variable] [, /DOUBLE]
[, MEASURE_ERRORS=vector] [, SIGMA=variable]
[, STATUS=variable] [, YBAND=variable]
[, YERROR=variable] [, YFIT=variable] )

POLYFILL - Fills the interior of a polygon.

POLYFILL, X [, Y [, Z]] [, IMAGE_COORD=variable]
[, /IMAGE_INTERP] [, /LINE_FILL]
[, PATTERN=variable] [, SPACING=centimeters]
[, TRANSPARENT=variable]

Graphics Keywords: [ , CLIP=[X0, Y0, X1, Y1]]
[, COLOR=value] [, /DATA] [, /DEVICE] [, /NORMAL]
[, LINESSTYLE=(0 | 1 | 2 | 3 | 4 | 5)] [, /NOCLIP]
[, ORIENTATION=ccw_degrees_from_horiz] [, /T3D]
[, THICK=value] [, Z=value]
POLYFILLV - Returns subscripts of pixels inside a polygon.
   \[ \text{Result} = \text{POLYFILLV}(X, Y, SzX, SzY, \{ \text{Run:\_Length} \} ) \]

POLYSHADE - Creates a shaded surface representation from a set of polygons.
   \[ \text{Result} = \text{POLYSHADE}( \text{Vertices}, \text{Polygons} ) \]
   or
   \[ \text{Result} = \text{POLYSHADE}(X, Y, Z, \text{Polygons} ) \]
   
   Keywords: [\{DATA\} [, \{NORM\}]
   \[ \{ \text{POLY\_SHADES=} \text{array} \} [, \{ \text{SHADES=} \text{array} \} [, /3D]\]
   \[ \{ \text{TOP=} \text{value} \} [, \{ \text{XSIZE=} \text{columns} \} [, \{ \text{YSIZE=} \text{rows} \} ] \]

POLYWARP - Performs polynomial spatial warping.
   \[ \text{POLYWARP}(X, \text{Xi}, Xo, \text{Yi}, Yo, \text{Degree}, Kx, Ky [, /DOUBLE] [, \{ STATUS=} \text{variable} ] \)

POPD - Removes the top directory on the working directory stack maintained by PUSHD/POPD.
   \[ \text{POPD}() \]

POWELL - Computes the first \( K \) prime numbers.
   \[ \text{PRIMES}(K) \]

PRINT/PRINTF - Writes formatted output to screen or file.
   \[ \text{PRINT}(), \ldots, \text{Expr}_n \]
   \[ \text{PRINTF}() [, \text{Unit} [, \text{Expr}_1, \ldots, \text{Expr}_n] ] \]
   
   Keywords: [\{AM\_PM=} \{string, string\}]
   \[ [, \{ \text{DAYS\_OF\_WEEK=} \text{array} \{ 7 \text{ names} \} ] [, \{ \text{FORMAT=} \text{value} \} [, \{ \text{MONTHS=} \text{string\_array} \{ 12 \text{ names} \} [, \{ \text{STD\_IO\_NON\_FINITE\} } ] \]

PRINTD - Prints contents of the directory stack maintained by PUSHD/POPD.
   \[ \text{PRINTD}() \]

PRIM - Defines a procedure.
   \[ \text{PRO}() \]
   \[ \text{Procedure}\_Name, \text{argument}_1, \ldots, \text{argument}_n \]
   \[ \ldots \]
   \[ \text{END} \]

PRODUCT - Returns the product of elements within an array.
   \[ \text{RESULT} = \text{PRODUCT}(\text{Array} [, \{ \text{Dimension} \} [, \{ \text{CUMULATIVE}\} [, \{ \text{INTEGER}\} [, \{ \text{NAN}\} [, \{ \text{PRESERVE\_TYPE}\} ] ] ] ] ] ] \]

PROFILE - Extracts a profile from an image.
   \[ \text{PROFILE}(\text{Image} [, \{ \text{XX, YY}\} ] [, \{ \text{NOMARK}\} [, \{ \text{XSTART=} \text{value} \} [, \{ \text{YSTART=} \text{value} \} [, \{ \text{XSIZE=} \text{columns} \} [, \{ \text{YSIZE=} \text{rows} \} ] ] ] ] ] \]

PROFILES - Interactively examines image profiles.
   \[ \text{PROFILES}(\text{Image} [, \{ \text{ORDER}\} [, \{ \text{SX=} \text{value} \} [, \{ \text{SY=} \text{value} \} [, \{ \text{WSIZE=} \text{value} \} ] ] ] ] ] \]

PROJECT_VOL - Returns a translucent rendering of a volume projected onto a plane.
   \[ \text{RESULT} = \text{PROJECT\_VOL}(\text{Vol}, \{ \text{X\_Sample}, \text{Y\_Sample}, \text{Z\_Sample} \} [, \{ \text{AVG\_INTENSITY}\} [, \{ \text{CUBIC}\} [, \{ \text{DEPTH\_Q=} \text{value} \} [, \{ \text{OPAQUE=} \text{3D\_array}\} [, \{ \text{TRANS=} \text{array}\} [, \{ \text{XSIZE=} \text{long\_word\_integer} \} [, \{ \text{YSIZE=} \text{long\_word\_integer} \} [, \{ \text{Z\_BUFFER}\} ] ] ] ] ] ] ] ] ] \]

PS_SHOW_FONTS - Displays all the PostScript fonts that IDL knows about.
   \[ \text{PS\_SHOW\_FONTS}() [, \{ \text{NOLATIN}\} ] \]

PSAFM - Converts Adobe Font Metrics file to IDL format.
   \[ \text{PSAFM}(\text{Input\_Filename}, \text{Output\_Filename}) \]

PSEUDO - Creates a pseudo-color table based on Lightness, Hue, and Brightness system.
   \[ \text{PSEUDO}(\text{Lilto}, \text{Litli}, \text{Solid}, \text{Sathi}, \text{Hue}, \text{Loops} [, \text{Colr}]) \]

PTR_FREE - Destroys a pointer.
   \[ \text{PTR\_FREE}(P_1, \ldots, P_n) \]

PTR_NEW - Creates a pointer.
   \[ \text{RESULT} = \text{PTR\_NEW}(\{ \text{InitExpr} \} [, \{ \text{ALLOCATE\_HEAP}\} [, \{ \text{NO\_COPY}\} ] ] ] ] \]

PTR_VALID - Verifies the validity of pointers.
   \[ \text{RESULT} = \text{PTR\_VALID}(\text{Arg} [, \{ \text{CAST}\} [, \{ \text{COUNT=} \text{variable} \} ] ] ] ] \]

PTRARR - Creates an array of pointers.
   \[ \text{RESULT} = \text{PTRARR}(D_1 [, \ldots, D_k] [, \{ \text{ALLOCATE\_HEAP}\} [, \{ \text{NO\_ZERO}\} ] ] ] ] \]

PUSHD - Pushes a directory to top of directory stack maintained by PUSHD/POPD.
   \[ \text{PUSHD}() \]
   \[ \text{PUSHD}(\text{Dir}) \]

Q

QGRID3 - Interpolates the dependent variable values to points in a regularly sampled volume.
   \[ \text{RESULT} = \text{QGRID3}(\text{XYZ}, \text{F}, \text{Tetrahedra} ) \]
   or
   \[ \text{RESULT} = \text{QGRID3}(\text{X, Y, Z, F, Tetrahedra} ) \]
   \[ \text{KEYWORDS:} [, \{ \text{DELTA=} \text{array} \} [, \{ \text{DIMENSION=} \text{array} \} [, \{ \text{MISSING=} \text{value} \} [, \{ \text{START=} \text{array} \} ] ] ] ] \]

QHULL - Constructs convex hulls, Delaunay triangulations, and Voronoi diagrams.
   \[ \text{QHULL}(\text{V}, \text{Tr} \\ or \text{QHULL}, \text{V}, \text{V1}, \{ \text{V2} \ldots, \text{V6} \} , \text{Tr} \\ [, \{ \text{BOUNDS=} \text{variable} \} [, \{ \text{CONNECTIVITY=} \text{variable} \} [, \{ \text{DELAUNAY}\} [, \{ \text{SPHERE=} \text{variable} \} [, \{ \text{VDIAGRAM=} \text{array}\} [, \{ \text{VNORMALS=} \text{array}\} [, \{ \text{VERTICES=} \text{array}\} ] ] ] ] ] ] ] ] \]

IDL Quick Reference

QHULL
QROMB - Evaluates integral over a closed interval.

Result = QROMB( Func, A, B [, /DOUBLE] [, EPS=value] [, JMAX=value] [, K=value] )

QROMO - Evaluates integral over an open interval.


QSIMP - Evaluates integral using Simpson’s rule.

Result = QSIMP( Func, A, B [, /DOUBLE] [, EPS=value] [, JMAX=value] )

QUERY_BMP - Obtains information about a BMP image file.

Result = QUERY_BMP ( Filename [, Info] )


Result = QUERY_BMP2000(Filename [, Info] )

QUERY_MRSID - Obtains information about a MrSID image file.

Result = QUERY_MRSID( Filename [, Info] )

QUERY_PICT - Obtains information about a PICT image file.

Result = QUERY_PICT ( Filename, Info )

QUERY_PNG - Obtains information about a PNG image file.

Result = QUERY_PNG ( Filename [, Info] )

QUERY_PPM - Obtains information about a PPM image file.

Result = QUERY_PPM ( Filename [, Info] [, MAXVAL=variable] )

QUERY_SRF - Obtains information about an SRF image file.

Result = QUERY_SRF ( Filename [, Info] )

QUERY_TIFF - Obtains information about a TIFF image file.

Result = QUERY_TIFF ( Filename [, Info] [, GEOTIFF=variable] [, IMAGE_INDEX=index] )

QUERY_WAV - Obtains information about a WAV sound file.

Result = QUERY_WAV ( Filename [, Info] )

R

R_CORRELATE - Computes rank correlation.


R_TEST - Runs test for randomness.

Result = R_TEST( X [, N0=variable] [, N1=variable] [, R=variable] )

RADON - Returns the Radon transform of a two-dimensional image.


RANDOM - Returns normally-distributed pseudo-random numbers.


RANDOMU - Returns uniformly-distributed pseudo-random numbers.


RANKS - Computes magnitude-based ranks.

Result = RANKS(X)

RDPIX - Interactively displays image pixel values.

RDPIX, Image [, X0, Y0]

READ/READF - Reads formatted input from keyboard or file.

READ, [Prompt] Var_j, ..., Var_n

READF, [Prompt] Unit, Var_j, ..., Var_n

Keywords: [ , AM_PM=string, string ]

[ , DAYS_OF_WEEK=string_array [7 names] ]

[ , FORMAT=variable ]

[ , MONTHS=string_array [12 names] ]

[ , PROMPT=string ]

READ_ASCII - Reads data from an ASCII file.

<table>
<thead>
<tr>
<th>Alphabetical List of IDL Routines</th>
<th>57</th>
</tr>
</thead>
</table>

**READ_BINARY** - Reads the contents of a binary file using a passed template or basic command line keywords.  
\[
\text{Result} = \text{READ_BINARY}(\{\text{Filename} | \text{FileUnit} [\[, \text{TEmplate} = \text{template} \] \[, \text{DATA}_{-}\text{START} = \text{value} \] \[, \text{DATA}_{-}\text{TYPE} = \text{typecodes} \] \[, \text{DATA}_{-}\text{ENDIAN} = \text{string} \])
\]

**READ_BMP** - Reads Microsoft Windows bitmap file (.BMP).  
\[
\text{Result} = \text{READ_BMP}(\text{Filename} [, \text{R}, \text{G}, \text{B}] [\[, \text{Ihdr} \] \[, \text{RGB} \])
\]

**READ_DICOM** - Reads an image from a DICOM file.  
\[
\text{Result} = \text{READ_DICOM}(\text{Filename} [, \text{Red}, \text{Green}, \text{Blue}] [\[, \text{Image}_{-}\text{INDEX} = \text{index} \])
\]

**READ_IMAGE** - Reads the image contents of a file and returns the image in an IDL variable.  
\[
\text{Result} = \text{READ_IMAGE}(\text{Filename} [, \text{Red}, \text{Green}, \text{Blue}] [\[, \text{Image}_{-}\text{INDEX} = \text{index} \])
\]

**READ_INTERFILE** - Reads Interfile (v3.3) file.  
\[
\text{Reads Interfile (v3.3) file.}
\]

**READ_JPEG** - Reads JPEG file.  
\[
\text{READ_JPEG}(\text{Filename} [, \text{UNIT} = \text{un}], \text{Image} \[, \text{Color} \text{table} \] \[, \text{BUFFER} = \text{variable} \] \[, \text{COLORS} = \text{value} \{8 to 256\} \] \[, \text{DITHER} = [0 | 1 | 2] \] \[, \text{GRAYSCALE} \] \[, \text{ORDER} \] \[, \text{TRUE} = [1 | 2 | 3] \] \[, \text{TWO}_{-}\text{PASS}_{-}\text{QUANTIZE} \])
\]

\[
\text{Result} = \text{READ_JPEG2000}(\text{Filename} [, \text{Red}, \text{Green}, \text{Blue}] [\[, \text{DISCARD}_{-}\text{LEVELS} = \text{value} \] \[, \text{MAX}_{-}\text{LAYER} = \text{value} \] \[, \text{ORDER} \] \[, \text{REGION} = \{\text{Start}X, \text{Start}Y, \text{Width}, \text{Height} \})
\]

**READ_MRSID** - Reads MrSID file.  
\[
\text{Result} = \text{READ_MRSID}(\text{Filename} [, \text{LEVEL} = \text{dvl}] [\[, \text{SUB}_{-}\text{RECT} = \text{rect} \])
\]

**READ_PICT** - Reads Macintosh PICT (version 2) bitmap file.  
\[
\text{READ_PICT}(\text{Filename}, \text{Image} [, \text{R}, \text{G}, \text{B}]
\]

\[
\text{Result} = \text{READ_PNG}(\text{Filename} [, \text{R}, \text{G}, \text{B}] [\[, \text{ORDER} \] \[, \text{VERBOSE} \] \[, \text{TRANS} \text{TRANSPARENT} = \text{variable} \])
\]

**READ_PPM** - Reads PGM (gray scale) or PPM (portable pixmap for color) file.  
\[
\text{READ_PPM}(\text{Filename}, \text{Image} [, \text{MAX} \text{VAL} = \text{variable}]
\]

**READ_SPR** - Reads a row-indexed sparse matrix from a file.  
\[
\text{Result} = \text{READ_SPR}(\text{Filename})
\]

**READ_SRF** - Reads Sun Raster Format file.  
\[
\text{READ_SRF}(\text{Filename}, \text{Image} [, \text{R}, \text{G}, \text{B}]
\]

**READ_SYLK** - Reads Symbolic Link format spreadsheet file.  
\[
\text{Result} = \text{READ_SYLK}(\text{File} [, \text{ARRAY} \] \[, \text{COLMAJOR} \] \[, \text{NCOLOM} = \text{columns} \] \[, \text{NROWS} = \text{rows} \] \[, \text{START}_{-}\text{COL} = \text{column} \] \[, \text{START}_{-}\text{ROW} = \text{row} \] \[, \text{USE}_{-}\text{DOUBLES} \] \[, \text{USE}_{-}\text{LONGS} \])
\]

**READ_TIFF** - Reads TIFF format file.  
\[
\text{Result} = \text{READ_TIFF}(\text{Filename} [, \text{R}, \text{G}, \text{B}] [\[, \text{CHANNELS} = \text{scalar} \text{or} \text{vector} \] \[, \text{DOT}_{-}\text{RANGE} = \text{variable} \] \[, \text{GEOTEFF} = \text{variable} \] \[, \text{IMAGE}_{-}\text{INDEX} = \text{variable} \] \[, \text{ICC}_{-}\text{PROFILE} = \text{variable} \] \[, \text{INTERL} = \{0 | 1 | 2\} \] \[, \text{ORIENTATION} = \text{variable} \] \[, \text{PHOTOSHOP} = \text{variable} \] \[, \text{PLANARCON} = \text{variable} \] \[, \text{SUB}_{-}\text{RECT} = \{x, y, \text{width}, \text{height} \}\] \[, \text{UNSIGNED} \] \[, \text{VERBOSE} \])
\]

**READ_WAV** - Reads the audio stream from the named .WAV file.  
\[
\text{Result} = \text{READ_WAV}(\text{Filename} [, \text{Rate}])
\]

**READ_WAVE** - Reads Wavefront Advanced Visualizer file.  
\[
\text{READ_WAVE}(\text{File}, \text{Variables}, \text{Names}, \text{Dimensions} [\[, \text{MESH}_{-}\text{NAMES} = \text{variable} \])
\]

\[
\text{READ_X11_BITMAP}(\text{File}, \text{Bitmap} [, \text{X}, \text{Y}] [\[, \text{EXP} \text{AND} \text{TO} \text{BYTES} \])
\]

**READ_XWD** - Reads X Windows Dump file.  
\[
\text{Result} = \text{READ_XWD}(\text{Filename} [, \text{R}, \text{G}, \text{B}])
\]

**READS** - Reads formatted input from a string variable.  
\[
\text{READS}(\text{Input}, \text{Var}_1, \ldots, \text{Var}_n [\[, \text{AM}_{-}\text{PM} = [\text{string}, \text{string}] \] \[, \text{DAYS} \_\text{OF} \_\text{WEEK} = \text{string}_{\_\text{array}} \{7 \text{ names} \} \] \[, \text{FORMAT} = \text{value} \] \[, \text{MONTH}_{-}\text{S} = \text{string}_{\_\text{array}} \{12 \text{ names} \}] \]
\]

**REDU** - Reads unformatted binary data from a file.  
\[
\text{REDU}(\text{Unit}, \text{Var}_1, \ldots, \text{Var}_n [\[, \text{TRANSFER}_{-}\text{C}OUNT = \text{variable} \])
\]

**REAL_PART** - Returns the real part of a complex-valued argument.  
\[
\text{Result} = \text{REAL}_{-}\text{PART}(z)
\]

**REBIN** - Resizes a vector or array by integer multiples.  
\[
\text{Result} = \text{REBIN}(\text{Array}, D_{x}, \ldots, D_{z} [\[, \text{SAMPLE} \])
\]

**RECALL_COMMANDS** - Returns entries in IDL’s command recall buffer.  
\[
\text{Result} = \text{RECALL}_{-}\text{COMMANDS}()
\]

**RECON3** - Reconstructs a 3D representation of an object from 2D images.  
\[
\text{Result} = \text{RECON3}(\text{Images}, \text{Obj}_{-}\text{Rot}, \text{Obj}_{-}\text{Pos}, \text{Focal}, \text{Dist}, \text{Vol}_{-}\text{Pos}, \text{Im}_{-}\text{Ref}, \text{Im}_{-}\text{Mag}, \text{Vol}_{-}\text{Size} [\[, \text{CUBIC} \] \[, \text{MISSING} = \text{variable} \] \[, \text{MODE} = \text{value} \] \[, \text{QUIET} \])
\]

**REDUCE_COLORS** - Reduces the number of colors used in an image by eliminating unused pixel values.  
\[
\text{REDUCE}_{-}\text{COLORS}(\text{Image}, \text{Values})
\]
### IDL Quick Reference

#### Alphabetical List of IDL Routines

**REFORM** - Changes array dimensions without changing the total number of elements.
- `Result = REFORM( Array, D1 [,...,Dk] [, /OVERWRITE] )`

**REGION_GROW** - Perform region growing.
- `Result = REGION_GROW( Array, ROIPixels [, /ALL_NEIGHBORS] [, /NAN] [, STDDEV_MULTIPLIER=value] [, THRESHOLD=[min,max]] )`

**REGISTER_CURSOR** - Associates the given name with the given cursor information.
- `REGISTER_CURSOR, Name[, Image], MASK=value [, HOTSPOT=value] [, /OVERWRITE]`

**REGRESS** - Computes fit using multiple linear regression.

**REPEAT...UNTIL** - Repeats statement(s) until expression evaluates to true. Subject is always executed at least once.
- `REPEAT statement UNTIL expression`
  - or
- `REPEAT BEGIN statements ENDREP UNTIL expression`

**REPLICATE** - Creates an array of given dimensions, filled with specified value.
- `Result = REPLICATE( Value, D1 [,...,Dk] [, Thread pool keywords] )`

**REPLICATE_INPLACE** - Updates an array by replacing all or selected parts of it with a specified value.
- `REPLICATE_INPLACE, X, Value [, D1, Loc1 [,...,Dk, Range]] [, Thread pool keywords]`

**RESOLVE_ALL** - Compiles any uncompiled routines.
- `RESOLVE_ALL [, CLASS=string] [, /CONTINUE_ON_ERROR] [, /QUIET]`

**RESOLVE_ROUTINE** - Compiles a routine.

**RESTORE** - Restores IDL variables and routines saved in an IDL SAVE file.

**RETAIL** - Returns control to the main program level.
- `RETAIL`

**RETURN** - Returns control to the next-higher program level.
- `RETURN [, Return_value]`

**REVERSE** - Reverses the order of one dimension of an array.
- `Result = REVERSE( Array [, Subscript_Index] [, /OVERWRITE] )`

**RK4** - Solves differential equations using fourth-order Runge-Kutta method.
- `Result = RK4( Y, Dydx, X, H, Derivs [, /DOUBLE] )`

**ROBERTS** - Returns an approximation of Roberts edge enhancement.
- `Result = ROBERTS(Image)`

**ROT** - Rotates an image by any amount.
- `Result = ROT(A, Angle, [Mag, X0, Y0] [, /INTERP] [, CUBIC=value{-1 to 0}] [, /MISSING=value] [, /PIVOT])`

**ROTATE** - Rotates/transposes an array in multiples of 90 degrees.
- `Result = ROTATE(Array, Direction)`

**ROUND** - Rounds the argument to its closest integer.
- `Result = ROUND( X [, /L64] [, Thread pool keywords] )`

**ROUTINE_INFO** - Provides information about compiled procedures and functions.

**RS_TEST** - Performs the Wilcoxon Rank-Sum test.
- `Result = RS_TEST(X, Y [, UX=variable] [, UY=variable] )`

**S**

**S_TEST** - Performs the Sign test.
- `Result = S_TEST( X, Y [, ZDIFF=variable] )`

**SAVGOL** - Returns coefficients of Savitzky-Golay smoothing filter.
- `Result = SAVGOL( Nleft, Nright, Order, Degree [, /DOUBLE] )`

**SAVE** - Saves variables, system variables, and IDL routines in a file for later use.

**SCALE3** - Sets up axis ranges and viewing angles for 3D plots.

**SCALE3D** - Scales 3D unit cube into the viewing area.
- `SCALE3D`
Alphabetical List of IDL Routines

59

SHMDEBUG • Print debugging information when a variable loses reference to an underlying shared memory segment. Result = SHMDEBUG(Enable)

SHMMAP • Maps anonymous shared memory, or local disk files, into the memory address space of the currently executing IDL process. SHMMAP [


SHMUNMAP • Removes a memory segment previously created by SHMMAP from the system. SHMUNMAP, SegmentName

IDL Quick Reference

SHIFT • Shifts elements of vectors or arrays by a specified number of elements. Result = SHIFT(Array, S1 [, ... , Sn])

SHADE_VOLUME • Contours a volume to create a list of vertices and polygons that can be displayed using POLYSHADE. SHADE_VOLUME, Volume, Value, Vertex, Poly

[ /LOW] [SHADES=Array] [ /VERBOSE]

[ /X RANGE=vector] [ /Y RANGE=vector]

[ /Z RANGE=vector]

SHADE_SURF • Creates a shaded-surface representation of a gridded dataset. SHADE_SURF, Z [, X, Y] [, AX=degrees] [AZ=degrees] IMAGE=value] [MAX_VALUE=value]

[MIN_VALUE=value] [PIXELS=pixels] [SAVE]

[SHADES=Array] [ /XLOG] [ /YLOG]

Graphics Keywords: [ CHAR_SIZE=value] [COLOR=integer] [DEVICE [, Normalize] [FON T=integer] [ /NODATA] [ POSITION=[X0, Y0, X1, Y1]]

[SUBTITLE=string] [ /T3D] [ /THICK=value]

[ /TICKLEN=value] [ /TITLE=string]

[X \ Y \ Z] CHAR SIZE=value

[X \ Y \ Z] GRIDSTYLE=value [0 to 5]

[X \ Y \ Z] MARGIN=[left, right]

[X \ Y \ Z] MINOR=integer

[X \ Y \ Z] RANGE=[min, max]

[X \ Y \ Z] STYLE=value

[X \ Y \ Z] THICK=value

[X \ Y \ Z] TICK_FORMAT=string

[X \ Y \ Z] TICK_INTERVAL=value

[X \ Y \ Z] TICK_LAYOUT=scalar

[X \ Y \ Z] TICKLEN=value

[X \ Y \ Z] TICKNAME=string_array

[X \ Y \ Z] TICKS=integer

[X \ Y \ Z] TICK_UNITS=string

[X \ Y \ Z] TICK_GET=variable

[X \ Y \ Z] TITLE=string

[Z_VALUE=value] [0 to 1]

SHADE_SURF_IRR • Creates a shaded-surface representation of an irregularly gridded dataset. SHADE_SURF_IRR, Z, X, Y, [AX=degrees]

[AZ=degrees] IMAGE=value] [PLIST=variable]

/T3D]

SEARCH3D • Finds “objects” or regions of similar data values within a 2D array. Result = SEARCH3D( Array, Xpos, Ypos, Min_Val, Max_Val [, /DECREASE, INCREASE]

[LPF_BAND={0 to 3}] [{X | Y | Z}MINOR=value] [{X | Y | Z}GRIDSTYLE=value] [{X | Y | Z}RANGE=value]

[{X | Y | Z}STYLE=value] [{X | Y | Z}THICK(value]

[{X | Y | Z}TICKLING=value]

[{X | Y | Z}TICKNAME=string_array

[{X | Y | Z}TICKS=integer

[{X | Y | Z}TICK_UNITS=string

[{X | Y | Z}TICK_GET=variable

[{X | Y | Z}TITLE=string

[{Z_VALUE=value] [0 to 1}]

SET_PLOT • Sets the output device used by the IDL direct graphics procedures. SET_PLOT, Device [, /COPY] [, /INTERPOLATE]

SET_SHADING • Sets the light source shading parameters. SET_SHADING [, /GOURAUD] [, LIGHT={x, y, z}]

[ /REJECT] [{VALUES={darkest, brightest}}]

SETENV • Adds or changes an environment variable. SETENV, Environment_Name

SET_PLOT, Device [, /COPY] [, /INTERPOLATE]

SET_SHADING [, /GOURAUD] [, LIGHT={x, y, z}]

[ /REJECT] [{VALUES={darkest, brightest}}]

SETENV • Adds or changes an environment variable. SETENV, Environment_Name


[ /APP_KEYPAD] [, /NUM_KEYPAD]

SHIFT • Performs polynomial fit to a surface. Result = SFit( Data, Degree

[ /ARREGULAR, KK=variable, /MAX DEGREE]

SHADE_SURF • Creates a shaded-surface representation of gridded data. SHADE_SURF, Z [, X, Y] [, AX=degrees] [AZ=degrees]

[IMAGE=value] [MAX_VALUE=value]

[MIN_VALUE=value] [PIXELS=pixels] [SAVE]

[SHADES=Array] [, /XLOG] [, /YLOG]

Graphics Keywords: CHAR_SIZE=value] [COLOR=integer] [, /DATA] [, /DEVICE] [, /NORMAL] [, FONT=integer]

[ /NODATA] [, POSITION=[X0, Y0, X1, Y1]]

[SUBTITLE=string] [, /T3D] [, /THICK=value]
SHMVAR - Creates an IDL array variable that uses the memory from a current mapped memory segment created by the SHMMAP procedure.


SHOW3 - Displays array as image, surface plot, and contour plot simultaneously.


SHOWFONT - Displays a TrueType or vector font

SHOWFONT, Font, Name [, /ENCAPSULATED] [, /TT_FONT] ()

SIMPLEX - Use the simplex method to solve linear programming problems.

Result = SIMPLEX( Equation, Constraints, M1, M2, M3 [, Tableau [, Izrow [, Iposv] ]] [, /DOUBLE] ()

SIN - Returns the trigonometric sine of X.

Result = SIN(X) 

SINDGEN - Returns a string array with each element set to its subscript.

Result = SINDGEN(D1 [, ...Dk]) ()

SINH - Returns the hyperbolic sine of X.

Result = SINH(X) [Thread pool keywords] ()

SIZE - Returns array size and type information.

Result = SIZE( Expression [, /L64] [, /DIMENSIONS [, /FILE_LUN [, /FILE_OFFSET], /N_ELEMENTS | , /N_DIMENSIONS | , /N_ELEMENTS | , /N_DIMENSIONS | , /FILENAME | , /STRUCTURE | , /NAME | , /TYPE | )]

SKEWNESS - Computes statistical skewness of an n-element vector.

Result = SKEWNESS(X [, /DOUBLE] [, /NAN]) ()

SKIP_LUN - Reads data in an open file and moves the file pointer.

SKIP_LUN, FromUnit, [ , Num] [, /EOF] [, /LINES] [, /TRANSFER_COUNT=variable] ()

SLICER3 - Interactive volume visualization tool.

SLICER3 [, hData3D] [, DATA_NAME=string[string_array]] [, /DETACH] [, GROUP=widget_id] [, /MODAL]

SLIDE_IMAGE - Creates a scrolling graphics window for examining large images.


SMOOTH - Smooths with a boxcar average.

Result = SMOOTH( Array, Width [, /EDGE_TRUNCATE] [, MISSING=value] [, /NAN])

SOBEL - Returns an approximation of Sobel edge enhancement.

Result = SOBEL(Image)

SOCKET - Opens client-side TCP/IP Internet socket as IDL file unit.


UNIX-Only Keywords: [, /STDOIO]

SORT - Returns indices of an array sorted in ascending order.

Result = SORT(Array [, /L64])

SPAWN - Spawns child process for access to operating system.

SPAWN [, Command, [ , Result], [ , ErrResult]] ()

Keywords (all platforms): [, COUNT=variable] [, EXIT_STATUS=variable] [, PID=variable]

UNIX-Only Keywords: [, /NOSHELL] [, /NOTTYRESET] [, /NULL_STDIN] [, /SH] [, /STDERR] [, /UNIT=variable {Command required, Result not allowed}]


SPH_4PNT - Returns center and radius of a sphere given 4 points.

SPH_4PNT, X, Y, Z, Xc, Yc, Zc, R [, /DOUBLE]

SPH_SCAT - Performs spherical gridding.

Result = SPH_SCAT(Lon, Lat, F [, BOUNDS=[lonmin, latmin, lonmax, latmax]] [, /BOUND=value] [, GOUT=variable] [, GS=[lonspacing, latspacing]] [, NLO=variable] [, NLAT=variable])

SPHER_HARM - Returns value of the spherical harmonic function.

Result = SPHER_HARM( Theta, Phi, L, M [, /DOUBLE])

SPL_INIT - Establishes the type of interpolating spline.

Result = SPL_INIT(X, Y [, /DOUBLE] [, YP0=value] [, /YPN_1=value])

SPL_INTERP - Performs cubic spline interpolation.

Result = SPL_INTERP(X, Y, Y2, X2 [, /DOUBLE])

Alphabetical List of IDL Routines

SHMVAR
Alphabetical List of IDL Routines

SPLINE - Performs cubic spline interpolation.
Result = SPLINE( X, Y, T [, Sigma] [, /DOUBLE] )

SPLINE_P - Performs parametric cubic spline interpolation.
SPLINE_P( X, Y, Xr, Yr [, /DOUBLE]
[ , INTERVAL=value] [, TANK0=[X0, Y0]] [, TANK1=[Xn-1, Yn-1]])

SPRSAB - Performs matrix multiplication on sparse matrices.
Result = SPRSAB(A, B [, /DOUBLE]
[ , THRESHOLD=value] )

SPRSAX - Multiplies sparse matrix by a vector.
Result = SPRSAX(A, X [, /DOUBLE] )

SPRSIN - Converts matrix to row-index sparse matrix.
Result = SPRSIN(A [, /COLUMN]
[ , THRESHOLD=value] ) or
Result = SPRSIN(Columns, Rows, Values, N
[ , /DOUBLE] [, THRESHOLD=value] )

SPRSTP - Constructs the transpose of a sparse matrix.
Result = SPRSTP(A)

SQRT - Returns the square root of X.
Result = SQRT(X [, Thread pool keywords] )

STANDARDIZE - Computes standardized variables.
Result = STANDARDIZE(A [, /DOUBLE] )

STDDEV - Computes the standard deviation of an n-element vector.
Result = STDDEV(X [, /DOUBLE] [, /NAN] )

STOP - Stops the execution of a running program or batch file.
STOP [, Exp1, ..., Exprn]

STRARR - Returns string array containing zero-length strings.
Result = STRARR(D1 [, ..., Dn])

STRCMP - Compares two strings.
Result = STRCMP( String1, String2 [, N]
[ , /FOLD_CASE] )

STRCOMPRESS - Removes whitespace from a string.
Result = STRCOMPRESS( String [, /REMOVE_ALL] )

STREXTIME - Generates the visualization graphics from a path.
STREXTIME( Verts, Conn, Normals, Outverts, Outconn
[ , ANISOTROPY=arrays] [, PROFILE=arrays] )

STREGEX - Performs regular expression matching.
Result = STREGEX( StringExpression, RegularExpression
[ , /BOOLEAN] [, /EXTRACT] [, LENGTH=variable
[ , /SUBEXPR]) [, /FOLD_CASE] )

STRETCH - Stretches color table for contrast enhancement.
STRETCH( [, Low, High [, Gamma]] [, /CHOP] )

STRING - Converts its arguments to string type.
Result = STRING( Expression1, ..., Expressionn
[ , AM_PM=string, string] [, /DAYs_of_Week=string_array{7 names}]
[ , FORMAT=value] [, MONTHS=string_array{12
names}] [, /PRINT] )

STROJOIN - Collapses a string scalar or array into merged strings.
Result = STROJOIN( String [, Delimiter] [, /SINGLE] )

STRLN - Returns the length of a string.
Result = STRLEN(Expression)

STRLWOCASE - Converts a string to lower case.
Result = STRLOWCASE(String)

STRMATCH - Compares search string against input string expression.
Result = STRMATCH( String, SearchString
[ , /FOLD_CASE] )

STREMSAGE - Returns the text of an error number.
Result = STREMSAGE( Err [, /BLOCK] [, /CODE] ,
/NAME) )

STRMID - Extracts a substring from a string.
Result = STRMID( Expression, First_Character [, Length
[ , /REVERSE_OFFSET])

STRPOS - Finds first occurrence of a substring within a string.
Result = STRPOS( Expression, SearchString [, Pos
[ , /REVERSE_OFFSET] [, /REVERSE_SEARCH] )

STRPUT - Inserts the contents of one string into another.
STRPUT, Destination, Source [, Position]

STRSLP - Splits its input string argument into separate substrings,
according to the specified pattern.
Result = STRSLP( String [, Pattern
[ , COUNT=variable] [, ESCAPE=string [, /REGEX
[ , /FOLD_CASE] [, /EXTRACT] [, LENGTH=variable
[ , /PRESEVE_NULL] )

STRTRIM - Removes leading and/or trailing blanks from string.
Result = STRTRIM( String [, Flag] )

STRUCT_ASSIGN - Performs "relaxed structure assignment" to
copy a structure.
STRUCT_ASSIGN, Source, Destination [, /NOZERO
[ , /VERBOSE]

STRUCT_HIDE - Prevents the IDL HELP procedure from displaying
information about structures or objects.
STRUCT_HIDE, Arg1 [, Arg2, ..., Argn]

STRUPCASE - Converts a string to upper case.
SURFACE  -  Plots an array as a wireframe mesh surface.
SURFACE, Z [, X, Y] [, AX=degrees] [, AZ=degrees]
[ , BOTTOM=index] [, /HORIZONTAL] [, /LEGO]
[ , /LOWER_ONLY] [, /UPPER_ONLY]
[ , MAX_VALUE=value] [, MIN_VALUE=value]
[ , /SAVE] [, SHADES=narray] [, SKIRT=value]
[ , /XLOG] [, /YLOG] [, ZAXIS=[1 | 2 | 3 | 4]] [, /ZLOG]

Graphics Keywords: Accepts all graphics keywords accepted by PLOT except for: PSYM, SYMSIZE.

SURFR  -  Sets up 3D transformations by duplicating rotation, translation, and scaling of SURFACE.
SURFR [, AX=degrees] [, AZ=degrees]

SVDC  -  Computes Singular Value Decomposition of an array.
SVDC, A [, W, U, V [, /COLUMN] [, /DOUBLE]]
[ , ITMAX=value]

SVDFIT  -  Multivariate least squares fit using SVD method.
Result = SVDFIT( X, Y [, , M] [, , Asvector]
[ , CHISQ=variable] [, , COVAR=variable] [, , /DOUBLE]
[ , FUNCTION_NAME=string] [, , /LEGENDRE]
[ , MEASURE_ERRORS=vector] [, , SIGMA=variable]
[ , SING_VALUES=variable] [, , SINGULAR=variable]
[ , STATUS=variable] [, , TOL=value]
[ , VARIANCE=variable] [, , YFIT=variable]

SVSOL  -  Solves set of linear equations using back-substitution.
Result = SVSOL( U, W, V [, /COLUMN] [, /DOUBLE])

SWAP_ENDIAN  -  Reverses the byte ordering of scalars, arrays or structures.
Result = SWAP_ENDIAN(Variable
[ , /SWAP_IF_BIG_ENDIAN]
[ , /SWAP_IF_LITTLE_ENDIAN])

SWAP_ENDIAN_INPLACE  -  Reverses the byte ordering of scalars, arrays or structures.
SWAP_ENDIAN_INPLACE, Variable
[ , /SWAP_IF_BIG_ENDIAN]
[ , /SWAP_IF_LITTLE_ENDIAN]

SWITCH  -  Selects one statement for execution from multiple choices, depending upon the value of an expression.
SWITCH expression OF
expression: statement
ELSE: statement
ENDSWITCH

SYSTIME  -  Returns the current time as either a string, as the number of seconds elapsed since 1 January 1970, or as a Julian date.
String = SYSTIME([0, Elapsed Seconds]) [, /UTC])
or
Seconds = SYSTIME([1] /SECONDS)
or
Julian = SYSTIME([/JULIAN] [, /UTC])

T

T_CVF  -  Computes the cutoff value in a Student’s t distribution.
Result = T_CVF(P, Df)

T_PDF  -  Computes Student’s t distribution.
Result = T_PDF(V, Df)

T3D  -  Performs various 3D transformations.
T3D [, Array [, /RESET]] [, , MATRIX=variable]
[ , OBLIQUE=vector] [, , PERSPECTIVE=sp{eye at
(0,0,p)}] [, , ROTATE=[x, y, z]] [, , SCALE=[x, y, z]]
[ , TRANSLATE=[x, y, z]] [, , /XEXCH] [, , /YEXCH]

TAG_NAMES  -  Returns the names of tags in a structure.
Result = TAG_NAMES( Expression
[ , /STRUCTURE_NAME])

TAN  -  Returns the tangent of X.
Result = TAN(X [, /Thread pool keywords])

TANH  -  Returns the hyperbolic tangent of X.
Result = TANH(X [, /Thread pool keywords])

TEK_COLOR  -  Loads color table based on Tektronix printer.
TEK_COLOR [, Start_Index, Colors]

TEMPORARY  -  Returns a temporary copy of a variable, and sets the original variable to “undefined”.
Result = TEMPORARY(Variable)

TETRA_CLIP  -  Clips a tetrahedral mesh to an arbitrary plane in space and returns a tetrahedral mesh of the remaining portion.
Result = TETRA_CLIP ( Plane, Vertsin, Connin, Vertsout, Connout [, , AUXDATA_IN=narray,
AUXDATA_OUT=variable]
[, , CUT_VERTS=variable])

TETRA_SURFACE  -  Extracts a polygonal mesh as the exterior surface of a tetrahedral mesh.
Result = TETRA_SURFACE ( Verts, Connin)

TETRA_VOLUME  -  Computes properties of tetrahedral mesh array.
Result = TETRA_VOLUME ( Verts, Conn
[ , AUXDATA=variable] [, , MOMENT=variable])

THIN  -  Returns the “skeleton” of a bi-level image.
Result = THIN( Image[, /NEIGHBOR_COUNT]
[, /PRUNE])

THREAED  -  Plots a 3D array as a pseudo 3D plot.
THREAED, A [, Sp] [, TITLE=string] [, XTITLE=string]
[ , YTITLE=string]

TIME_TEST2  -  Performs speed benchmarks for IDL.
TIME_TEST2 [, Filename]
TIMEGEN - Returns an array of double-precision floating-point values that represent times in Julian values.

Result = TIMEGEN([D1,...,D8 | FINAL=value | [DAYS=vector | HOURS=vector]]

[MINUTES=vector | MONTHS=vector | SECONDS=vector | [START=value]]

[STEP_SIZE=value | [UNITs=string]

[YEAR=value]

[PAIRED]

TM_TEST - Performs t-means test.

Result = TM_TEST( X, Y, [ , PAIRED | , UNEQUAL ]

TOTAL - Sums of the elements of an array.

Result = TOTAL([Array | , Dimension]

[ , CUMULATIVE | , DOUBLE | , INTEGER]

[ , NAN | , PRESERVE_TYPE | , Thread pool keywords]

TrackBall Object - See “TrackBall” on page 109.

TRIANGULATE - Interpolates gridded set of points with a smooth quintic surface.

Result = TRI_SURF( Z, X, Y | , EXTRAPOLATE]

[ , MISSING=value | , REGULAR | XGRID=[xstart, xspacing] | , XVALUES=[array]]

[ , YVALUES=[array]]

[ , YGRID=[ystart, yspacing] | , BOUNDS=[xmin, ymin, xmax, ymax]]

[ , NX=value | , NY=value]

TRIANGULATE - Constructs Delaunay triangulation of a planar set of points.

TRIANGULATE, X, Y, Triangles [, B]

[ , CONNECTIVITY=variable | , SPHERE=variable | , DEGREES =]

[ , EVALUATE=variable | , REPEATS=variable]

TRIGRID - Interpolates irregularly-gridded data to a regular grid.

Result = TRIGRID(X, Y, Z, Triangles | , GS, Limits)]

For spherical gridding:

Result = TRIGRID(F, GS, Limits, SPHERE=S)

Keywords: [ , DEGREES | , EXTRAPOLATE=variable | , QUINTIC | INPUT=variable]

[ , MAX_VALUE=value | , MIN_VALUE=value]

[ , MISSING=value | , NX=value | , NY=value]

[ , SPHERE=variable | , XGRID=variable]

[ , XGRID=variable | , XOUT=vector | YOUT=vector]

TRI - Determines eigenvalues and eigenvectors of tridiagonal array.

TRIQL, D, A [, , DOUBLE]

TRIRED - Reduces a real, symmetric array to tridiagonal form.

TRIRED, A, D, E [, , DOUBLE]

TRISOL - Solves tridiagonal systems of linear equations.

Result = TRISOL( A, B, C, R [, , DOUBLE]

TRUNCATE_LUN - Truncates an open file at the location of the current file pointer.

TRUNCATE_LUN, Unit1, ... , UnitN

TS_COEF - Computes the coefficients for autoregressive time-series.

Result = TS_COEF(X, P | , /DOUBLE | , MSE=variable)

TS_DIFF - Computes the forward differences of a time-series.

Result = TS_DIFF(X, K | , /DOUBLE)

TS_FCAST - Computes future or past values of a stationary time-series.

Result = TS_FCAST(X, P, Nvalues | , /BACKCAST | , /DOUBLE)

TS_SMOOTH - Computes moving averages of a time-series.

Result = TS_SMOOTH(X, Nvalues | , /BACKWARD | , /DOUBLE | , /FORWARD | , ORDER=value)

TV - Displays an image.

TV, Image [, Position]

or

TV, Image | , X, Y | , Channel[]

Keywords: [ , /CENTIMETERS | , /INCHES | , /ORDER | TRUE=[1 | 2 | 3]]

[ , /WORDS | , XSIZE=value]

[ , YSIZE=value]

Graphics Keywords: [ , CHANNEL=value | , /DATA | , /DEVICE | , /NORMAL | , /T3D | Z=value]

TVCRS - Manipulates the image display cursor.

TVCRS [, ON_OFF]

or

TVCRS [, X, Y]

Keywords: [ , /CENTIMETERS | , /INCHES | , /HIDE_CURSOR]

[ , /ORDER | , TRUE=[1 | 2 | 3]]

[ , /WORDS]

[ , XSIZE=value]

[ , YSIZE=value]

[ , /NORMAL | , /T3D | Z=value]

TVLCT - Loads display color tables.

TVLCT, V1, V2, V3 [, Start | , /GET | , /HLS | , /HSV]

or

TVLCT, V [, Start | , /GET | , /HLS | , /HSV]

TVRD - Reads an image from a window into a variable.

Result = TVRD | , X0, Y0, N, N, [ , Channel[]]

[ , CHANNEL=value | , /ORDER | TRUE=[1 | 2 | 3]]

[ , /WORDS]

TVSCL - Scales and displays an image.

TVSCL, Image [, Position]

or

TVSCL, Image | , X, Y | , Channel[]

Keywords: [ , /CENTIMETERS | , /INCHES | , /NAN | , /ORDER | TRUE=[1 | 2 | 3]]

[ , /WORDS | , XSIZE=value | , YSIZE=value]

[ , /NORMAL | , /T3D | Z=value | , Thread pool keywords]
V

VALUE_LOCATE - Finds the intervals within a given monotonic vector that brackets a given set of one or more search values.
Result = VALUE_LOCATE ( Vector, Value [, /L64] )

VARIANCE - Computes the statistical variance of an n-element vector.
Result = VARIANCE( X [, /DOUBLE] [, /NAN] )

VECTOR_FIELD - Places colored, oriented vectors of specified length at each vertex in an input vertex array.
VECTOR_FIELD, Field, Outverts, Outconn
[, ANISOTROPY=array] [, SCALE=value]
[, VERTICES=array]

VEL - Draws a velocity (flow) field with streamlines.
VEL, U, V [, NV ECS=value] [, XMAX=value[, xsize]] [, LENGTH=value{longest|steps}]
[, NSTEPS=value] [, TITLE=string]

VELOVECT - Draws a 2D velocity field plot.
VELOVECT, U, V [, X, Y] [, COLOR=index]
[, MISSING=value [, /DOTS]] [, LENGTH=index]
[, OVERPLOT] [Also accepts all PLOT keywords]

VERT_T3D - Transforms a 3D array by a 4x4 transformation matrix.
Result = VERT_T3D( Vertex_List [, DOUBLE=value]
[, MATRIX=4x4_array] [, /NO_COPY] [, /NO_DIVIDE
[, SAVE_DIVIDE=variable] )

VOIGHT - Calculates intensity of atomic absorption line (Voight) profile.
Result = VOIGHT(A, U [, Thread pool keywords])

VORONOI - Computes Voronoi polygon given Delaunay triangulation.
VORONOI X, Y, I0, C, Xp, Yp, Rect

VOXEL_PROJ - Creates volume visualizations using voxel technique.
Result = VOXEL_PROJ( V [, RGBO]
[, BACKGROUND=array] [, CUTTING_PLANE=arrays] [, /INTERPOLATE]
[, /MAXIMUM_INTENSITY] [, STEP=Sx, Sy, Sz]
[, XSIZE=pixels] [, YSIZE=pixels]
[, ZBUFFER=int_array] [, ZPIXELS=byte_array] )

W

WAIT - Suspends execution of an IDL program for a specified period.
WAIT, Seconds

WARP_TRI - Warps an image using control points.
Result = WARP_TRI( Xo, Yo, Xi, Yi, Image
[, /EXTRAPOLATE] [, OUTPUT_SIZE=vector]
[, /QUINTIC] [, /TPS] )

WATERSHED - Applies the morphological watershed operator to a grayscale image.
Result = WATERSHED ( Image
[, CONNECTIVITY={4 | 8}] [, /LONG]
[, NREGIONS=variable] )

WDELETE - Deletes IDL graphics windows.
WDELETE [ Window_Index [, ...] ]
**WF_DRAW** - Draws weather fronts with smoothing.

**WIDGET_BASE** - Creates base widget (containers for other widgets).

**WIDGET_ACTIVEX** - Create an ActiveX control and place it into an IDL widget hierarchy.

**WHERE** - Returns subscripts of nonzero array elements.

**WHILE** - Performs statement(s) as long as expression evaluates to true. Subject is never executed if condition is initially false.

**WIDGET_COMBOBOX** - Creates editable droplist widgets.

**WRAP** - Displays text, wrapping to fit a specified width.
WIDGET_COMBOBOX - continued
[ , SCR_MODE=value ] [ , /SENSITIVE ]
[ , SCR_SIZE=value ] [ , /TRACKING_EVENTS ]
[ , UNAME=value ] [ , UNITS={0 | 1 | 2 } ]
[ , UNVALUE=value ] [ , VALUE=value ]
[ , XOFFSET=value ] [ , XSIZE=value ]
[ , YOFFSET=value ] [ , YSIZE=value ]

WIDGET_CONTROL - Realizes, manages, and destroys widgets.
WIDGET_CONTROL [, Widget_ID]
All widgets: [ , BAD_ID=variable ] [ , /CLEAR_EVENTS ]
[ , DEFAULT_FONT=string ] [ {do not specify Widget_ID} ]
[ , /DELAY_DESTROY {do not specify Widget_ID} ]
[ , EVENT_FUNC=string ] [ , FUNC_GET_VALUE=string ]
[ , GET_UVALUE=variable ] [ , GROUP_LEADER=widget_id ]
[ , /HOURGLASS {do not specify Widget_ID} ]
[ , KILL_NOTIFY=string ] [ , /MAP ] [ , /NO_COPY ]
[ , NOTIFY_REALIZE=string ] [ , PRO_SET_VALUE=string ]
[ , /PUSHBUTTON_EVENTS ] [ , /REALIZE ]
[ , /RESET {do not specify Widget_ID} ]
[ , SCR_XSIZE=width ] [ , SCR_YSIZE=height ]
[ , SEND_EVENT=structure ] [ , /SENSITIVE ]
[ , SET_UNAME=string ] [ , SET_UVALUE=value ]
[ , /SHOW ] [ , TIMER=value ]
[ , TLB_GET_OFFSET=variable ] [ , /TLB_KILL_REQUEST_EVENTS ]
[ , TLB_GET_SIZE=variable ] [ , TLB_SET_TITLE=string ]
[ , TLB_SET_XOFFSET=value ] [ , TLB_SET_YOFFSET=value ]
[ , /TRACKING_EVENTS ] [ , /UPDATE ]
[ , X_OFFSET=value ] [ , XSIZE=value ]
[ , Y_OFFSET=value ] [ , YSIZE=value ]

WIDGET_BASE: [ , CANCEL_BUTTON=widget_id ] [ , /CONTEXT_EVENTS ]
[ , DEFAULT_BUTTON=widget_id ] [ , /ICONIFY ]
[ , /KBRD_FOCUS_EVENTS ] [ , /TLB_ICONIFY_EVENTS ]
[ , /TLB_KILL_REQUEST_EVENTS ] [ , /TLB_MOVE_EVENTS ] [ , /TLB_SIZE_EVENTS ]
[ , TOOLTIP=string ]

WIDGET_DRAW: [ , /DRAW_BUTTON_EVENTS ]
[ , /DRAW_EXPOSE_EVENTS ]
[ , DRAW_KEYBOARD_EVENTS={0 | 1 | 2 } ]
[ , /DRAW_MOTION_EVENTS ]
[ , DRAW_VIEWPORT_EVENTS ]
[ , DRAW_XSIZE=integer ] [ , DRAW_YSIZE=integer ]
[ , GET_DRAW_VIEW=variable ]
[ , /INPUT_FOCUS ] [ , SET_DRAW_VIEW=x,y ]
[ , TOOLTIP=string ]

WIDGET_DROPLIST: [ , /DYNAMIC_RESIZE ]
[ , GET_VALUE=value ] [ , SET_DROPLIST_SELECT=integer ]
[ , SET_VALUE=value ] [ , TAB_MODE=value ]

WIDGET_LABEL: [ , /DYNAMIC_RESIZE ]
[ , GET_VALUE=value ] [ , SET_VALUE=value ] [ , TAB_MODE=value ]

WIDGET_LIST: [ , /CONTEXT_EVENTS ]
[ , SET_LIST_SELECT=value ] [ , SET_LIST_TOP=integer ]
[ , SET_VALUE=value ] [ , TAB_MODE=value ]

WIDGET_PROPERTIESHEET: [ , /CONTEXT_EVENTS ] [ , /EDITABLE ]
[ , /MULTIPLE_PROPERTIES ]
[ , PROPERTIESHEET_SETSELECTED=empty string, string, or array of strings ]
[ , REFRESH_PROPERTY=string, string array, or integer ]

WIDGET_SLIDER: [ , GET_VALUE=value ]
[ , SET_SLIDER_MAX=value ] [ , SET_SLIDER_MIN=value ]
[ , SET_VALUE=value ] [ , TAB_MODE=value ]

WIDGET_TAB: [ , SET_TAB_CURRENT=index ]
[ , SET_TAB_MULTILINE=value ] [ , TAB_MODE=value ]
WIDGET_DISPLAYCONTEXTMENU - Displays a context-sensitive menu.

WIDGET_DISPLAYCONTEXTMENU, Parent, X, Y, ContextBase_ID

WIDGET_DRAW - Creates drawable widgets.

Result = WIDGET_DRAW( Parent, [ /APP_SCROLL ]
[ , /BUTTON_EVENTS ] [ , /COLOR_MODEL ]
[ , /COLORS(integer) ] [ , EVENT_FUNC=string ]
[ , EVENT_PRO=string ] [ , /EXPLOSE_EVENTS ]
[ , FRAME=width ] [ , /FUNCTION=VALUE=string ]
[ , GRAPHICS_LEVEL=2 ]
[ , GROUP_LEADER=widget_id ]
[ , IGNORE_ACCELERATORS=value ]

WIDGET_DROPLIST - Creates droplist widgets.

Result = WIDGET_DROPLIST( Parent, [ /ALL_EVENTS ]
[ , /CONTEXT_EVENTS ] [ , /EDITABLE ]
[ , /EVENT_FUNC=string ] [ , /NO_COPY ]
[ , /OPERATION=string ] [ , /TRACKING_EVENTS ]
[ , /UNAME=string ] [ , /USE_TABLE_SELECT ]
[ , /USE_TABLE_SELECT=integer ] [ , /USE_TEXT_SELECT ]

WIDGET_EVENT - Returns events for the widget hierarchy.

Result = WIDGET_EVENT( Widget_ID )

WIDGET_INFO - Obtains information about widgets.

Result = WIDGET_INFO( Widget_ID )

All widgets: [ /ACTIVE ] [ , /CHILD ] [ , /EVENT_FUNC ]
[ , /EVENT_PRO ] [ , FIND_BY_NAME=string ]
[ , /FRAME=width ] [ , /FUNCTION=VALUE=string ]
[ , /GRAPHICS_LEVEL=2 ]
[ , /GROUP_LEADER=integer ] [ , /NAME=string ]
[ , /PARENT ]
[ , /PUSHBUTTON_EVENTS ] [ , /REALIZED ]
[ , /SENSITIVE ] [ , /SIDEING ] [ , /SYSTEM_COLORS ]
[ , TAB_MODE=value ] [ , /TRACKING_EVENTS ]
[ , /TYPE ] [ , /UPDATE ] [ , /VALID_ID=string ]
[ , /VERSION ]

WIDGET_BASE - Displays a context-sensitive menu.

WIDGET_BASE, ContextBase_ID, Parent, [ /APP_SCROLL ]
[ , /BUTTON_EVENTS ] [ , /COLOR_MODEL ]
[ , /COLORS(integer) ] [ , EVENT_FUNC=string ]
[ , EVENT_PRO=string ] [ , /EXPLOSE_EVENTS ]
[ , FRAME=width ] [ , /FUNCTION=VALUE=string ]
[ , GRAPHICS_LEVEL=2 ]
[ , GROUP_LEADER=integer ] [ , /NAME=string ]
[ , /PARENT ]
[ , /PUSHBUTTON_EVENTS ] [ , /REALIZED ]
[ , /SENSITIVE ] [ , /SIDEING ] [ , /SYSTEM_COLORS ]
[ , TAB_MODE=value ] [ , /TRACKING_EVENTS ]
[ , /TYPE ] [ , /UPDATE ] [ , /VALID_ID=string ]
[ , /VERSION ]

UNIX Keywords: [ , /YIELD_TO_TTY ]

ID: IDL Quick Reference

Alphabetical List of IDL Routines

67
Alphabetical List of IDL Routines

WIDGET_INFO - continued

WIDGET_INFO:

WIDGET_BUTTON: [. /BUTTON_SET]
[. /DYNAMIC_RESIZE] [. /PUSHBUTTON_EVENTS]
[. /TOOLTIP]

WIDGET_COMBOBOX: [. /COMBOBOX_GETTEXT]
[. /COMBOBOX_NUMBER] [. /DYNAMIC_RESIZE]

WIDGET_DRAW: [. /DRAW_BUTTON_EVENTS]
[. /DRAW_EXPOSE_EVENTS]
[. /DRAW_KEYBOARD_EVENTS]
[. /DRAW_MOTION_EVENTS]
[. /DRAW_NOTIFY_EVENTS]
[. /DRAW_VIEWPORT_EVENTS] [. /TOOLTIP]

WIDGET_DROPDOWN: [. /DROPDOWN_NUMBER]
[. /DROPDOWN_SELECT] [. /DYNAMIC_RESIZE]

WIDGET_LABEL: [. /DYNAMIC_RESIZE]

WIDGET_LIST: [. /CONTEXT_EVENTS]
[. /LIST_MULTIPLE] [. /LIST_NUMBER]
[. /LIST_NUM_VISIBLE] [. /LIST_SELECT] [. /LIST_TOP]

WIDGET_PROPERTIESHEET:
[. /CONTEXT_EVENTS] [. COMPONENT=objref]
[. /MULTIPLE_PROPERTIES]
[. PROPERTY_VALUE=string]
[. PROPERTY_VALID=string]
[. /PROPERTIESHEET_SELECTED]
[. /PROPERTIESHEET_SELECTED]

WIDGET_SLIDER: [. /SLIDER_MIN_MAX]

WIDGET_TAB:
[. /TAB_CURRENT]
[. /TAB_MULTILINE] [. /TAB_NUMBER]

WIDGET_TABLE:
[. /CONTEXT_EVENTS]
[. /CONTEXT_EVENTS]
[. /ROW_HEIGHTS [not supported in Windows]]
[. /TABLE_ALL_EVENTS]
[. /TABLE_DISJOINT_SELECTION]
[. /TABLE_EDITABLE] [. /TABLE_EDIT_CELL]
[. /TABLE_SELECT] [. /TABLE_VIEW]
[. /USE_TABLE_SELECT]

WIDGET_TEXT:
[. /CONTEXT_EVENTS]
[. /TEXT_ALL_EVENTS] [. /TEXT_EDITABLE]
[. /TEXT_NUMBER]
[. /TEXT_OFFSET_TO_XY=integer]
[. /TEXT_SELECT] [. /TEXT_TOP_LINE]
[. /TEXT_XY_TO_OFFSET=[column, line]]

WIDGET_TREE: [. /CONTEXT_EVENTS]
[. /TREE_EXPANDED] [. /TREE_ROOT]
[. /TREE_SELECT]

WIDGET_PROPERTIESHEET:

WIDGET_DRAW:

WIDGET_LABEL:

WIDGET_LIST:

WIDGET_PROPERTIESHEET:

---

WIDGET_INFO:

IDL Quick Reference
Alphabetical List of IDL Routines

WIDGET_SLIDER - Creates slider widgets.
\[ Result = \text{WIDGET\_SLIDER( Parent [, /DRAG]} \]
\[ /EVENT\_FUNC=string [, EVENT\_PRO=string] \]
\[ /FONT=string [, FRAME=width] \]
\[ /FUNC\_GET\_VALUE=string \]
\[ /GROUP\_LEADER=widget\_id \]
\[ /KILL\_NOTIFY=string [, MAXIMUM=value] \]
\[ /MINIMUM=value [, /NO\_COPY] \]
\[ /NOTIFY\_REALIZE=string \]
\[ /PRO\_SET\_VALUE=string \]
\[ /RESIZABLE\_COLUMNS \]
\[ /RESIZABLE\_ROWS{not supported in Windows}] \]
\[ /RESOURCE\_NAME=string \]
\[ /ROW\_HEIGHTS=string \]
\[ /SCR\_XSIZES=string \]
\[ /SCR\_YSIZE=height [, SCROLL] [, SENSITIVE] \]
\[ /SCR\_YSIZE=height [, /DRAG] \]
\[ /SCROLL=value [, XOFFSET=value] [, YOFFSET=value] \]
\[ /TAB\_MODE=value [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string \]
\[ /RESIZABLE\_ROWS{not supported in Windows}] \]
\[ /RESIZABLE\_COLUMNS \]
\[ /RESOURCE\_NAME=string \]
\[ /ROW\_HEIGHTS=string \]
\[ /SCR\_XSIZES=string \]
\[ /SCR\_YSIZE=height [, SCROLL] [, SENSITIVE] \]
\[ /SCR\_YSIZE=height [, /DRAG] \]
\[ /SCROLL=value [, XOFFSET=value] [, YOFFSET=value] \]

WIDGET_TAB - Creates tab widgets.
\[ Result = \text{WIDGET\_TAB( Parent [, /ALIGN\_BOTTOM]} \]
\[ /ALIGN\_CENTER] [, /ALIGN\_LEFT] [, /ALIGN\_TOP] \]
\[ /EVENT\_FUNC=string [, EVENT\_PRO=string] \]
\[ /FONT=string [, FRAME=width] \]
\[ /FUNC\_GET\_VALUE=string \]
\[ /GROUP\_LEADER=widget\_id \]
\[ /KILL\_NOTIFY=string [, LOCATION=0 \[1 \[2 \[3] \]
\[ /MULTILINE=value \]
\[ /NO\_COPY] [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, SENSITIVE] \]
\[ /TAB\_MODE=value [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]

WIDGET_TABLE - Creates table widgets.
\[ Result = \text{WIDGET\_TABLE( Parent [, /ALIGN\_BOTTOM]} \]
\[ /ALIGN\_CENTER] [, /ALIGN\_LEFT] [, /ALIGN\_TOP] \]
\[ /COLUMN\_LABELS=string\_array \]
\[ /COLUMN\_MAJOR] [, /ROW\_MAJOR] \]
\[ /COLUMN\_WIDTHS=string\_array \]
\[ /CONTEXT\_EVENTS \]
\[ /DAYS\_OF\_WEEK=string\_array{7 names} \]
\[ /DISJOINT\_SELECTION] [, /EDITABLE] \]
\[ /EVENT\_FUNC=string [, EVENT\_PRO=string] \]
\[ /FONT=string [, FORMAT=value] [, FRAME=width] \]
\[ /FUNC\_GET\_VALUE=string \]
\[ /GROUP\_LEADER=widget\_id \]
\[ /IGNORE\_ACCELERATORS=value \]
\[ /KBRD\_FOCUS\_EVENTS] [, KILL\_NOTIFY=string] \]
\[ /MONTHS=string\_array{12 names}] \]
\[ /NO\_COLUMN\_HEADERS] [, /NO\_COPY] \]
\[ /NO\_ROW\_HEADERS] [, /NO\_NEWLINE] \]
\[ /NOTIFY\_REALIZE=string \]
\[ /PRO\_SET\_VALUE=string \]
\[ /RESIZABLE\_COLUMNS \]
\[ /RESIZABLE\_ROWS{not supported in Windows}] \]
\[ /RESOURCE\_NAME=string \]
\[ /ROW\_HEIGHTS=string \]
\[ /SCR\_XSIZES=string \]
\[ /SCR\_YSIZE=height [, SCROLL] [, SENSITIVE] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /TAB\_MODE=value [, /KBRD\_FOCUS\_EVENTS] \]
\[ /UNAMES=string [, UNITS=0 \[1 \[2] \]
\[ /XOFFSET=value] [, YOFFSET=value] [, /NO\_COPY] [, /NEWLINE] \]
\[ /YOFFSET=value] [, YOFFSET=value] [, /NO\_COPY] [, /NEWLINE] \]
\[ /YOFFSET=value] [, YOFFSET=value] [, /NO\_COPY] [, /NEWLINE] \]

WIDGET_TREE - Creates tree widgets.
\[ Result = \text{WIDGET\_TREE( Parent [, /ALL\_EVENTS]} \]
\[ /CONTEXT\_EVENTS] [, /EDITABLE] \]
\[ /EVENT\_FUNC=string [, EVENT\_PRO=string] \]
\[ /FONT=string [, FRAME=width] \]
\[ /FUNC\_GET\_VALUE=string \]
\[ /GROUP\_LEADER=widget\_id \]
\[ /IGNORE\_ACCELERATORS=value \]
\[ /KBRD\_FOCUS\_EVENTS] [, KILL\_NOTIFY=string] \]
\[ /NO\_COPY] [, /KILL\_NOTIFY=string [, NOTIFY\_REALIZE=string] \]
\[ /PRO\_SET\_VALUE=string [, SCR\_XSIZE=width] \]
\[ /SCR\_YSIZE=height [, SCROLL] [, SENSITIVE] \]
\[ /SCR\_YSIZE=height [, /KBRD\_FOCUS\_EVENTS] \]
\[ /TAB\_MODE=value [, /KBRD\_FOCUS\_EVENTS] \]
\[ /UNAMES=string [, UNITS=0 \[1 \[2] \]
\[ /XOFFSET=value] [, YOFFSET=value] [, /NO\_COPY] [, /NEWLINE] \]
\[ /XOFFSET=value] [, YOFFSET=value] [, /NO\_COPY] [, /NEWLINE] \]
\[ /XOFFSET=value] [, YOFFSET=value] [, /NO\_COPY] [, /NEWLINE] \]

IDL Quick Reference

WIDGET_TREE
WINDOW - Create a window for the display of graphics or text.
WINDOW [ , Window_Index ] [ , COLORS=value ] [ , /FREE ] [ , /PIXMAP ] [ , RETAIN=[0 | 1 | 2] ] [ , TITLE=string ] [ , XPOS=value ] [ , YPOS=value ] [ , XSIZE=pixels ] [ , YSIZE=pixels ]

WRITE_BMP - Write a Microsoft Windows Version 3 device independent bitmap file. (BMP)

WRITE_IMAGE - Write an image and its color table vectors, if any, to a file of a specified type.
WRITE_IMAGE, Filename, Format, Data [, Red, Green, Blue ] [, /APPEND]

WRITE_JPEG - Write a JPEG file.
WRITE_JPEG, Filename [, UNIT=1] [ , Image ] [ , ORDER= ] [ , /PROGRESSIVE] [ , QUALITY=value[10 to 100] ] [ , TRUE=[1 | 2 | 3] ]

WRITE_JPEG2000 - Write a JPEG2000 file.
WRITE_JPEG2000, Filename, Image [, Red, Green, Blue ] [, N_LAYERS=value ] [ , N_LEVELS=value ] [, /ORDER= ] [ , /REVERSIBLE ]

WRITE_NRIF - Write NCAR Raster Interchange Format rasterfile.
WRITE_NRIF, Image [, /ASCII]

WRITE_PICT - Write a Macintosh PICT (version 2) bitmap file.
WRITE_PICT, Filename [, Image, R, G, B ]

WRITE_PNG, Filename, Image [, R, G, B ] [, /VERBOSE ] [ , TRANSPARENT= ] [ , /ORDER= ]

WRITE_PPM - Write PPM (true-color) or PGM (gray scale) file.
WRITE_PPM, Filename, Image [, /ASCII]

WRITE_SPR - Write a row-indexed sparse array structure to a file.
WRITE_SPR, AS, Filename

WRITE_SRF - Write a Sun Raster File (SRF).

WRITE_SYLK - Write SYLK (Symbolic Link) spreadsheet file.
WRITE_SYLK, Filename, Data [, STARTCOL=column ] [, STARTROW=row ]

WRITE_TIFF - Write TIFF file with 1 to 3 channels.
WRITE_TIFF, Filename, Image [, /APPEND ] [, /FREE ] [ , BITS_PER_SAMPLE=[0 | 1 | 8] ] [ , RED=value ] [ , GREEN=value ] [ , BLUE=value ] [, /CMYK] [ , COMPRESSION=[0 | 1 | 2 ] [ , /ASCII ] [ , DESCRIPTION=string ] [ , DOCUMENT_NAME=string ] [ , DOT_RANGE=intarray ] [ , GEOTIFF=structure ] [ , ICC_PROFILE= ] [ , /LONG ] [ , /SHORT ] [ , /FLOAT ] [ , ORIENTATION=value ] [ , /JPEG ] [ , /PHOTOSHOP=bytearray ] [ , /PLANARCONFIG=[0 | 1 | 2 ] ] [ , /VERBOSE ] [ , /XPOSITION=units ] [ , XRESOL=pixels/inch ] [ , /YPOSITION=units ] [ , YRESOL=pixels/inch ]

Note: LZW compression (COMPRESSION=1) is only available with the appropriate license feature.

WRITE_WAV - Write audio stream to a named .WAV file.
WRITE_WAV, Filename, Data [, Rate ]

WRITE_WAVE - Write Wavefront Advanced Visualizer (.WAV) file.
WRITE_WAVE, File, Array [, /BIN ] [, DATANAME=string ] [ , MESHNAME=string ] [, /NOMESHDEF ] [, /VECTOR]

WRITEU - Write unformatted binary data to a file.
WRITEU, Unit, Expr1, ..., Exprn [, TRANSFER_COUNT=variable ]

WSET - Selects the current window.
WSET [, Window_Index ]

WSHOW - Exposes or hides the designated window.
WSHOW [, Window_Index ] [, Show ] [, /ICONIC ]

WTN - Return wavelet transform of the input array.

X

XBM_EDIT - Create, edit, and bitmap icons for IDL widget button labels.

XDISPLAYFILE - Display ASCII text file in scrolling text widget.

XDXF - Utility for displaying and interactively manipulating DXF objects.
XDXF, Filename [, /BLOCK ] [, GROUP=widget_id ] [, SCALE=value ] [, /TEST ] [ /keywords to XOBJVIEW ]

XFONT - Create modal widget to select and view an X Windows font.
XFONT, Result=XFONT( [, GROUP=widget_id ] [, /PRESERVE_FONT_INFO ] )
**Alphabetical List of IDL Routines**

- **XINTERANIMATE** - Displays animated sequence of images.
  **Keywords for initialization:** [.SET=[sizex, sizey, nframes]], [.BLOCK], [.CYCLE]
  [.GROUP=widget_id], [.MODAL], [.MPEG_BITRATE=value]
  [.MPEG_FRAMES=value]
  [.MPEG_MOTION_VEC_LENGTH=[1 2 3]]
  [.MPEG_OPEN, MPEG_FILENAME=string]
  [.MPEG_QUALITY=value(0 to 100)], [.SHOWLOAD]
  [.TRACK], [.TITLE=string]

- **Keywords for loading images:** [.FRAME=value(0 to (nframes-1)]], [.IMAGE=value] [.ORDER]
  [.WINDOW=window_num [x0, y0, sx, sy]]

- **Keywords for running animations:** [.CLOSE]
  [.KEEP_PIXMAPS], [.MPEG_CLOSE]
  [.XOFFSET=], [.YOFFSET=]
  [.WINDOW=window_num [x0, y0, sx, sy]]

- **XMANAGER** - Provides event loop manager for IDL widgets.
  **Keywords:** [.NAME=string], [.GROUP=string]
  [.CLEANUP=string]
  [.EVENT_HANDLER=procedure]
  [.GROUP_LEADER=widget_id], [.UPDATETECDATA=value]
  [.USE_CURENTP]

- **XMNG_TMPL** - Template for creating widgets.
  **Keywords:** [.GROUP=string]

- **XOBJECT** - Displays object viewer widget.
  **Keywords:** [.OBJ=Obj], [.BACKGROUND=[r, g, b]]
  [.BLOCK], [.DOUBLE_VIEW], [.GROUP=widget_id]
  [.JUST_REG], [.MODAL], [.REFRESH=widget_id]
  [.RENDBER=[0 1]], [.SCALE=value]
  [.STATIONARY=[objref(s)]], [.TEST], [.TITLE=string]
  [.TLB=value], [.XOFFSET=value], [.XSIZE=pxels]
  [.YOFFSET=value], [.YSIZE=pxels]

- **XOBJECT_ROTATE** - Programmatically rotate the object currently displayed in XOBJECTVIEW.
  **Keywords:** [.AXIS], [.ANGLE], [.PREMULTIPLY]

- **XOBJECT_WRITE_IMAGE** - Write the object currently displayed in XOBJECTVIEW to an image file.
  [.DIMENSIONS=[x, y]]

- **XPALLETTE** - Displays widget used to create and modify color tables.
  **Keywords:** [.BLOCK], [.GROUP=widget_id]
  [.UPDATECALLBACK=procedure_name]
  [.UPDATECBDATA=value]

- **XPCOLOR** - Adjusts the value of the current foreground plotting color.
  **Keywords:** [.GROUP=string], [.COLOR=string]

- **XPLOT3D** - Utility for creating and interactively manipulating 3D plots.
  **Keywords:** [.DOUBLE_VIEW], [.GROUP=string]
  [.LINESTYLE=[0 1 2 3 4 5 6]], [.MODAL]
  [.NAME=string], [.OVERPLOT]
  [.SYMBOL=[objref(s)]], [.TEST]
  [.THICK=points[1.0 to 10.0]], [.TITLE=string]
  [.XSCALE=value], [.YSCALE=value], [.ZSCALE=value]
  [.XTITLE=string], [.YTITLE=string], [.ZTITLE=string]

- **XRREGISTERED** - Returns registration status of a given widget.
  **Result:** XREGISTERED(Name [, NOSHOW])}

- **XROI** - Utility for interacting with ROIs.
  **Keywords:** [.IMAGE=Data [, R] [, G] [, B]], [.BLOCK]
  [.DOUBLE_VIEW], [.GROUP=string]
  [.GROUP=string], [.MODAL]
  [.REGIONS_IN=value], [.REGIONS_OUT=value]
  [.REJECTED=variable], [.RENDBER=0 1]
  [.ROI_COLOR=[r, g, b] or variable]
  [.ROI_GEOMETRY=variable]
  [.ROI_SELECT_COLOR=[r, g, b] or variable]
  [.ROI_SELECT_COLOR=variable]
  [.STATISTICS=variable], [.TITLE=string]
  [.TOOLS=string or string array]
  [.X_SCROLL_SIZE=value], [.Y_SCROLL_SIZE=value]

- **XRESULT** - Computes Chi-square goodness-of-fit test.
  **Result:** XRESULT(Objref, Expref)
  [.EXCELL=variable], [.OCCBELL=variable]
  [.RESIDUAL=variable]

- **XSURFACE** - Provides GUI to SURFACE and SHADE_SURF.
  **Keywords:** [.DATA=Data [, BLOCK]], [.GROUP=widget_id]

- **XVAREDIT** - Provides widget-based editor for IDL variables.
  **Keywords:** [.VAR=variable], [.GROUP=string]
  [.X_SCROLL_SIZE=value], [.Y_SCROLL_SIZE=value]

- **XVOLUME** - Utility for viewing and interactively manipulating volumes and isosurfaces.
  **Keywords:** [.VOL=Vol [, BLOCK]], [.GROUP=widget_id]
  [.INTERPOLATE=variable], [.MODAL], [.RENDERER=[0 1]]
  [.REPLACE], [.SCALE=value], [.TEST]
  [.XSIZE=pxels], [.YSIZE=pxels]
XYOUTS - Draws text on currently-selected graphics device.

```
XYOUTS, [X, Y] String [, ALIGNMENT=value{0.0 to 1.0}] [, CHARSIZE=value] [, CHARTHICK=value]
[, TEXT_AXES={0 | 1 | 2 | 3 | 4 | 5}]
[, WIDTH=variable]
```

**Graphics Keywords:**

```
[, CLIP={X0, Y0, X1, Y1}]
[, COLOR=value] [, /DATA | /DEVICE | /NORMAL]
[, FONT=integer]
[, ORIENTATION=ccw_degrees_from_horiz]
[, /NOCLIP] [, /T3D] [, Z=value]
```

---

**Z**

**ZOOM** - Zooms portions of the display.

```
ZOOM [, /CONTINUOUS] [, FACT=integer] [, /INTERP]
[, /KEEP] [, /NEW_WINDOW] [, XSIZE=value]
[, YSIZE=value] [, /NOCLIP] [, ZOOM_WINDOW=variable]
```

**ZOOM_24** - Zooms portions of true-color (24-bit) display.

```
ZOOM_24 [, FACT=integer] [, /RIGHT] [, XSIZE=value]
[, YSIZE=value]
```
Objects

This section lists all IDL objects and their methods. In addition to the syntax conventions discussed in "IDL Syntax Conventions" on page 26, note the following:

- The Object_Name::Init method for each object has keywords that are followed by either {Get}, {Set}, or {Get, Set}. Properties retrievable via Object_Name::GetProperty are indicated by {Get}; properties settable via Object_Name::SetProperty are indicated by {Set}. Properties that are both retrievable and settable are indicated by {Get, Set}. Do not include the braces, Get, or Set in your call.

- Each object’s Cleanup method lists two possible syntaxes. The second syntax (Obj->Object_Name::Cleanup) can be used only in a subclass’ Cleanup method.

- Some objects have Init methods that list two possible syntaxes. The second syntax (Obj->Object_Name::Init) can be used only in a subclass’ Init method.

IDL_Container - Object used to hold other objects. No superclasses. Subclasses: IDLgrModel IDLgrScene IDLgrView IDLgrViewgroup.

IDL_Container::Add - Adds a child object to the container.

IDL_Container::Cleanup - Performs all cleanup on the object.

IDL_Container::Count - Returns the number of objects contained by the container object.

IDL_Container::Get - Returns an array of object references to objects in a container.

IDL_Container::Init - Initializes the container object.

IDL_Container::IsContained - Returns true (1) if the specified object is in the container, or false (0) otherwise.

IDL_Container::Move - Moves an object from one position in a container to a new position.

IDL_Container::Remove - Removes an object from the container.

IDL_Savefile - Object that provides complete query and restore capabilities for IDL SAVE files.

IDL_Savefile::Cleanup - Performs all cleanup on the object.

IDL_Savefile::Contents - returns a structure variable of type IDL_SAVEFILE_CONTENTS containing information about the associated SAVE file and its contents.

IDL_Savefile::Init - Initializes the Savefile object.

IDL_Savefile::Names - Returns names, or heap variable identifiers, of items contained within the SAVE file.

IDL_Savefile::Restore - Selectively restores individual items from the associated SAVE file.

IDL_Savefile::Size - Returns the size and type information for the specified variable, system variable, or heap variable in the SAVE file.
IDLanROI - Represents a region of interest. Superclass of IDLgrROI.

Properties:
- ALL{Get}=[variable] [, BLOCKSIZE{Get, Set}=dataarray] [ , DATA{Get, Init, Set}=dataarray] [ , DOUBLE{Get, Init, Set}=value] [, /INTERIOR{Get, Init, Set}=true|false] [ , N_VERTS{Get}=variable] [ , ROI_XRANGE{Get}=variable] [ , ROI_YRANGE{Get}=variable] [ , ROI_ZRANGE{Get}=variable] [, TYPE{Get, Init, Set}=value] [ , VERTICES{Get}=[variable]]

IDLanROI::AppendData - Appends vertices to the region.

IDLanROI::Init - Initializes a region of interest.

IDLanROI::ComputeGeometry - Computes the geometrical values for area, perimeter, and centroid of the region.

IDLanROI::Cleanup - Performs all cleanup for the object.

IDLanROI::ComputeMask - Prepares a two-dimensional mask for the region.

IDLanROI::ContainsPoints - Determines whether the given data coordinates are contained within the closed polygon region.

IDLanROI::GetProperty - Retrieves the value of a property or group of properties for the region.

IDLanROI::Init - Initializes a region of interest.

IDLanROI::RemoveData - Removes vertices from the region.

IDLanROI::ReplaceData - Replaces vertices in the region with alternate values.

IDLanROI::Rotate - Modifies the vertices for the region by applying a rotation.

IDLanROI::Scale - Modifies the vertices for the region by applying a scale.

IDLanROI::SetProperty - Sets the value of a property or group of properties for the region.

IDLanROI::Translate - Modifies the vertices for the region by applying a translation.

IDLanROIGroup - This object is an analytical representation of a group of regions of interest. Subclass of IDL_Container. Superclass of IDLgrROIGroup.

Properties:
- ALL{Get}=[variable] [, ROI_XRANGE{Get}=variable] [ , ROI_YRANGE{Get}=variable] [ , ROI_ZRANGE{Get}=variable] [ , ROIGROUP_XRANGE{Get}=variable] [ , ROIGROUP_YRANGE{Get}=variable] [ , ROIGROUP_ZRANGE{Get}=variable]

IDLanROIGroup::Add - Adds a region to the region group.

IDLanROIGroup::Cleanup - Performs all cleanup for the object.

IDLanROIGroup::ContainsPoints - Determines whether the given points (in data coordinates) are contained within the closed polygon regions within this group.

IDLanROIGroup::ComputeMesh - Triangulates a surface mesh with optional capping from the stack of regions contained within this group.

IDLanROIGroup::ComputeMask - Prepares a 2-D mask for this group of regions.

IDLanROIGroup::RemoveData - Removes vertices from the region.

IDLanROIGroup::Translate - Modifies the vertices for the region by applying a translation.

IDLanROIGroup::SetProperty - Sets the value of a property or group of properties for the region.
Alphabetical List of IDL Routines

IDLanROIGroup::GetProperty - Retrieves the value of a property or group of properties for the region group.

IDLffDICOM::DumpElements - Dumps a description of the DICOM data elements of IDLffDICOM object to the screen or to a file.

IDLanROIGroup::Init - Initializes a region of interest group object.

IDLcomIDispatch::Init - Initializes a COM object and establishes a link between the resulting IDL object and the IDispatch interface.

IDLffDICOM::Rotate - Rotates the vertices for the region by applying a rotation.

IDLanROIGroup::Scale - Scales the vertices for the region within the group by applying a scale.

IDLanROIGroup::Translate - Translates the vertices for the region within the group by applying a translation.

IDLcomActiveX - Creates an IDL object that encapsulates an ActiveX control.

IDLcomDispatch - Creates a COM object that implements an IDispatch interface. A dynamic sub-class of IDLcomDispatch is created when the object is instantiated.

IDLcomDispatch::GetProperty - Get properties for an IDispatch interface.

IDLcomDispatch::SetProperty - Set properties for an IDispatch interface.

IDLffDICOM - Contains the data for one or more images embedded in a DICOM part 10 file. No superclasses. No subclasses.

IDLffDICOM::Cleanup - Destroys the IDLffDICOM object.

IDLffDICOM::DumpElements - Dumps a description of the DICOM data elements of IDLffDICOM object to the screen or to a file.

IDLffDICOM::GetChildren - Finds the member element references of a DICOM sequence.

IDLffDICOM::GetDescription - Takes optional DICOM group and element arguments and returns array of STRING descriptions.

IDLffDICOM::GetProperty - Takes optional DICOM group and/or element arguments and returns an array of DICOM Element numbers for those parameters.

IDLffDICOM::GetElement - Takes optional DICOM group and/or element arguments and returns an array of DICOM Element numbers for those parameters.

IDLffDICOM::GetGroup - Takes optional DICOM group and/or element arguments and returns an array of DICOM Group numbers for those parameters.

IDLffDICOM::GetLength - Takes optional DICOM group and/or element arguments and returns an array of LONGs.

IDLffDICOM::GetParent - Finds the parent references of a set of elements in a DICOM sequence.

IDLffDICOM::GetParent( ReferenceList ) - Returns a list of references to the matching elements in the object.

IDLffDICOM::GetReference - Takes optional DICOM group and/or element arguments and returns an array of references to matching elements in the object.

IDLffDICOM::GetValue - Takes optional DICOM group and/or element arguments and returns an array of POINTERs to the values of the elements matching those parameters.

IDLffDICOM::GetVR - Takes optional DICOM group and/or element arguments and returns an array of VR Strings for those parameters.

IDLffDICOM::Init - Creates a new IDLffDICOM object and optionally reads the specified file as defined in the IDLffDICOM::Read method.

IDLffDICOM::Read - Opens and reads from the specified disk file, places the information into the DICOM object, then closes the file.

IDLffDICOM::Reset - Removes all of the elements from the IDLffDICOM object, leaving the object otherwise intact.

IDLffDicomEx - See the Medical Imaging in IDL manual.

IDL Quick Reference
IDLffDXF - Object that contains geometry, connectivity, and attributes for graphics primitives. No superclasses. No subclasses.

IDLffDXF::Cleanup - Performs all cleanup on the object.

IDLffDXF::GetContents - Returns the DXF entity types contained in the object.

IDLffDXF::GetEntity - Returns an array of vertex data for the requested entity type.

IDLffDXF::GetPalette - Returns current color table in the object.

IDLffDXF::Init - Initializes the DXF object.

IDLffDXF::PutEntity - Inserts an entity into the DXF object.

IDLffDXF::Read - Reads a file, parsing the DXF object information contained in the file, and inserts it into itself.

IDLffDXF::RemoveEntity - Removes the specified entity or entities from the DXF object.

IDLffDXF::Reset - Removes all the entities from the DXF object.

IDLffDXF::SetPalette - Sets the current color table in the object.

IDLffDXF::Write - Writes a file for the DXF entity information this object contains.

IDLffJPEG2000 - Object class used for reading and writing JPEG2000 files. No superclasses. No subclasses.

IDLffJPEG2000::Init - Initializes an IDLffJPEG2000 object.

IDLffJPEG2000::SetData - Writes data to the IDLffJPEG2000 object.

IDLffJPEG2000::TileProperty - Retrieves the properties of a tile in an IDLffJPEG2000 object.

IDLffJPEG2000::Write - Writes a file for the DXF entity information this object contains.


IDLffJPEG2000::GetProperty - Retrieves the value of a property or group of properties for the IDLffJPEG2000 object.

Obj->IDLffJPEG2000::GetProperty [ , PROPERTY=variable ]

IDQffJPEG2000::TileProperty - Retrieves the properties of a tile in an IDLffJPEG2000 object.

Obj->[IDLffJPEG2000::TileProperty [ , TileIndex [ , TILE_COMPONENT] [ , N_LAYERS=value ] [ , PROGRESSION=value ] [ , REVERSIBLE=value ] [ , TILE_DIMENSIONS=value ] [ , TILE_OFFSET=value ] [ , YCC=value ]

IDQffJPEG2000::Init - Initializes an IDLffJPEG2000 object.

Obj = OBJ_NEW('IDLffJPEG2000' [ , PROPERTY=variable ]

IDQffJPEG2000::SetData - Writes data to the IDLffJPEG2000 object.

Obj->[IDLffJPEG2000::SetData ([P1, ..., Pn] [ , COMPONENT=value ] [ , /ORDER ] [ , TILE_INDEX=value ]

Object contains.

Result = Obj->IDLffDXF::PutEntity [ , BLOCK/string ] [ , COUNT=variable ] [ , INDEX=value ] [ , LAYER=string ]

Result = Obj->IDLffDXF::GetContents [ , Filter ] [ , BLOCK=string ] [ , COUNT=variable ] [ , LAYER=string ]

Result = Obj->IDLffDXF::GetPalette [ , BLOCK=string ] [ , COUNT=variable ] [ , LAYER=string ]

Result = Obj->IDLffDXF::GetEntity [ , Type [ , BLOCK=string ] [ , INDEX=value ] [ , LAYER=string ]

Result = Obj->IDLffDXF::GetPalette [ , Red=[ , Green=[ , Blue=[ ] , BLOCK=string ] [ , COUNT=variable ] [ , LAYER=string ]

Result = Obj->IDLffDXF::Init [ , FileName ]

Result = Obj->IDLffDXF::Read [ , Filename ]

Result = Obj->IDLffDXF::RemoveEntity [ , Type [ , INDEX=value ]

Result = Obj->IDLffDXF::Reset [ , Filename ]

Result = Obj->IDLffDXF::SetPalette [ , Red=[ , Green=[ , Blue=[ ] , BLOCK=string ] [ , COUNT=variable ] [ , LAYER=string ]
IDLffLangCat::Init - Initializes an IDLffLangCat object.
Result = OBJ_NEW('IDLffLangCat', Language
[, PROPERTY=value] [, /CONTINUE_ON_ERROR]

IDLffLangCat::Query - Returns the string or string array that corresponds to all supplied key values.
Result = Obj->[IDLffLangCat::]Query(Key
[, DEFAULT_STRING='string'] )

IDLffLangCat::SetProperty - Sets the value of a property or group of properties for an IDLffLangCat object.
Obj->[IDLffLangCat::]SetProperty
[ PROPERTY=variable ]
IDLffShape::Init - Initializes or constructs a Shapefile object.
Result = Obj->[IDLffShape::Init( [ /ATTRIBUTE_NAMES=variable ] [ /ATTRIBUTE_INFO=variable ] [ /ENTITY_TYPE=variable ] [ /IS_OPEN=variable ] [ /FILENAME=variable ] [ /N_ENTITIES=variable ] [ /N_ATTRIBUTES=variable ] )]

IDLffShape::GetEntity - Returns an array of entity structures from a Shapefile.
Result = Obj->[IDLffShape::GetEntity( [ Index ] [ , /ALL ] [ , /ATTRIBUTES ] )]

IDLffShape::GetProperty - Returns the values of properties associated with a Shapefile object.
Obj->[IDLffShape::GetProperty [ /ATTRIBUTE_INFO=variable ] [ /ATTRIBUTES=variable ] [ /ENTITY_TYPE=variable ] [ /IS_OPEN=variable ] [ /FILENAME=variable ] [ /N_ATRIBUTES=variable ] [ /N_ENTITIES=variable ]]

IDLffShape::Open - Opens a specified Shapefile.
Result = Obj->[IDLffShape::Open( 'Filename' [ , /DBF_ONLY ] [ , /UPDATE ] [ , /ENTITY_TYPE=variable ] )]

IDLffShape::PutEntity - Inserts an entity into the Shapefile object.
Obj->[IDLffShape::PutEntity( Entity, Data )]

IDLffShape::SetAttributes - Modifies the attributes for a specified entity in a Shapefile object.
Obj->[IDLffShape::SetAttributes( Index, Attribute_Name, Value )]

IDLffXMLDOM Classes - Represents classes that provide support for IDL’s XML Document Object Model (DOM).
IDLffXMLDOMAttr - Represents an attribute that is a part of an element object in an XML document. Subclass of IDLffXMLDOMNode.
IDLffXMLDOMAttr::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.
Result = OBJ_Destroy, Obj

IDLffXMLDOMCharacterData::AppendData - Extends the IDLffXMLDOM character class with methods for accessing character data in the DOM tree. Subclass of IDLffXMLDOMNode.
Obj->[IDLffXMLDOMCharacterData::AppendData( String )]

IDLffXMLDOMCharacterData::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by the object; does not modify the actual DOM tree. Should not be subclassed.
Result = OBJ_Destroy, Obj

IDLffXMLDOMCharacterData::DeleteData - Deletes a number of characters from the node’s character data, starting at an offset.
Obj->[IDLffXMLDOMCharacterData::DeleteData( Offset, Count )]

IDLffXMLDOMCharacterData::GetData - Returns the node’s character data.
Result = Obj->[IDLffXMLDOMCharacterData::GetData()]

IDLffXMLDOMCharacterData::GetLength - Returns the number of characters in the node.
Result = Obj->[IDLffXMLDOMCharacterData::GetLength()]

IDLffXMLDOMCharacterData::InsertData - Inserts a string in the node’s character data, starting at an offset.
Obj->[IDLffXMLDOMCharacterData::InsertData( Offset, String )]

IDLffXMLDOMCharacterData::ReplaceData - Replaces a number of characters, starting at an offset in the node’s character data, with a string.
Obj->[IDLffXMLDOMCharacterData::ReplaceData( Offset, Count, String )]

IDLffXMLDOMCharacterData::SetData - Sets the node’s character data to a string
Obj->[IDLffXMLDOMCharacterData::SetData( String )]

IDLffXMLDOMCharacterData::SubstringData - Returns a string composed of a substring of the node’s character data.
Result = Obj->[IDLffXMLDOMCharacterData::SubstringData( Offset, Count )]

IDLffXMLDOMCDATASection - Used to escape blocks of text in an XML document containing text that would otherwise be regarded as market. Subclass of IDLffXMLDOMNode, IDLffXMLDOMCharacterData, and IDLffXMLDOMText.
IDLffXMLDOMDocument - Represents the entire XML document as the document tree's root and by providing primary access to the document's data. Subclass of IDLffXMLDOMNode.

IDLffXMLDOMComment - Represents the content of a comment (characters between "<!--" and "-->") in an XML document. Subclass of IDLffXMLDOMCharacterData and IDLffXMLDOMText.

IDLffXMLDOMComment::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLDOMDocument - Represents the entire XML document as the document tree's root and by providing primary access to the document’s data. Subclass of IDLffXMLDOMNode.

IDLffXMLDOMDocument::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree.

IDLffXMLDOMDocument::CreateAttribute - Creates and names an attribute node owned by the XML document.

IDLffXMLDOMDocument::CreateCDATASection - Creates and fills a CDATASection node owned by the XML document.

IDLffXMLDOMDocument::CreateComment - Creates and fills a comment node owned by the XML document.

IDLffXMLDOMDocument::CreateDocumentFragment - Creates a document fragment node owned by the XML document.

IDLffXMLDOMDocument::CreateDocumentFragment::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree.

IDLffXMLDOMDocument::CreateElement - Creates and names an element node owned by the XML document.

IDLffXMLDOMDocument::CreateEntityReference - Creates and names an entity reference node owned by the XML document.

IDLffXMLDOMDocument::CreateProcessingInstruction - Creates and stores two strings in a ProcessingInstruction node owned by the XML document.

IDLffXMLDOMDocument::CreateTextNode - Creates and fills a text node owned by the XML document.

IDLffXMLDOMDocument::CreateProcessingInstruction() - Creates and stores two strings in a ProcessingInstruction node owned by the XML document.

IDLffXMLDOMDocument::CreateTextNode() - Creates and fills a text node owned by the XML document.

IDLffXMLDOMDocument::Doctype - Creates an instance of IDLffXMLDOMDocumentType.

IDLffXMLDOMDocument::GetDocumentElement - Creates an instance of IDLffXMLDOMElement.

IDLffXMLDOMDocument::GetElementsByTagName - Creates an IDLffXMLDOMNodeList object containing all element nodes in the XML document with the specified tag name.

IDLffXMLDOMDocument::Init - Initializes the object.

IDLffXMLDOMDocument::Load - Loads and parses XML data from a specified source; creates a DOM document tree (accessed through this object) from the data.

IDLffXMLDOMDocument::Save - Serializes the current DOM document and writes it to an output source.

IDLffXMLDOMDocumentFragment - References a document fragment node. Subclass of IDLffXMLDOMNode.

IDLffXMLDOMDocumentFragment::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

OBJ_DESTROY, Obj
IDLffXMLDOMDocumentType - References a DocumentType node. Subclass of IDLffXMLDOMNode.

IDLffXMLDOMDocumentType::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLDOMDocumentType::GetEntities - Returns the external and internal entities declared in the DTD.

Result = Obj->[IDLffXMLDOMDocumentType::]GetEntities()

IDLffXMLDOMDocumentType::GetName - Returns the DTD's name.

Result = Obj->[IDLffXMLDOMDocumentType::]GetName()

IDLffXMLDOMDocumentType::GetNotations - Returns the notations declared in the DTD.

Result = Obj->[IDLffXMLDOMDocumentType::]GetNotations()

IDLffXMLDOMElement - References an element node. Subclass of IDLffXMLDOMNode.

IDLffXMLDOMElement::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLDOMElement::SetAttribute - Adds and sets a new attribute to the element node.

Result = Obj->[IDLffXMLDOMElement::]setAttribute(Name, Value)

IDLffXMLDOMElement::SetAttributeNode - Adds a new attribute node to the element node.

Result = Obj->[IDLffXMLDOMElement::]setAttributeNode(NewAttr)

IDLffXMLDOMElement::GetAttribute - Returns the value of the named attribute.

Result = Obj->[IDLffXMLDOMElement::]getAttribute(Name)

IDLffXMLDOMElement::GetAttributeNode - Creates an named IDLffXMLDOMAttr object.

Result = Obj->[IDLffXMLDOMElement::]getAttributeNode(Name)

IDLffXMLDOMElement::GetElementsByTagName - Creates an IDLffXMLDOMNodeList object containing all element nodes in the XML document with the specified tag name.

Result = Obj->[IDLffXMLDOMElement::]getElementsByTagName(TagName)

IDLffXMLDOMElement::GetTagName - Returns the element's name.

Result = Obj->[IDLffXMLDOMElement::]getTagName()

IDLffXMLDOMElement::RemoveAttribute - Removes the named attribute from the element node.

Result = Obj->[IDLffXMLDOMElement::]removeAttribute(Name)

IDLffXMLDOMElement::RemoveAttributeNode - Removes the named attribute node from the element node.

Result = Obj->[IDLffXMLDOMElement::]removeAttributeNode(OldAttr)

IDLffXMLDOMElement::SetAttribute - Adds a new attribute to the element node.

Result = Obj->[IDLffXMLDOMElement::]setAttribute(Name, Value)
### Alphabetical List of IDL Routines

**IDLffXMLDOMNode::GetAttributes** - Creates a named node map used to access an element object’s attributes.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetAttributes()} \]

**IDLffXMLDOMNode::GetChildNodes** - Creates a node list used to access the calling node’s children.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetChildNodes()} \]

**IDLffXMLDOMNode::GetFirstChild** - Creates an IDLffXMLDOMNode node that refers to the first child in the DOM tree.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetFirstChild()} \]

**IDLffXMLDOMNode::GetLastChild** - Creates an IDLffXMLDOMNode node that refers to the last child in the DOM tree.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetLastChild()} \]

**IDLffXMLDOMNode::GetNextSibling** - Creates an IDLffXMLDOMNode node that refers to the next sibling in the DOM tree.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetNextSibling()} \]

**IDLffXMLDOMNode::GetNodeName** - Returns the node’s name, depending on the subclass type.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetNodeName()} \]

**IDLffXMLDOMNode::GetNodeType** - Returns the node’s type.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetNodeType()} \]

**IDLffXMLDOMNode::GetNodeValue** - Returns the node’s value, depending on the subclass type.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetNodeValue()} \]

**IDLffXMLDOMNode::GetOwnerDocument** - Returns an object reference to the IDLffXMLDOMDocument used to create the node.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetOwnerDocument()} \]

**IDLffXMLDOMNode::GetParentNode** - Creates an IDLffXMLDOMNode node that refers to the parent in the DOM tree.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetParentNode()} \]

**IDLffXMLDOMNode::GetPreviousSibling** - Creates an IDLffXMLDOMNode node that refers to the previous sibling in the DOM tree.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetPreviousSibling()} \]

**IDLffXMLDOMNode::HasChildNodes** - Returns a value indicating whether the calling node has children.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::HasChildNodes()} \]

**IDLffXMLDOMNode::InsertBefore** - Inserts a new node into the calling node’s children before a reference node.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::InsertBefore}(\text{NewChild}, \text{RefChild}) \]

**IDLffXMLDOMNode::RemoveChild** - Removes an existing node from the calling node’s child list.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::RemoveChild}(\text{OldChild}) \]

**IDLffXMLDOMNode::ReplaceChild** - Replaces an existing child node with a new node in the calling node’s child list.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::ReplaceChild}(\text{NewChild}, \text{OldChild}) \]

**IDLffXMLDOMNode::SetNodeValue** - Sets the node’s value to the contents of a string.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::SetNodeValue}(\text{NodeValue}) \]

**IDLffXMLDOMNodeList::GetLength** - Returns the number of nodes in the node list.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNodeList::GetLength()} \]

**IDLffXMLDOMNodeList::Item** - Returns an object reference to the node that contains the indexed item.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNodeList::Item}(\text{Index}) \]

**IDLffXMLDOMNode::GetPublicID** - Returns the notation’s public ID.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetPublicID()} \]

**IDLffXMLDOMNode::GetSystemID** - Returns the notation’s system ID.

\[ \text{Result} = \text{Obj} \rightarrow \text{IDLffXMLDOMNode::GetSystemID()} \]
IDLffXMLDOMProcessingInstruction - References a processing instruction node in the XML document. Subclass of IDLffXMLDOMNode.

IDLffXMLDOMProcessingInstruction::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLDOMProcessingInstruction::GetData - Returns the content of the processing instruction.

IDLffXMLDOMProcessingInstruction::GetTarget - Returns the target of the processing instruction.

IDLffXMLDOMText - Represents a text node in the XML document. Subclass of IDLffXMLDOMNode and IDLffXMLDOMCharacterData.

IDLffXMLDOMText::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLDOMText::IsIgnorableWhitespace - Indicates whether the text node contains ignorable whitespace.

IDLffXMLDOMText::SplitText - Breaks a text node into two sibling nodes in the same tree, splitting the text at the specified offset.

IDLffXMLSAX - Represents an XML SAX Level 2 parser. No superclasses. In order to use this class, you must write your own subclass.

IDLffXMLSAX::AttributeDecl - Called when the parser detects an <ATTLIST ... > declaration in a DTD.

IDLffXMLSAX::AttributeDecl::Cleanup - Destroys both the accessor object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLSAX::Cleanup - Performs all cleanup on the object.

IDLffXMLSAX::Comment - Called when the parser detects a comment section of the form <!-- .... -->.

IDLffXMLSAX::Comment::Cleanup - Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLSAX::ElementDecl - Called when the parser detects an <ELEMENT ... > declaration in the DTD.

IDLffXMLSAX::ElementDecl::Cleanup - Destroys both the accessor object in the IDL tree and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLSAX::EndCDATA - Called when the parser detects the end of a <![CDATA[...]]> text section.

IDLffXMLSAX::EndElement - Called when the parser detects the end of an element.

IDLffXMLSAX::EndDocument - Called when the parser detects the end of the XML document.

IDLffXMLSAX::EndDTD - Called when the parser detects the end of a Document Type Definition (DTD).

IDLffXMLSAX::EndEntity - Called when the parser detects the end of an internal or external entity expansion.

IDLffXMLSAX::EndPrefixMapping - Called when the previously declared prefix mapping goes out of scope.

IDLffXMLSAX::Errorprocedure - Called when the parser detects error that is not expected to be fatal.

IDLffXMLSAX::ExternalEntityDecl - Called when the parser detects an <!ENTITY ... > declarations in the DTD for a parsed external entity.

IDLffXMLSAX::FatalError - Called when the parser detects a fatal error.

IDLffXMLSAX::GetProperty - Used to get the values of various properties of the parser.

IDLffXMLSAX::IgnorableWhitespace - Called when the parser detects whitespace that separates elements in an element content model.
IDL Quick Reference

IDLgrAxis

IDLffXMLSAX::Init - Initializes an XML parser object.

Object = OBJ_NEW('IDLffXMLSAX', Result = OBJ_NEW('IDLffXMLSAX', PROPERTY = value))

IDLffXMLSAX::StartElement - Called when the parser detects an <ENTITY ...> declaration in a DTD for (parsed) internal entities.

Obj->IDLffXMLSAX::StartElement, Name, Value

IDLffXMLSAX::InternalEntityDecl - Called when the parser detects a &lt;ENTITY ... &gt; declaration in a DTD for (parsed) internal entities.

Obj->IDLffXMLSAX::InternalEntityDecl, Name, Value

IDLffXMLSAX::NotationDecl - Called when the parser detects a &lt;NOTATION ... &gt; declaration in a DTD.

Obj->IDLffXMLSAX::NotationDecl, Name, PublicID, SystemID

IDLffXMLSAX::ParseFile - Parses the specified XML file.

Obj->IDLffXMLSAX::ParseFile, Filename

IDLffXMLSAX::ProcessingInstruction - Called when the parser detects a processing instruction.

Obj->IDLffXMLSAX::ProcessingInstruction, Target, Data

IDLffXMLSAX::SetProperty - Used to set the values of various properties of the parser.

Obj->IDLffXMLSAX::SetProperty, Property = value

IDLffXMLSAX::SkippedEntity - Called when the parser skips an entity and validation is not being performed.

Obj->IDLffXMLSAX::SkippedEntity

IDLffXMLSAX::StartCDATA - Called when the parser detects the beginning of a &lt;CDATA[...]]&gt; text section.

Obj->IDLffXMLSAX::StartCDATA

IDLffXMLSAX::StartDocument - Called when the parser begins processing a document, and before any data is processed.

Obj->IDLffXMLSAX::StartDocument

IDLffXMLSAX::StartDTD - Called when the parser detects the beginning of a Document Type Definition (DTD).

Obj->IDLffXMLSAX::StartDTD, Name, PublicID, SystemID

IDLffXMLSAX::StartElement - Called when the parser detects the beginning of an element.

Obj->IDLffXMLSAX::StartElement, URI, Local, qName, attName, attValue

IDLffXMLSAX::StartElement - Called when the parser detects the start of an internal or external entity expansion.

Obj->IDLffXMLSAX::StartElement, Name

IDLffXMLSAX::StartPrefixmapping - Called when the parser detects the beginning of a namespace declaration.

Obj->IDLffXMLSAX::StartPrefixmapping, Prefix, URI

IDLffXMLSAX::StopParsing - Used during a parse operation to halt the operation and cause the ParseFile method to return.

Obj->IDLffXMLSAX::StopParsing

IDLffXMLSAX::UnparsedEntityDecl - Called when the parser detects an &lt;ENTITY ... &gt; declaration that includes the NDATA keyword, indicating that the entity is not meant to be parsed.

Obj->IDLffXMLSAX::UnparsedEntityDecl, Name, PublicID, SystemID, Notation

IDLffXMLSAX::Warning - Called when the parser detects a problem during processing.

Obj->IDLffXMLSAX::Warning, SystemID, LineNumber, ColumnNumber, Message

IDLgrAxis - Represents a single vector that may include a set of tick marks, tick labels, and a title.

Properties:
- [ , ALL{Get}=variable]
- [ , ALPHACHANNEL{Get, Init, Set}]=variable
- [ , AM_PM{Get, Init, Set}]=array
- [ , COLOR{Get, Init, Set}]=index or RGB_vector
- [ , CRANGE{Get}=variable]
- [ , DAYS_OF_WEEK{Get, Init, Set}]=array
- [ , DEPTHTESTDISABLE{Get, Init, Set}]=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
- [ , DEPTHTESTFUNCTION{Get, Init, Set}]=variable
- [ , DEPTH_WRITE_DISABLE{Get, Init, Set}]=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
- [ , DEPTH_WRITEFUNCTION{Get, Init, Set}]=variable
- [ , DIR{Get, Init, Set}]=integer
- [ , EXACT{Get, Init, Set}]=variable
- [ , EXTEND{Get, Init, Set}]=variable
- [ , GRIDSTYLE{Get, Init, Set}]=integer
- [ , /HIDE{Get, Init, Set}]=variable
- [ , LOCATION{Get, Init, Set}]=x, y, z
- [ , LOG{Get, Init, Set}]=variable
- [ , MAJOR{Get, Init, Set}]=integer
- [ , MINOR{Get, Init, Set}]=integer
- [ , MONTHS{Get, Init, Set}]=array
- [ , NOTEXT{Get, Init, Set}]=variable
- [ , PALETTE{Get, Init, Set}]=objref
- [ , PARENT{Get}=variable]
- [ , RANGE{Get, Init, Set}]=min, max
- [ , REGISTER_PROPERTIES{Get, Init, Set}]=variable
- [ , SUBTICKLEVEL{Get, Init, Set}]=variable
- [ , TEXTALIGNMENTS{Get, Init, Set}]=variable
- [ , TEXTBASELINE{Get, Init, Set}]=variable
- [ , TEXTPOS{Get, Init, Set}]=variable
- [ , TEXTUPDIR{Get, Init, Set}]=variable
- [ , THICK{Get, Init, Set}]=variable
- [ , TICKS{Get, Init, Set}]=variable
- [ , TICKFORMAT{Get, Init, Set}]=variable
- [ , TICKFRMTDATA{Get, Init, Set}]=variable
- [ , TICKINTERVAL{Get, Init, Set}]=variable
- [ , TICKLAYOUT{Get, Init, Set}]=variable
- [ , TICKLERY{Get, Init, Set}]=variable
- [ , TICKUNIT{Get, Init, Set}]=variable
- [ , TICKVALUES{Get, Init, Set}]=variable
- [ , ZRANGE{Get}=variable]
IDLgrAxis::Cleanup - Performs all cleanup on the object.
    OBJ_DESTROY, Obj or Obj->[IDLgrAxis::]Cleanup

IDLgrAxis::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.
    Result = Obj->[IDLgrAxis::]GetCTM( 
        [, DESTINATION=objref] [, PATH=objref(s)] 
        [, TOP=objref] )

IDLgrAxis::GetProperty - Retrieves the value of a property or group of properties for the axis.
    Obj->[IDLgrAxis::]GetProperty [,, PROPERTY=variable]

IDLgrAxis::SetProperty - Sets the value of a property or group of properties for the axis.
    Obj->[IDLgrAxis::]SetProperty [,, PROPERTY=variable]

IDLgrBuffer - An in-memory, off-screen destination object.
    Properties: [,, ALL=[Get]=variable] 
        [, COLOR_MODEL=[Get, Init=[0 | 1]]] 
        [, DIMENSIONS=[Get, Init=[width, height]]] 
        [, GRAPHICS_TREE=[Get, Init, Set]=objref] 
        [, QUALITY=[Get, Init, Set]=[0 | 1 | 2]] 
        [, REGISTER_PROPERTIES=[Get, Init, Set]] 
        [, RESOLUTION=[Get, Init, Set]=[xres, yres]] 
        [, SCREEN_DIMENSIONS=[Get]=variable] 
        [, UNITS=[Get, Init, Set]=[0 | 1 | 2 | 3]] 
        [, ZBUFFER_DATA=[Get]=variable]

IDLgrBuffer::Cleanup - Performs all cleanup on the object.
    OBJ_DESTROY, Obj or Obj->[IDLgrBuffer::]Cleanup

IDLgrBuffer::Draw - Draws picture to this graphics destination.
    Obj->[IDLgrBuffer::]Draw [, Picture] 
        [, CREATE_INSTANCE=[1 | 2]] 
        [, DRAW_INSTANCE]

IDLgrBuffer::Erase - Erases this graphics destination.
    Obj->[IDLgrBuffer::]Erase [,, COLOR=index or RGB vector]

IDLgrBuffer::GetContiguousPixels - Returns an array of long integers whose length is equal to the number of colors available in the index color mode (value of the N_COLORS property).
    Return = Obj->[IDLgrBuffer::]GetContiguousPixels( )
**Alphabetical List of IDL Routines**

IDLgrClipboard - A destination object representing the native clipboard.

**Properties:**
- ALL\{Get\}=[variable]
- COLOR\_MODEL\{Get\}=[variable]
- DIMENSIONS\{Get\}=[width, height]
- GRAPHICS\_TREE\{Get\}=[variable]
- N\_COLORS\{Get\}=[integer]\[
- PALETTE\{Get\}=[variable]
- QUALITY\{Get\}=[variable]
- RESOLUTION\{Get\}=[variable]
- SCREEN\_DIMENSIONS\{Get\}=[variable]
- UNITS\{Get\}=[variable]

IDLgrClipboard::Cleanup - Performs all cleanup on the object.

Example:
```
OBJ\_DESTROY, Obj or Obj->[IDLgrClipboard::]Cleanup
```

IDLgrClipboard::Draw - Draws a picture to a graphics destination.

Example:
```
Obj->[IDLgrClipboard::]Draw [, Picture] [, CMKY]
```

IDLgrClipboard::GetContiguousPixels - Returns array of long integers whose length is equal to the number of colors available in the index color mode (value of the N\_COLORS property).

Example:
```
Return = Obj->[IDLgrClipboard::]GetContiguousPixels()
```

IDLgrClipboard::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.

Example:
```
Obj->[IDLgrClipboard::]GetDeviceInfo [, ALL=variable]
```

IDLgrClipboard::GetFontnames - Returns the list of available fonts that can be used in IDLgrFont objects.

Example:
```
Return = Obj->[IDLgrClipboard::]GetFontnames( FamilyName [, IDL\_FONTS=[0 | 1 | 2]] [, STYLE=string])
```

IDLgrClipboard::GetProperty - Retrieves the value of a property or group of properties for the clipboard buffer.

Example:
```
Obj->[IDLgrClipboard::]GetProperty [\
  PROPERTY=variable]
```

IDLgrClipboard::GetTextDimensions - Retrieves the dimensions of a text object that will be rendered in the clipboard buffer.

Example:
```
Result = Obj->[IDLgrClipboard::]GetTextDimensions( TextObj [, DESCENT=variable] [, PATH=objcref(s)])
```

IDLgrClipboard::Init - Initializes the clipboard object.

Example:
```
Obj = OBJ\_NEW([IDLgrClipboard\{, PROPERTY=value\}])
```

IDLgrClipboard::SetProperty - Sets the value of a property or group of properties for the clipboard buffer.

Example:
```
Obj->[IDLgrClipboard::]SetProperty [\
  PROPERTY=variable]
```

IDLgrColorbar - Consists of a color-ramp with an optional framing box and annotation axis.

**Properties:**
- ALL\{Get\}=[variable]
- BLUE\_VALUES\{Get\}=[variable]
- COLOR\{Get\}=[variable]
- DIMENSIONS\{Get\}=[variable]
- GREEN\_VALUES\{Get\}=[variable]
- HIDE\{Get\}=[variable]
- MAJOR\{Get\}=[variable]
- MINOR\{Get\}=[variable]
- PARENT\{Get\}=[variable]
- RED\_VALUES\{Get\}=[variable]
- SHOW\_AXIS\{Get\}=[variable]
- SHOW\_OUTLINE\{Get\}=[variable]
- SUB\_TICK\{Get\}=[variable]
- TITLE\{Get\}=[variable]
- THICK\{Get\}=[variable]
- TICK\{Get\}=[variable]
- TICK\_FORMAT\{Get\}=[variable]
- TICK\_VALUES\{Get\}=[variable]
- XCOORD\_CONV\{Get\}=[variable]
- YCOORD\_CONV\{Get\}=[variable]
- ZRANGE\{Get\}=[variable]
- ZCOORD\_CONV\{Get\}=[variable]

IDLgrColorbar::Cleanup - Performs all cleanup on the object.

Example:
```
OBJ\_DESTROY, Obj or Obj->[IDLgrColorbar::]Cleanup
```

IDLgrColorbar::ComputeDimensions - Retrieves the dimensions of a colorbar object for the given destination object.

Example:
```
Result = Obj->[IDLgrColorbar::]ComputeDimensions( DestinationObj [, PATH=objcref(s)])
```

IDLgrColorbar::GetProperty - Retrieves the value of a property or group of properties for the colorbar.

Example:
```
Obj->[IDLgrColorbar::]GetProperty [\
  PROPERTY=variable]
```

IDLgrColorbar::Init - Initializes the colorbar object.

Example:
```
Obj = OBJ\_NEW([IDLgrColorbar\{, aRed, aGreen, aBlue\} [, PROPERTY=value]])
```

IDLgrColorbar::SetProperty - Sets the value of a property or group of properties for the colorbar.

Example:
```
Obj->[IDLgrColorbar::]SetProperty [\
  PROPERTY=variable]
```
IDLgrContour - Draws a contour plot from an array or unstructured point data. No superclasses. No subclasses.

Properties: [: ALL{Get}=
i\[variable\] 
[: ALPHA_CHANNEL{Get, Init, Set}=value] 
[: AM_PM{Get, Init, Set}=vector of two strings] 
[: ANISOTROPY{Get, Init, Set}=[x, y, z]] 
[: C_COLOR{Get, Init, Set}=vector] 
[: C_FILL_PATTERN{Get, Init, Set}=array of IDLgrPattern objects] 
[: C_LABEL_INTERVAL{Get, Init, Set}=vector] 
[: C_LABEL_OBJECTS{Get, Init, Set}=array of object references] 
[: C_LABEL_SHOW{Get, Init, Set}=vector] 
[: C_LINESTYLE{Get, Init, Set}=vector of linestyles] 
[: C_THICK{Get, Init, Set}=float array[each element 1.0 to 10.0]] 
[: C_USE_LABEL_COLOR{Get, Init, Set}=vector of values] 
[: C_USE_LABEL_ORIENTATION{Get, Init, Set}=vector of values] 
[: C_VALUE{Get, Init, Set}=scalar or vector] 
[: CLIP_PLANES{Get, Init, Set}=array of seven strings] 
[: DEPTH_OFFSET{Get, Init, Set}=value] 
[: DEPTH_TEST_DISABLE{Get, Init, Set}=[0 | 1 | 2] 3 | 4 | 5 | 6 | 7 | 8] [] DEPTH_WRITE_DISABLE{Get, Init, Set}=[] DOUBLE_DATA{Get}] [] DOUBLE_GEOM{Init}] [] DOWNHILL{Get, Init, Set}=[] GEOM{Get}][variable] 
[: GEOM{Init, Set}=vector or 2D array] 
[: GEOM{Init, Set}=vector or 2D array] 
[: GEOM{Init, Set}=vector or 2D array] 
[: GEOMZ{Init, Set}=vector or 2D array] 
[: HIDE{Get, Init, Set}] [] LABEL_FONT{Get, Init, Set}=objref] [] LABEL_FORMAT{Get, Init, Set}=string] [] LABEL_FRTMDATA{Init}=[value] [] LABEL_UNITS{Get, Init, Set}=string] 
[: MAX_VALUE{Get, Init, Set}=value] 
[: MIN_VALUE{Get, Init, Set}=value] 
[: MONTHS{Get, Init, Set}=vector of 12 values] 
[: N_LEVELS{Get, Init, Set}=[value] Palette{Get, Init, Set}=objref} [] PARENT{Get}=[variable] 
[: PLANAR{Get, Init, Set}=vector] [] POLYGONS{Get, Init, Set}=array of polygon descriptions] 
[: REGISTER_PROPERTIES{Get, Init, Set}=vector] 
[: SHADE_RANGE{Get, Init, Set}=[min, max] ] 
[: SHADING{Get, Init, Set}=array of polygons] 
[: TICKINTERVAL{Get, Set}=value] [] TICKLEN{Get, Init, Set}=value] [] USE_TEXT_ALIGNMENT{Get, Init, Set}=value] [] XCOORD_CONV{Get, Init, Set}=vector] [] XRANGE{Get}=[variable] 
[: YCOORD_CONV{Get, Init, Set}=vector] [] YRANGE{Get}=variable] [] ZCOORD_CONV{Get, Init, Set}=vector] [] ZRANGE{Get}=variable]

IDLgrContour::Cleanup - Performs all cleanup on the object.

IDLgrContour::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object

Result = Obj->[IDLgrContour::GetCTM [, DESTINATION=objref] [, PATH=objref(s)] [, TOP=objref] ]

IDLgrContour::GetLabelInfo - Retrieves information about the labels for a contour

Result = |IDLgrContour::GetLabelInfo[, Destination, LevelIndex [, LABEL_OFFSETS=variable] [, LABEL_POLYS=variable] [, LABEL_OBJECTS=variable] ]

IDLgrContour::GetProperty - Retrieves the value of a property or group of properties for the contour.

Obj->[IDLgrContour::GetProperty[, PROPERTY=variable] ]

IDLgrContour::Init - Initializes the contour object.

Obj = OBJ_NEW('IDLgrContour'[, Values] [, PROPERTY=variable] ) or
Result = Obj->[IDLgrContour::Init([Values] [, PROPERTY=variable] )]

IDLgrContour::SetProperty - Sets the value of a property or group of properties for the contour.

Obj->[IDLgrContour::SetProperty[, PROPERTY=variable] ]

IDLgrFont - Represents a typeface, style, weight, and point size that may be associated with text objects.

Properties: [: ALL{Get}=[variable] [, SIZE=Get, Set]=points] [, SUBSTITUTE{Get, Init, Set}='Hershey' | 'Courier' | 'Times' | 'Symbol' | 'Helvetica' | 'Courier' | 'Times' | 'Symbol' ]

IDLgrFont::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->[IDLgrFont::Cleanup]

IDLgrFont::GetProperty - Retrieves the value of a property or group of properties for the font.

Obj->[IDLgrFont::GetProperty[, PROPERTY=variable] ]

IDLgrFont::Init - Initializes the font object.

Obj = OBJ_NEW('IDLgrFont'[, Fontname ] [, PROPERTY=variable] ) or
Result = Obj->[IDLgrFont::Init([Fontname] [, PROPERTY=variable] )]

IDLgrFont::SetProperty - Sets the value of a property or group of properties for the font.

Obj->[IDLgrFont::SetProperty[, PROPERTY=variable] ]
IDLgrImage - Represents a mapping from a 2D array of data values to a 2D array of pixel colors, resulting in a flat 2D-scaled version of the image, drawn at Z = 0.

**Properties:**
- ALL: GET = variable
- BLEND_FUNCTION: GET = [vector]
- CHANNEL: GET, INIT, SET = [hexadecimal bitmask]
- CLIP_PLANES: GET, INIT, SET = [array]
- DATA: GET, INIT, SET = [nxm, 2xnxm, 3xnxm, or 4xnxm array of image data]
- DEPTH_TEST_DISABLE: GET, INIT, SET = [0 | 1 | 2]
- DEPTH_TEST_FUNCTION: GET, INIT, SET = [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8]
- DEPTH_WRITE_DISABLE: GET, INIT, SET = [0 | 1 | 2]
- DIMENSIONS: GET, INIT, SET = [width, height]
- /GREYSCALE: GET, INIT, SET = [true | false]
- LOCATION: GET, INIT, SET = [x, y, z, w]
- /NO_COPY: GET, INIT, SET = [true | false]
- /ORDER: GET, INIT, SET = [true | false]
- PALETTE: GET, INIT, SET = [objref]
- PARENT: GET = [objref]
- SHARE_DATA: GET, INIT, SET = [objref]
- SUB_RECT: GET, INIT, SET = [x, y, xdim, ydim]
- XCOORD_CONV: GET, INIT, SET = [vector]
- XRANGE: GET = [variable]
- YCOORD_CONV: GET, INIT, SET = [vector]
- YRANGE: GET = [variable]
- ZCOORD_CONV: GET, INIT, SET = [vector]
- ZRANGE: GET = [variable]

**IDLgrImage::Cleanup** - Performs all cleanup on the object.

**IDLgrImage::GetCTM** - Returns the 4 x 4 graphics transform matrix from the current object.

**IDLgrImage::GetProperty** - Retrieves the value of a property or a group of properties for the image.

**IDLgrImage::Init** - Initializes the image object.

**IDLgrImage::SetProperty** - Sets the value of a property or a group of properties for the image.

---

IDLgrLegend - Provides a simple interface for displaying a legend.

**Properties:**
- ALL: GET = variable
- BORDER_GAP: GET, INIT, SET = [integer]
- COLUMNN: GET, INIT, SET = [integer]
- FILL_COLOR: GET, INIT, SET = [index or RGB vector]
- FONT: GET, INIT, SET = [objref]
- GAP: GET, INIT, SET = [integer]
- GLYPH_WIDTH: GET, INIT, SET = [value]
- /HIDE: GET, INIT, SET = [true | false]
- ITEM_COLOR: GET, INIT, SET = [array of colors]
- ITEM_LINE_STYLE: GET, INIT, SET = [integer array]
- ITEM_NAME: GET, INIT, SET = [string array]
- ITEM_OBJECT: GET, INIT, SET = [IDLgrSymbol or IDLgrPattern objref(s)]
- ITEM_THICK: GET, INIT, SET = [float array] (each element 0.0 to 10.0)
- ITEM_TYPE: GET, INIT, SET = [integer array]
- OUTLINE_COLOR: GET, INIT, SET = [index or RGB vector]
- OUTLINE_THICK: GET, INIT, SET = [float array] (each element 0.0 to 10.0)
- PARENT: GET = variable
- RECOMPUTE: SET = [0 | 1] (0 prevents recompute, 1 is the default)
- /SHOW_FILL: GET, INIT, SET = [true | false]
- /SHOW_OUTLINE: GET, INIT, SET = [true | false]
- TEXT_COLOR: GET, INIT, SET = [index or RGB vector]
- TITLE: GET, INIT, SET = [objref]
- ITEM_COLOR: GET, INIT, SET = [array of colors]
- ITEM_OBJECT: GET, INIT, SET = [array of IDLgrSymbol or IDLgrPattern objref(s)]

**IDLgrLegend::Cleanup** - Performs all cleanup on the object.

**IDLgrLegend::ComputeDimensions** - Retrieves the dimensions of a legend object for the given destination object.

**IDLgrLegend::GetProperty** - Retrieves the value of a property or a group of properties for the legend.

**IDLgrLegend::Init** - Initializes the legend object.

**IDLgrLegend::SetProperty** - Sets the value of a property or a group of properties for the legend.
IDLgrLight – Represents a source of illumination for 3D graphic objects.

Properties: [ ALL{Get}=variable ]
  [ ATTENUATION[Get, Init, Set]=constant, linear, quadratic ]
  [ CONEANGLE[Get, Init, Set]=degrees ]
  [ DIRECTION[Get, Init, Set]=3-element vector ]
  [ FOCUS[Get, Init, Set]=value ]
  [ HIDE[Get, Init, Set]=true/false ]
  [ INTENSITY[Get, Init, Set]=value ]
  [ LOCATION[Get, Init, Set]=[x, y, z] ]
  [ PARENT[Get]=variable ]
  [ REGISTER_PROPERTIES[Get, Init, Set]=true ]
  [ TYPE[Get, Init, Set]=[0 | 1 | 2 | 3] ]
  [ XCOORD_CONV[Get, Init, Set]=vector ]
  [ YCOORD_CONV[Get, Init, Set]=vector ]
  [ ZCOORD_CONV[Get, Init, Set]=vector ]

IDLgrLight::Cleanup - Performs all cleanup on the object.

IDLgrLight::GetCTM - Returns a 4x4 graphics transform matrix from the current object.

IDLgrLight::GetProperty - Retrieves the value of a property or group of properties for the light.

Obj->IDLgrLight::GetProperty( PROPERTY=variable )

IDLgrLight::Init - Initializes the light object.

Obj = OBJ_NEW('IDLgrLight'[ PROPERTY=variable ]

IDLgrLight::SetProperty - Sets the value of a property or group of properties for the light.

Obj->IDLgrLight::SetProperty( PROPERTY=variable )

IDLgrModel – Represents a graphical item or group of items that can be transformed (rotated, scaled, and/or translated).

Properties: [ ALL{Get}=variable ]
  [ CLIP_PLANES[Get, Init, Set]=array ]
  [ DEPTH_TEST_DISABLE[Get, Init, Set]=0 | 1 | 2 ]
  [ DEPTH_TEST_FUNCTION[Get, Init, Set]=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ]
  [ DEPTH_WRITE_DISABLE[Get, Init, Set]=0 | 1 | 2 ]
  [ LIGHTING[Get, Init, Set]=0 | 1 | 2 ]
  [ PARENT[Get]=variable ]
  [ REGISTER_PROPERTIES[Get, Init, Set]=true ]
  [ SELECT_TARGET[Get, Init, Set]=true ]
  [ TRANSFORM[Get, Init, Set]=4x4 transformation matrix ]

IDLgrModel::Add - Adds a child to this Model.

Obj->IDLgrModel::Add( Object[ ALLIAS ]

IDLgrModel::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLgrModel::Cleanup

IDLgrModel::Draw - Draws the specified picture to the specified graphics destination. This method is provided for purposes of subclassing only, and is intended to be called only from the Draw method of a subclass of IDLgrModel.

Obj->IDLgrModel::Draw( Destination, Picture )

IDLgrModel::GetByAlias - Finds contained objects by name and returns the object reference to the named object.

Result = Obj->IDLgrModel::GetByAlias( Name )

IDLgrModel::GetCTM - Returns the 4x4 graphics transform matrix from the current object

Result = Obj->IDLgrModel::GetCTM( DESTINATION=objectref, PATH=objectref(s) )

IDLgrModel::GetProperty - Retrieves the value of a property or group of properties for the model.

Obj->IDLgrModel::GetProperty( PROPERTY=variable )

IDLgrModel::Init - Initializes the model object.

Obj = OBJ_NEW( 'IDLgrModel'[ PROPERTY=variable ]

IDLgrModel::Reset - Sets the current transform matrix for the model object to the identity matrix.

Obj->IDLgrModel::Reset

IDLgrModel::Rotate - Rotates the model about the specified axis by the specified angle.

Obj->IDLgrModel::Rotate( Axis, Angle[ /PREMULTIPLY ]

IDLgrModel::Scale - Scales the model by the specified scaling factors.

Obj->IDLgrModel::Scale( Sx, Sy, Sz[ /PREMULTIPLY ]

IDLgrModel::SetProperty - Sets the value of a property or group of properties for the model.

Obj->IDLgrModel::SetProperty( PROPERTY=variable )

IDLgrModel::Translate - Translates the model by the specified translation offsets.

Obj->IDLgrModel::Translate( Tx, Ty, Tz[ /PREMULTIPLY ]

IDLgrMPEG – Creates an MPEG movie from an array of image frames.

Properties: [ ALL{Get}=variable ]
  [ BITRATE[Get, Init, Set]=value ]
  [ FILENAME[Get, Init, Set]=string ]
  [ FORMAT[Get, Init, Set]=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ]
  [ FRAME_RATE[Get, Init, Set]=1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ]
  [ FRAME_GAP[Get, Init, Set]=integer value ]
  [ /INTERLACED[Get, Init, Set]=true ]
  [ MOTION_VEC_LENGTH[Get, Init, Set]=1 | 2 | 3 ]
  [ QUALITY[Get, Init, Set]=value ]
  [ SCALE[Get, Init, Set]=xscale, yscale ]
  [ /STATISTICS[Get, Init, Set]=true ]
  [ TEMP_DIRECTORY[Get, Init, Set]=string ]

IDLgrMPEG::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLgrMPEG::Cleanup
IDLgrPalette::GetProperty - Retrieves the value of a property or group of properties for the palette.

IDLgrPalette::SetProperty - Sets the value of a property or group of properties for the palette.

IDLgrMPEG::Init - Initializes the MPEG object.

IDLgrMPEG::Put - Puts a given image into the MPEG sequence at the specified frame.

IDLgrMPEG::Save - Encodes and saves an MPEG sequence to a file.

IDLgrPalette - Represents a color lookup table that maps indices to red, green, and blue values.

IDLgrPalette::NearestColor - Returns the index of the color in the palette that matches the given RGB values.

IDLgrPalette::SetRGB - Sets the color values at a specified index in the palette to the specified Red, Green and Blue values.

IDLgrPattern - Describes which pixels are filled and which are left blank when an area is filled.

IDLgrPattern::Cleanup - Performs all cleanup on the object.

IDLgrPattern::GetProperty - Retrieves the value of a property or group of properties for the pattern.

IDLgrPattern::SetProperty - Sets the value of a property or group of properties for the pattern.

IDLgrPlot - Creates set of polylines connecting data points in 2D space.

IDLgrPalette::LoadCT - Loads one of the IDL predefined color tables into an IDLgrPalette object.

IDLgrPalette::NearestColor - Returns the index of the color in the palette that matches the given RGB values.

IDLgrPalette::SetRGB - Sets the color values at a specified index in the palette to the specified Red, Green and Blue values.

IDL Quick Reference
IDLgrPlot::Cleanup - Performs all cleanup on the object.
Obj->IDLgrPlot::Cleanup

IDLgrPlot::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.
Result = Obj->IDLgrPlot::GetCTM()
[ , DESTINATION=Objref [] , PATH=Objref(s) ] [ , TOP=Objref to IDLgrModel object ]

IDLgrPlot::GetProperty - Retrieves the value of the property or group of properties for the plot.
Obj->IDLgrPlot::GetProperty(), PROPERTY=variable

IDLgrPlot::Init - Initializes the plot object.
Obj = OBJ_NEWIDLgrPlot(), [X, Y] [] , PROPERTY=value] or
Result = Obj->IDLgrPlot::Init([X, Y], [PROPERTY=value])

IDLgrPlot::SetProperty - Sets the value of the property or group of properties for the plot.
Obj->IDLgrPlot::SetProperty(), PROPERTY=variable

IDLgrPolygon - Represents one or more polygons that share a set of vertices and rendering attributes.
Properties: [], ALL=variable
[ , ALPHA_CHANNEL=Get, Init, Set=value ]
[ , AMBIENT=Get, Init, Set=RGB vector ]
[ , BOTTOM=Get, Init, Set=index or RGB vector ]
[ , CLIP_PLANES=Get, Init, Set=array ]
COLOR=Get, Init, Set=index or RGB vector , VERT_COLORS=Get, Init, Set=value ]
DATA=Get, Init, Set=array ]
[ , DEPTH_OFFSET=Get, Init, Set=value ]
[ , DEPTH_TEST_DISABLE=Get, Init, Set=[0 | 1 | 2] ]
[ , DEPTH_TEST_FUNCTION=Get, Init, Set=[0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9] ]
[ , DEPTH_WRITE_DISABLE=Get, Init, Set=value ]
[ , DOUBLE=Get, Init, Set=value ]
[ , EMMISSION=Get, Init, Set=RGB vector ]
[ , FILL_PATTERN=Get, Init, Set=objetref to IDLgrPattern object ]
[ , HIDDEN_LINES=Get, Init, Set=variable ]
[ , HID=Get, Init, Set=string ]
[ , LINESTYLE=Get, Init, Set=value ]
[ , NAME=Get, Init, Set=value ]
[ , NORMALS=Get, Init, Set=variable ]
[ , PALETTE=Get, Init, Set=variable ]
[ , PARENT=Get, Init, Set=variable ]
[ , POLYGONS=Get, Init, Set=array of polygon descriptions ]
[ , REGISTER_PROPERTIES=Get, Init, Set=[] ]
[ , REJECT=Get, Init, Set=[0 | 1 | 2] ]
[ , RESET_DATA=Get, Init, Set=variable ]
[ , SHADE_RANGE=Get, Init, Set=variable ]
[ , SHADING=Get, Init, Set=value ]
[ , SHININESS=Get, Init, Set=value ]
[ , SPECULAR=Get, Init, Set=RGB vector ]
[ , STYLE=Get, Init, Set=[0 | 1 | 2] ]
[ , TEXTURE_COORD=Get, Init, Set=variable ]
[ , TEXTURE_INTERP=Get, Init, Set=variable ]
[ , TEXTURE_MAP=Get, Init, Set=objetref to IDLgrImage object ]
[ , THICK=Get, Init, Set=points{1.0 to 10.0} ]

[ , XCOORD_CONV=Get, Init, Set=vector ]
[ , XRANGE=Get=variable ]
[ , YCOORD_CONV=Get, Init, Set=vector ]
[ , ZCOORD_CONV=Get, Init, Set=vector ]
[ , ZRANGE=Get=variable ]

IDLgrPolygon::Cleanup - Performs all cleanup on the object.
Obj->IDLgrPolygon::Cleanup

IDLgrPolygon::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.
Result = Obj->IDLgrPolygon::GetCTM()
[ , DESTINATION=Objref [] , PATH=Objref(s) ] [ , TOP=objetref to IDLgrModel object ]

IDLgrPolygon::GetProperty - Retrieves the value of the property or group of properties for the polygons.
Obj->IDLgrPolygon::GetProperty(), PROPERTY=variable

IDLgrPolygon::Init - Initializes the polygons object.
Obj = OBJ_NEWIDLgrPolygon[], [X, Y, Z] ] , PROPERTY=value] or
Result = Obj->IDLgrPolygon::Init([X, Y, Z], [PROPERTY=value])

IDLgrPolygon::SetProperty - Sets the value of the property or group of properties for the polygons.
Obj->IDLgrPolygon::SetProperty(), PROPERTY=variable

IDLgrPolyline - Represents one or more polylines that share a set of vertices and rendering attributes.
Properties: [], ALL=variable
[ , ALPHA_CHANNEL=Get, Init, Set=value ]
[ , CLIP_PLANES=Get, Init, Set=variable ]
[ , COLOR=Get, Init, Set=RGB vector ]
[ , DIFFUSE=Get, Init, Set=variable ]
[ , DEPTH_TEST_DISABLE=Get, Init, Set=[0 | 1 | 2] ]
[ , DEPTH_TEST_FUNCTION=Get, Init, Set=[0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9] ]
[ , DEPTH_WRITE_DISABLE=Get, Init, Set=value ]
[ , DOUBLE=Get, Init, Set=value ]
[ , EMIS=Get, Init, Set=variable ]
[ , FILL_PATTERN=Get, Init, Set=variable ]
[ , HIDDEN_LINES=Get, Init, Set=variable ]
[ , HID=Get, Init, Set=variable ]
[ , LINESTYLE=Get, Init, Set=variable ]
[ , GROUP=Get, Init, Set=variable ]
[ , LABEL=Get, Init, Set=variable ]
[ , NORMALS=Get, Init, Set=variable ]
[ , PALETTE=Get, Init, Set=variable ]
[ , PARENT=Get, Init, Set=variable ]
[ , POLYGONS=Get, Init, Set=variable ]
[ , REGISTER_PROPERTIES=Get, Init, Set=[] ]
[ , REJECT=Get, Init, Set=[0 | 1 | 2] ]
[ , RESET_DATA=Get, Init, Set=variable ]
[ , SHADE_RANGE=Get, Init, Set=variable ]
[ , SHADING=Get, Init, Set=variable ]
[ , SHININESS=Get, Init, Set=variable ]
[ , SPECULAR=Get, Init, Set=RGB vector ]
[ , STYLE=Get, Init, Set=[0 | 1 | 2] ]
[ , TEXTURE_COORD=Get, Init, Set=variable ]
[ , TEXTURE_INTERP=Get, Init, Set=variable ]
[ , TEXTURE_MAP=Get, Init, Set=variable ]
[ , THICK=Get, Init, Set=variable ]

[ , XCOORD_CONV=Get, Init, Set=vector ]
[ , XRANGE=Get=variable ]
[ , YCOORD_CONV=Get, Init, Set=vector ]
[ , ZCOORD_CONV=Get, Init, Set=vector ]
[ , ZRANGE=Get=variable ]

[ , ZERO_OPACITY_SKIP=Get, Init, Set=[0 | 1 | 2] ]
[ , ZRANGE=Get=variable ]
IDLgrPolyLine Properties - continued

[USE_LABEL_COLOR{Get, Init, Set}=vector]
[USE_LABEL_ORIENTATION{Get, Init, Set}=vector]
[USE_TEXT_ALIGNMENTMENTS{Get, Init, Set}=vector]
[XCOORD_CONV{Get, Init, Set}=vector]
[YCOORD_CONV{Get, Init, Set}=vector]
[XRANGE{Get}=variable], YRANGE{Get}=variable]
[ZCOORD_CONV{Get, Init, Set}=vector]
[ZRANGE{Get}=variable]

IDLgrPolyLine::Cleanup - Performs all cleanup on the object.

IDLgrPolyLine::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.

IDLgrPolyLine::GetProperty - Retrieves the value of a property or group of properties for the polylines.

IDLgrPolyLine::Init - Initializes the polylines object.

IDLgrPolyLine::SetProperty - Sets the value of a property or group of properties for the polylines.

IDLgrPrinter - Represents a hardcopy graphics destination.

IDLgrPrinter::GetContiguousPixels - Returns an array of long integers whose length is equal to the number of colors available in the index color mode (value of N_COLORS property).

_IDLgrPrinter::GetFontnames - Returns the list of available fonts that can be used in IDLgrFont objects.

IDLgrROI - Object graphics representation of a region of interest.

Properties: [ALL{Get}=variable]

[ALPHA_CHANNEL{Get, Init, Set}=value]

[CLIP_PLANES( Get, Init, Set )=array]

[COLOR{Get, Init, Set}=vector]

[DEPTH_TEST_DISABLE{Get, Init, Set}=0 | 1 | 2]

[DEPTH_TEST_FUNCTION{Get, Init, Set}=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9]

[DEPTH_WRITE_DISABLE{Get, Init, Set}=0 | 1 | 2]

[DUAL{Get, Init, Set}=0 | 1 | 2]

[DOUBLE{Get, Init, Set}=variable]

[EQUAL{Get, Init, Set}=0 | 1 | 2 | 3]

[EQUAL{Get, Init, Set}=0 | 1 | 2 | 3]

[OPEN{Get, Init, Set}=0 | 1 | 2 | 3]

[LINESTYLE{Get, Init, Set}=variable]

[LINETYPE{Get, Init, Set}=variable]

[REGISTER_PROPERTIES{Get, Init, Set}=variable]

[RENDER{Get, Init, Set}=variable]

[RESOLUTION{Get}=variable], UNITS=variable]

[TEXT_ALIGNMENTS{Get, Init, Set}=variable]

[TEXT_FONT{Get, Init, Set}=variable]

[THICK{Get, Init, Set}=variable]

[XCOORD_CONV{Get, Init, Set}=variable]

[YCOORD_CONV{Get, Init, Set}=variable]

[ZCOORD_CONV{Get, Init, Set}=variable]

[ZRANGE{Get}=variable]
IDLgrROI::GetProperty - Retrieves the value of a property or group of properties for the Object Graphics region.

Obj -> [IDLgrROI::GetProperty[, PROPERTY=variable]]

IDLgrROI::Init - Initializes an Object Graphics region of interest.

Obj = OBJ_NEW( 'IDLgrROI' [, X, Y, Z])
[, PROPERTY=value]

Result = Obj -> [IDLgrROI::Init[X, Y, Z]]
[, PROPERTY=value]

IDLgrROI::PickRegion - Picks a region within the group that, when projected onto the given destination device, is nearest to the given 2D device coordinate.

Result = Obj -> [IDLgrROI::PickRegion(Dest, View, Point [, PATH=objcref])]

IDLgrROIGroup::Add - Adds a region to the region group.

Obj = OBJ_NEW( 'IDLgrROIGroup' [, Obj])

Result = Obj -> [IDLgrROIGroup::Add[Obj]]
[, PROPERTY=value]

IDLgrROIGroup::Cleanup - Performs all cleanup for the object.

Result = Obj -> [IDLgrROIGroup::Cleanup]

IDLgrROIGroup::GetProperty - Retrieves the value of a property or group of properties for the region group.

Obj -> [IDLgrROIGroup::GetProperty[, PROPERTY=variable]]

IDLgrROIGroup::SetProperty - Sets the value of a property or group of properties for the region group.

Obj -> [IDLgrROIGroup::SetProperty[PROPERTY=variable]]
### IDLgrSurface Properties - continued

- MIN_VALUE{Get, Init, Set}={value} [, PALETTE{Get, Init, Set}=objref] [, REGISTER_PROPERTIES{Get, Init, Set}]] [, /RESET_DATA{Init, Set}]]
- SHADE_RANGE{Get, Init, Set}={index of darkest pixel, index of brightest pixel]} [, SHADING{Get, Init, Set}=[0 | 1]} [, SHARE_DATA{Init, Set}=objref] [, SHININESS{Get, Init, Set}=value] [, /SHOW_SKIRT{Get, Init, Set}]} [, SKIRT{Get, Init, Set}]=value] [, SPECULAR{Get, Init, Set}=RGB vector] [, STYLE{Get, Init, Set}=[0 | 1 | 2 | 3 | 4 | 5 | 6]} [, TEXTURE_COORD{Get, Init, Set}=array] [, /TEXTURE_HIGHRES{Get, Init, Set}]} [, /TEXTURE_INTERP{Get, Init, Set}]} [, TEXTURE_MAP{Get, Init, Set}=objref to IDLgrImage} [, THICK{Get, Init, Set}]=points{1.0 to 10.0}] [, /USE_TRIANGLES{Get, Init, Set}]} [, VERT_COLORS{Get, Init, Set}=vector] [, XCOORD_CONV{Get, Init, Set}=vector] [, YCOORD_CONV{Get, Init, Set}=vector] [, ZCOORD_CONV{Get, Init, Set}=vector] [, ZERO_OPACITY_SKIP{Get, Init, Set}=value] [, ALL{Get}={value} [, PALETTE{Get, Init, Set}=objref] [, REGISTER_PROPERTIES{Get, Init, Set}]] [, /RESET_DATA{Init, Set}]]

### IDLgrSurface::Cleanup
- Performs all cleanup on the object.

### IDLgrSurface::GetCTM
- Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.

### IDLgrSurface::GetProperty
- Retrieves the value of a property or group of properties for the symbol.

### IDLgrSymbol::GetProperty
- Retrieves the value of a property or group of properties for the symbol.

### IDLgrSurface::Init
- Initializes the surface object.

### IDLgrSymbol::Init
- Initializes the plot symbol.

### IDLgrSurface:: SetProperty
- Sets the value of a property or group of properties for the symbol.

### IDLgrSymbol::SetProperty
- Sets the value of a property or group of properties for the symbol.

### IDLgrTessellator
- Converts a simple concave polygon (or a simple polygon with "holes") into a number of simple convex polygons (general triangles).

### IDLgrTessellator::AddPolygon
- Adds a polygon to the tessellator object.

### IDLgrTessellator::Cleanup
- Performs all cleanup on the object.

### IDLgrTessellator::Reset
- Resets the object’s internal state.

### IDLgrText
- Represents one or more text strings that share common rendering attributes.

### IDLgrSymbol::Symbol
- Represents a graphical element that is plotted relative to a particular position.

### IDLgrText
- Represents one or more text strings that share common rendering attributes.

### IDLgrTessellator::Tessellate
- Performs the actual tessellation.

### IDLgrTessellator::Tessellate
- Performs the actual tessellation.

### IDLgrTessellator::Tessellate
- Performs the actual tessellation.

### IDLgrTessellator::Tessellate
- Performs the actual tessellation.
IDLgrText::Cleanup - Performs all cleanup on the object.
Obj_DESTROY, Obj or Obj->IDLgrText::Cleanup

IDLgrText::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.
Result = Obj->IDLgrText::GetCTM([ DESTINATION=objetref], [ PATH=objetref(s)]), [ TOP=objet to IDLgrModel object])

IDLgrText::GetProperty - Retrieves the value of a property or group of properties for the text.
Obj->IDLgrText::GetProperty(), PROPERTY=variable

IDLgrText::Init - Initializes the text object.
Obj = OBJ_NEW('IDLgrText'[, String/string array] [ , PROPERTY=value]) or
Result = Obj->IDLgrText::Init(String/string array) [ , PROPERTY=value]

IDLgrText::SetProperty - Sets the value of a property or group of properties for the text.
Obj->IDLgrText::SetProperty(), PROPERTY=variable

IDLgrView - Represents a rectangular area in which graphics objects are drawn. It is a container for objects of the IDLgrModel class.
Properties: [ , ALL={Get=variable} [ , PARENT={Get=variable} [ , COLOR={Get, Init, Set=index or RGB vector} [ , DEPTH_CUE={Get, Init, Set}={0 | 1 | 2}]] , DIMENSIONS={Get, Init, Set}={width, height}] , DOUBLE={Get, Init, Set}={0 | 1} , LOCATION={Get, Init, Set}={x, y} , PROJECTION={Get, Init, Set}={1 | 2} ] , REGISTER_PROPERTIES={Get, Init, Set}] , .TRANSPARENT={Get, Init, Set}] , UNITS={Get, Init, Set}={0 | 1 | 2 | 3}] , VIEWPLANE_RECT={Get, Init, Set}={x, y, width, height}] , zCLIP={Get, Init, Set}={near, far}

IDLgrView::Add - Adds a child to this view.
Obj->IDLgrView::Add(Model [, POSITION=index]

IDLgrView::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrView::Cleanup

IDLgrView::GetByIndex - Finds contained objects by name.
Result = Obj->IDLgrView::GetByName(Name)

IDLgrView::GetProperty - Retrieves the value of a property or group of properties for the view.
Obj->IDLgrView::GetProperty() [ , PROPERTY=variable]

IDLgrView::Init - Initializes the view object.
Obj = OBJ_NEW(IDLgrView[, PROPERTY=value]) or
Result = Obj->IDLgrView::Init() [ , PROPERTY=value]

IDLgrView::SetProperty - Sets the value of a property or group of properties for the view.
Obj->IDLgrView::SetProperty(), PROPERTY=variable

IDLgrViewgroup - A simple container object that contains one or more IDLgrView objects. An IDLScene can contain one or more of these objects.
Properties: [ , ALL={Get=variable} [ , HIDE={Get, Init, Set}] , PARENT={Get=variable} [ , REGISTER_PROPERTIES={Get, Init, Set}] ]

IDLgrViewgroup::Add - Verifies that the added item is not an instance of the IDLScene or IDLgrViewgroup object.
Obj->IDLgrViewgroup::Add(Object [, POSITION=index]

IDLgrViewgroup::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrViewgroup::Cleanup

IDLgrViewgroup::GetByName - Finds contained objects by name.
Result = Obj->IDLgrViewgroup::GetByName(Name)

IDLgrVolume - Represents mapping from a 3D array of data to a 3D array of voxel colors, which, when drawn, are projected to two dimensions.
Properties: [ , ALL={Get=variable} [ , ALPHA_CHANNEL={Get, Init, Set}={value} [ , AMBIENT={Get, Init, Set}={RGB vector} [ , BOUNDS={Get, Init, Set}={xmin, ymin, zmin, xmax, ymax, zmax}] , CLIP_PLANES={Get, Init, Set}={array} [ , COMPOSITE_FUNCTION={Get, Init, Set}={0 | 1 | 2 | 3}] , DATA0={Get, Init, Set}={d0, d0, d0} , DATA1={Get, Init, Set}={d0, d0, d0} , DATA2={Get, Init, Set}={d0, d0, d0} , DATA3={Get, Init, Set}={d0, d0, d0} [ , DEPTH_CUE={Get, Init, Set}={brightness, zdim}]

IDLgrText Properties

IDL Quick Reference
IDLgrVolume::PickVoxel - Computes the coordinates of the voxel projected to a location specified by the 2D device coordinates point, (x, y), and the current Z-buffer.

Result = Obj->IDLgrVolume::PickVoxel ( Win, View, Point [, PATH=objref(s) ] )

IDLgrVolume::SetProperty - Sets the value of a property or group of properties for the volume.

Obj->IDLgrVolume::SetProperty[. PROPERTY=value]

IDLgrVMRL::SetProperty - Saves the contents of an Object Graphics hierarchy into a VRML 2.0 format file.

Properties: [], ALL{Get, Set}=[/variable]

[. SCREEN_DIMENSIONS{Get}=[variable]
[. COLOR_MODEL{Get, Init}=[0 | 1]
[. DIMENSIONS{Get, Init, Set}=[/width, height]]
[. FILENAME{Get, Init, Set}=[string]
[. GRAPHICS_TREE{Get, Init, Set}=[objref]
[. N_COLORS{Get, Init}=[integer][2 to 256]]
[. PALETTE{Get, Init, Set}=[objref]]
[. QUALITY{Get, Init, Set}=[0 | 1 | 2]]
[. REGISTER_PROPERTIES{Get, Init, Set}=[/RESOLUTION{Get, Init, Set}=[/xres, yres]]
[. UNITS{Get, Init, Set}=[0 | 1 | 2 | 3]]
[. WORLDINFO{Get, Init}=[string array]
[. WORLDTITLE{Init}=[string]

IDLgrVRML::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLgrVRML::Cleanup

IDLgrVRML::Draw - Draws a picture to this graphics destination.

Obj->IDLgrVRML::Draw[. Picture]

IDLgrVRML::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.

Obj->IDLgrVRML::GetDeviceInfo[. ALL=string]

[. MAX_NUM_CLIP_PLANES=variable]
[. MAX_TEXTURE_DIMENSIONS=variable]
[. MAX_VIEWPORT_DIMENSIONS=variable]
[. NAME=variable]
[. NUM_CPUS=variable]
[. VENDOR=variable]
[. VERSION=variable]

IDLgrVRML::GetFontnames - Returns the list of available fonts that can be used in IDLgrFont objects.

Return = Obj->IDLgrVRML::GetFontnames

( FamilyName [, IDL_FONTs=[0 | 1 | 2]]
[. STYLES=string ]
)

IDLgrVRML::GetProperty - Retrieves the value of a property or group of properties for the VRML object.

Obj->IDLgrVRML::GetProperty

[. PROPERTY=variable]

IDLgrVRML::GetTextDimensions - Retrieves the dimensions of a text object that will be rendered in the clipboard buffer.

Result = Obj->IDLgrVRML::GetTextDimensions( TextObj [, DESCENT=variable] [, PATH=objref(s) ] )

IDLgrVRML::Init - Initializes the VRML object.

Obj = OBJ_NEW(IDLgrVRML[, PROPERTY=value]) or
Result = Obj->IDLgrVRML::Init([PROPERTY=value])

IDLgrVRML::SetProperty - Sets the value of a property or group of properties for the VRML world.

Obj->IDLgrVRML::SetProperty[. PROPERTY=value]
IDAgrWindow - Represents an on-screen area on a display device that serves as a graphics destination.

Properties: [ , ALL{Get}=variable ]
[ , COLOR{MODEL}{Get, Init}={0 | 1} ]
[ , DIMENSIONS{Get, Init, Set}=[width, height] ]
[ , DISPLAY{NAME}{Get}{, X Windows Only}=variable ]
[ , IMAGE{DATA}{Get}=variable ]
[ , GRAPHICS{TREE}{Get, Init, Set}=objref of type IDLgrScene, IDLgrViewgroup, or IDLgrView ]
[ , LOCATION{Get, Init, Set}=[x, y] ]
[ , MINIMUM{VIRTUAL_DIMENSIONS}{Get, Init, Set}=[width, height] ]
[ , NColors{Get, Init, Set}={0 | 1} ]
[ , RENDERER{Get, Init}={0 | 1} ]
[ , RESOLUTION{Get}=[variable ]]
[ , TITLE{Get, Init, Set}=string ]
[ , UNITS{Get, Init, Set}={0 | 1 | 2 | 3} ]
[ , VIRTUAL_DIMENSIONS{Get, Init, Set}=[width, height] ]
[ , VISIBLE{LOCATION}{Get}=[x, y] ]
[ , ZBUFFER{DATA}{Get}=variable ]
[ , ZOOM{BASE}{Get, Init, Set}=value ]
[ , ZOOM{NSTEP}{Get, Init, Set}=variable ]
[ , ZOOM{SCALE}{Get, Init, Set}=variable ]
[ , ZOOM{TRANSFORM}{Get, Init, Set}=variable ]

IDLgrWindow::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrWindow:::Cleanup

IDLgrWindow::Draw - Draws the specified scene or view object to this graphics destination.
Obj->IDLgrWindow:::Draw [ , {Picture} ]
[ , CREATE{INSTANCE}={1 | 2} ]
[ , /DRAW{INSTANCE} ]

IDLgrWindow::Erase - Erases the entire contents of the window.
Obj->IDLgrWindow:::Erase [ , COLOR=index or RGB vector ]

IDLgrWindow::GetContiguousPixels - Returns an array of long integers whose length is equal to the number of colors available in the index color mode (value of N_COLORS property).
Return = Obj->IDLgrWindow:::GetContiguousPixels()

IDLgrWindow::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.
Obj->IDLgrWindow:::GetDeviceInfo [ , ALL=variable ]
[ , MAX{NUM_CLIP_PLANES}=variable ]
[ , MAX{TEXTURE_DIMENSIONS}=variable ]
[ , MAX{VIEWPORT_DIMENSIONS}=variable ]
[ , NAME=variable ]
[ , NUM{CPU{S}}=variable ]
[ , VENDOR=variable ]
[ , VERSION=variable ]

IDLgrWindow::GetDimensions - Returns a two-element vector, [width, height], representing the visible dimensions (in device units) of this window.
Result = Obj->IDLgrWindow:::GetDimensions [ , MINIMUM{VIRTUAL_DIMENSIONS}=variable ]
[ , ORIGINAL{VIRTUAL_DIMENSIONS}=variable ]
[ , VIRTUAL_DIMENSIONS=variable ]
[ , VISIBLE{LOCATION}=variable ]

IDLgrWindow::GetFontNames - Returns the list of available fonts that can be used in IDLgrFont objects.
Return = Obj->IDLgrWindow:::GetFontNames(FamilyName [
, IDL_FONTS={0 | 1 | 2} ]
, STYLES={string} )

IDLgrWindow::GetProperty - Retrieves the value of a property or group of properties for the window.
Obj->IDLgrWindow:::GetProperty [ , PROPERTY=variable ]

IDLgrWindow::GetTextDimensions - Retrieves the dimensions of a text object that will be rendered in the window.
Result = Obj->IDLgrWindow:::GetTextDimensions( TextObj [
, DESCENT=variable ]
, PATH=objref(s) )

IDLgrWindow::Iconify - Iconifies or de-iconifies the window.
Obj->IDLgrWindow:::Iconify, IconFlag

IDLgrWindow::Init - Initializes the window object.
Obj = OBJ_NEW(’IDLgrWindow’[,
, PROPERTY=variable ]
) or Result = Obj->IDLgrWindow:::Init( [ , PROPERTY=variable ]

IDLgrWindow::PickData - Maps a point in the 2D device space of the window to a point in the 3D data space of an object tree.
Result = Obj->IDLgrWindow:::PickData(View, Object,
Location, XYZLocation [,
, DIMENSIONS=[width, height] ]
, PATH=objref(s) ]
, PICK{STATUS}=variable ]

IDLgrWindow::Read - Reads an image from a window.
Result = Obj->IDLgrWindow:::Read()

IDLgrWindow::Select - Returns a list of objects selected at a specified location.
Result = Obj->IDLgrWindow:::Select Picture, XY [,
, DIMENSIONS=[width, height] ]
, /ORDER ]
[ , SUB_SELECTION=variable ]
, UNITS={0 | 1 | 2 | 3} ]

IDLgrWindow::SetCurrentCursor - Sets the current cursor image to be used while positioned over a drawing area.
Obj->IDLgrWindow:::SetCurrentCursor( CursorName [
, IMAGE=16 x 16 bitmap ]
, MASK=16 x 16 bitmap ]
, HOTSPOT=[x, y] )
X Windows Only Keywords: [ , STANDARD=index ]

IDLgrWindow::SetCurrentZoom - Sets the current zoom factor for this window. The current zoom factor, the virtual canvas dimensions, and the visible dimensions of the window are updated to reflect the new zoom factor.
Obj->IDLgrWindow:::SetCurrentZoom, ZoomFactor [ , /RESET ]
IDLQuickReference

IDLitCommand - The base functionality for the iTools command buffer system.


IDLitCommand::AddItem - Adds the specified data item to the data dictionary associated with this object.

Result = Obj->IDLitCommand::AddItem(StringItem, Item [, /OVERWRITE])

IDLitCommand::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLitCommand::Cleanup()

IDLitCommand::GetItem - Retrieves the specified item from the data dictionary associated with this object.

Result = Obj->IDLitCommand::GetItem(StringItem, Item)

IDLitCommand::GetProperty - Retrieves the value of a property or group of properties of a command object.

Obj->IDLitCommand::GetProperty [ , PROPERTY=variable]

IDLitCommand::GetSize - Returns an approximate value for the amount of memory being used by the items in the data dictionary associated with this object.

Result = Obj->IDLitCommand::GetSize( [, /KILOBYTES])

IDLitCommand::Init - Initializes the object and allows specification of items associated with it.

Obj = OBJ_NEW('IDLitCommand' [, PROPERTY=value]) or

Result = Obj->IDLitCommand::Init( [PROPERTY=value])

IDLitCommand::SetProperty - Sets the value of a property or group of properties for the command object.

Obj->IDLitCommand::SetProperty [ , PROPERTY=value]

IDLitCommandSet - A container for IDLitCommand objects, which allows a group of commands to be managed as a single item.

IDLitCommandSet::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLitCommandSet::Cleanup()

IDLitCommandSet::GetSize - Returns an approximate value for the amount of memory being used by the items contained by this command set.

Result = Obj->IDLitCommandSet::GetSize( [, /KILOBYTES])

IDLitCommandSet::Init - Initializes the object.

Obj = OBJ_NEW('IDLitCommandSet') or

Result = Obj->IDLitCommandSet::Init()

IDLitComponent - A core or base component, from which all other components subclass.


IDLitComponent::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLitComponent::Cleanup()

IDLitComponent::EditUserDefProperty - Defines the interface that is displayed when a user selects the "Edit" button on a user-defined property in the property sheet interface.

Result = Obj->IDLitComponent::EditUserDefProperty( iTool, PropertyIdentifier)

IDLitComponent::GetFullIdentifier - Navigates the iTool object container hierarchy of the object on which it is called and retrieves the fully-qualified object identifier.

Result = Obj->IDLitComponent::GetFullIdentifier( [Objref])

IDLitComponent::GetProperty - Retrieves the value of an IDLitComponent property or properties.

Obj->IDLitComponent::GetProperty [ , PROPERTY=variable]

IDLitComponent::GetPropertyAttribute - Retrieves property attribute values for a registered property.

Obj->IDLitComponent::GetPropertyAttribute, PropertyIdentifier [, TYPE=variable]

IDLitComponent::GetPropertyByIdentifier - Retrieves the value of an IDLitComponent property.

Result = Obj->IDLitComponent::GetPropertyByIdentifier( PropertyIdentifier, Value)
IDL Component::AddByIdentifier - Adds an object to the container hierarchy using the specified identifier to locate the object.

Result = Obj->IDLComponent::AddByIdentifier( Identifier )

IDL Component::Add - Adds items to the container object.

Result = Obj->IDLComponent::Add( Components )

IDL Component::CleanUp - Performs any transitional work required after an object has been restored from a SAVE file.

Result = Obj->IDLComponent::CleanUp()

IDL Component::Get - Retrieves items from the container.

Result = Obj->IDLComponent::Get( [ , /NO_NOTIFY ]
[ , COUNT=variable ] [ , /ALL ]
[ , ISA=string ]
[ , POSITIONS=index or array of indices ]
[ , /SKIP_PRIVATE ]
)

IDL Component::Init - Initializes the object.

Obj = OBJ_NEW( 'IDLitComponent' )

IDL Component::QueryProperty - Checks whether a property identifier is registered, or retrieves a list of all registered properties.

Result = Obj->IDLComponent::QueryProperty( [PropertyIdentifier] )

IDL Component::RegisterProperty - Registers a property as belonging to the component.

Org->IDLComponent::RegisterProperty( PropertyIdentifier [, Type] [ , /BOOLEAN ] [ , /COLOR ]
[ , DESCRIPTIOn=strstring ], ENUMLIST=stringvector ]
[ , /FLOAT ] [ , /HIDE ] [ , /INTEGER ] [ , /LINESTYLE ]
[ , NAME=strstring ] [ , /SENSITIVE ] [ , /STRING ]
[ , /SYMBOL ] [ , /THICKNESS ] [ , /UNDEFINED ]
[ , USERDEF=strstring ] [ , VALID_RANGE=vector ]

IDL Component::RemoveByIdentifier - Removes an object from a container hierarchy using the specified identifier to locate the object.

Result = Obj->IDLComponent::RemoveByIdentifier( Identifier )

IDL Component::Remove - Removes items from the container.

Result = Obj->IDLComponent::Remove( Components )
[ , /NO_NOTIFY ]

IDL Component::GetByComponentVersion - Updates the value of the COMPONENT_VERSION property for the specified object to the current release of IDL.

Result = Obj->IDLComponent::GetByComponentVersion()

IDL Container - A specialization of the IDL Container class that manages a collection of IDLitComponents and provides methods for working with the Identifier system of the iTools framework.

IDL Container::GetByIdentifier - Retrieves an object from a container hierarchy using the specified identifier.

Result = Obj->IDLContainer::GetByIdentifier( Identifier )

IDL Container::Get - Retrieves items from a container hierarchy.

Result = Obj->IDLContainer::Get( [ , /NO_NOTIFY ]
[ , COUNT=variable ] [ , /ALL ]
[ , ISA=string ]
[ , POSITIONS=index or array of indices ]
[ , /SHORT_NAMES ]
[ , /SKIP_PRIVATE ]
)

IDL Container::GetByType - Returns all contained objects of the specified iTool data type.

Result = Obj->IDLContainer::GetByType( Type )
[ , COUNT=variable ]
[ , /NO_NOTIFY ]

IDL Container::Init - Initializes the IDLitContainer object.

Obj = OBJ_NEW( 'IDLitContainer' )

IDL Container::RemoveByIdentifier - Removes an object from the container hierarchy using the specified identifier.

Result = Obj->IDLContainer::RemoveByIdentifier( Identifier )

IDL Container::Remove - Removes items from a container.

Result = Obj->IDLContainer::Remove( Components )
[ , /NO_NOTIFY ]

IDL Data - A generic data storage object that can hold any IDL data type available. It provides typing, metadata, and data change notification functionality. When coupled with IDLitDataContainer, it forms the element for the construction of composite data types.

Properties:
[ , /HIDE{Get, Init, Set} ]
[ , /NO_COPY{Get, Init, Set} ]
[ , READ_ONLY{Get, Init} ]
[ , TYPE{Get, Init} ]
[ , SKIP_PRIVATE ]

IDL Data::AddDataObserver - Specifies an object (the Observer) that will be notified when the contents of the data object are changed.

Obj->IDLData::AddDataObserver( Observer )

IDL Data::Cleanup - Performs all cleanup operations on the object.

OBJ_DESTROY, Obj or Obj->IDLData::Cleanup()

IDL Data::Copy - Returns an exact copy of the data object and its contents, including registered property values.

Result = Obj->IDLData::Copy()

IDL Data::GetByType - Returns all contained objects of the specified iTool data type.

Result = Obj->IDLData::GetByType( Type )
[ , COUNT=variable ]

IDL Data::GetData - Retrieves the data stored in the object.

Result = Obj->IDLData::GetData( [Data], Identifier )
[ , NAN=variable ]
[ , /NO_COPY ]

IDL Data::GetProperty - Retrieves the value of an IDLitData property.

Result = Obj->IDLData::GetProperty( PROPERTY=variable )

IDL Component::Add - Adds items to the container object.

Result = Obj->IDLComponent::Add( Components )
[ , /NO_NOTIFY ]

IDL Component::CleanUp - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLContainer::CleanUp

IDL Component::GetByIdentifier - Retrieves an object from a container hierarchy using the specified identifier.

Result = Obj->IDLComponent::GetByIdentifier( Identifier )

IDL Component::Init - Initializes the IDLitComponent object.

Obj = OBJ_NEW( 'IDLitComponent' )
[ , PROPERTY=value ]
or
Result = Obj->IDLComponent::Init( [ , PROPERTY=value ]
)

IDL Component::QueryProperty - Checks whether a property identifier is registered, or retrieves a list of all registered properties.

Result = Obj->IDLComponent::QueryProperty( [PropertyIdentifier] )

IDL Component::RegisterProperty - Registers a property as belonging to the component.

Org->IDLComponent::RegisterProperty( PropertyIdentifier [, Type] [ , /BOOLEAN ] [ , /COLOR ]
[ , DESCRIPTIOn=strstring ], ENUMLIST=stringvector ]
[ , /FLOAT ] [ , /HIDE ] [ , /INTEGER ] [ , /LINESTYLE ]
[ , NAME=strstring ] [ , /SENSITIVE ] [ , /STRING ]
[ , /SYMBOL ] [ , /THICKNESS ] [ , /UNDEFINED ]
[ , USERDEF=strstring ] [ , VALID_RANGE=vector ]
A container for IDLitData and IDLitDataContainer objects. This container is used to form hierarchical data structures. Data and DataContainer objects can be added and removed to/from a DataContainer during program execution, allowing for dynamic creation of intermediate data types.

IDLlitDataContainer::Add - Adds items to the data container object.

IDLlitDataContainer::Cleanup - Performs all cleanup operations on the object.

IDLlitDataContainer::GetData - Retrieves the data value contained in the data object specified by the Identifier argument.

IDLlitDataContainer::GetIdentifiers - Retrieves the object identifiers for all data and data container objects contained in the data container object.

IDLlitDataOperator::GetProperty - Retrieves the value of an IDLitDataOperator property.

IDLlitDataOperator::Init - Initializes the IDLitDataOperator object.

IDLlitDataOperator::DoExecuteUI - Provides a way for the iTool developer to request user input before performing an operation.

IDLlitDataOperator::Execute - Contains the execution logic for the operation.

IDLlitDataOperator::GetProperty - Retrieves the value of a property or group of properties of an operation object.

IDLlitDataOperator::UndoExecute - Is called when a user selects the Undo operation after executing an IDLitDataOperation that sets the value of the REVERSIBLE_OPERATION property to 1.

IDLQuickReference

IDLlitDataOperation::UndoExecute
IDLitIMessaging - An interface providing common methods to send or trigger messaging and error actions, which may occur during execution.

IDLitIMessaging::AddOnNotifyObserver - Is used to register a specified iTool component object to receive messages generated by the DoOnNotify method of another specified iTool component object.

IDLitIMessaging::DoOnNotify - Is used to broadcast a notification message to iTool component objects that are observing the source of the message.

IDLitIMessaging::DoOnNotify, IdOriginator, StrMessage

IDLitIMessaging::ErrorMessge - Is used to display an error message to the user.

IDLitIMessaging::ErrorMessage, StrMessage

IDLitIMessaging::GetTool - Returns an object reference to the iTool object associated with the object on which it is called.

IDLitIMessaging::ProbeStatusMessage - Is used to display a status message to the user, which is displayed in a data-specific region of the user interface.

IDLitIMessaging::ProgressBar - Is used to display a progress bar to the user and update the displayed values.

IDLitIMessaging::PromptUserText - Is used to prompt the iTool user with a question and retrieve a text answer.

IDLitIMessaging::PromptUserYesNo - Is used to prompt the user with a yes or no question and return an answer.

IDLitIMessaging::RemoveOnNotifyObserver - Is used to unregister a specified iTool component object as wishing to receive messages generated by the DoOnNotify method of another specified iTool component object.

IDLitIMessaging::SignalError - Is used to signal an error in the system.

IDLitIMessaging::StatusMessage - Is used to display a status message to the user.

IDLitManipulator - The base functionality of the iTools manipulator system.


IDLitManipulator::Cleanup - Performs all cleanup on the object.

OBJ_Destroy, Obj or Obj->[IDLitManipulator::]Cleanup()

IDLitManipulator::CommitUndoValues - Is used to complete a transaction that is occurring as a result of the manipulator interaction.

Result = Obj->[IDLitManipulator::]CommitUndoValues( ![UNCOMMENT])

IDLitManipulator::GetCursorType - Retrieves the name of the cursor to display for the manipulator.

Result = Obj->[IDLitManipulator::]GetCursorType( TypeIn, KeyMods)

IDLitManipulator::GetProperty - Retrieves the value of an IDLManipulator property

Obj->[IDLitManipulator::]GetProperty [.PROPERTY=value]

IDLitManipulator::Init - Initializes the IDLManipulator object

Obj = OBJ_NEW([IDLitManipulator] [.PROPERTY=value]) or Result = Obj->[IDLitManipulator::]Init( [.PROPERTY=value])

IDLitManipulator::OnKeyboard - Is used when a keyboard event occurs on the target window (Win).

Obj->[IDLitManipulator::]OnKeyboard, Win, IsASCII, Character, Key Value, X, Y, Press, Release, KeyMods

IDLitManipulator::OnLoseCurrentManipulator - Is used when this manipulator is no longer the current manipulator in the system.

Obj->[IDLitManipulator::]OnLoseCurrentManipulator

IDLitManipulator::OnMouseDown - Is used when a mouse down event occurs on the target window.

Obj->[IDLitManipulator::]OnMouseDown, Win, X, Y, IButton, KeyMods, NCClicks

IDLitManipulator::OnMouseMotion - Manages the setting of the cursor on the window if no mouse button is down.

Obj->[IDLitManipulator::]OnMouseMotion, Win, X, Y, KeyMods
**Alphabetical List of IDL Routines**

**IDLitManipulatorContainer**

- **OnMouseUp** - Is used when a mouse up event occurs on the target window.
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::OnMouseUp, Win, X, Y, IButton
  ```

- **RecordUndoValues** - Is used to begin recording the transaction that is occurring as a result of the manipulator interaction. This method works in conjunction with the CommitUndoValues method.
  
  ```idl```
  ```
  Result = Obj->IDLitManipulator::RecordUndoValues()
  ```

- **RegisterCursor** - Defines the appearance and name of the cursor associated with the manipulator.
  
  ```idl```
  ```
  Obj->IDLitManipulator::RegisterCursor, ArvCursor, Name [, /DEFAULT]
  ```

- **SetCurrentManipulator** - Sets the manipulator as the current manipulator in the system.
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::SetCurrentManipulator
  ```

- **SetProperty** - Sets the value of an IDLitManipulator property
  
  ```idl```
  ```
  Obj->IDLitManipulator::SetProperty [PROPERTY=value]
  ```

- **GetCurrent** - Is used to get the object reference of the current manipulator of the container.
  
  ```idl```
  ```
  Result = Obj->IDLitManipulatorContainer::GetCurrent()
  ```

- **GetCurrentManipulator** - Is used to get the current manipulator of the system.
  
  ```idl```
  ```
  Result = Obj->IDLitManipulatorContainer::GetCurrentManipulator()
  ```

- **GetProperty** - Retrieves the value of an IDLitManipulatorContainer property
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::GetProperty [PROPERTY=variable]
  ```

- **Init** - Initializes the IDLitManipulatorContainer object.
  
  ```idl```
  ```
  Obj = OBJ_NEW('IDLitManipulatorContainer'
  [PROPERTY=value]) or
  Result = Obj->IDLitManipulatorContainer::Init([PROPERTY=value])
  ```

- **OnKeyboard** - Is used when a keyboard event occurs on the target window (Win).
  
  ```idl```
  ```
  Obj->IDLitManipulator::OnKeyboard, Win, IsASCII, Character, KeyVal, X, Y, Press, Release, KeyMods
  ```

- **OnMouseDown** - Is used when a mouse down event occurs on the target window.
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::OnMouseDown, Win, X, Y, IButton, KeyMods, NClicks
  ```

**IDLitManipulatorContainer**

- **OnMouseMotion** - Manages the setting of the cursor on the window if no mouse button is down.
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::OnMouseMotion, Win, X, Y, KeyMods
  ```

- **OnMouseUp** - Is used when a mouse up event occurs on the target window.
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::OnMouseUp, Win, X, Y, IButton
  ```

- **SetCurrent** - Is used to set a manipulator within the container to be the current manipulator.
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::SetCurrent, Manipulator
  ```

- **SetCurrentManipulator** - Is used to set the current child manipulator in a manipulator hierarchy.
  
  ```idl```
  ```
  Obj->IDLitManipulatorContainer::SetCurrentManipulator, Identifier
  ```

- **AddManipulatorObserver** - Is used to add an observer to the manipulator system.
  
  ```idl```
  ```
  Obj->IDLitManipulatorManager::AddManipulatorObserver, Observer
  ```

- **GetDefaultManipulator** - Returns a reference to the manipulator that was most recently added as the default manipulator.
  
  ```idl```
  ```
  Obj->IDLitManipulatorManager::GetDefaultManipulator()
  ```

- **Init** - Initializes the IDLitManipulatorManager object
  
  ```idl```
  ```
  Obj = OBJ_NEW('IDLitManipulatorManager'
  [PROPERTY=value]) or
  Result = Obj->IDLitManipulatorManager::Init([PROPERTY=value])
  ```

- **RemoveManipulatorObserver** - Is used to remove a manipulator observer from the manipulator object.
  
  ```idl```
  ```
  Obj->IDLitManipulatorManager::RemoveManipulatorObserver, Observer
  ```

**IDLitManipulatorVisual** - The basis of all visual elements associated with an interactive manipulator.

- **Properties**: [ /UNIFORM_SCALE] [ /VISUAL_TYPE=string]
IDLManipulatorVisual::Cleanup - Performs all cleanup on the object.

IDLManipulatorVisual::GetProperty - Retrieves the value of an IDLManipulatorVisual property.

IDLManipulatorVisual::Init - Initializes the IDLManipulatorVisual object.

IDLManipulatorVisual::setProperty - Sets the value of an IDLManipulatorVisual property.

IDLOperation - The basis for all iTool operations. It defines how an operation is executed and how information about the operation is recorded for the command transaction (undo-redo) system.

IDLParameter - An interface providing parameter management methods to associate parameter names with IDLitData objects.

IDLParameter::Cleanup - Removes the data observer from each data object in the visualization object’s parameter set, and cleans up the objects and pointers defined to hold parameter data when the visualization object was created.

IDLParameter::GetParameterSet - Returns a reference to the IDLitParameterSet object associated with the visualization object.

IDLParameter::Init - Initializes object instance fields that contain parameter data.

IDLParameter::OnDataDisconnect - Is called when a data value has been disconnected from a parameter.

IDLParameter::OnDataChangeUpdate - Is called when a data value has been updated or a new data object has been associated with the visualization object.

IDLParameter::QueryParameter - Checks whether a parameter is registered, or retrieves a list of the of all registered parameters.

IDLOperation::RedoOperation - Is called by the iTool system when the user requests the re-execution of an operation (usually by selecting Redo from the iTool Edit menu or toolbar).

IDLOperation::UndoOperation - Is called by the iTool system when the user requests the un-execution of an operation (usually by selecting Undo from the iTool Edit menu or toolbar).

IDLParameter::GetParameterSet - Returns a reference to the IDLitParameterSet object associated with a registered parameter.

IDLParameter::GetParameterAttribute - Retrieves the value of an attribute associated with a registered parameter.

IDLParameter::GetParameter - Retrieves the IDLitData object associated with a registered parameter.

IDLParameter::SetProperty - Sets the value of a property or group of properties for the operation.

IDLParameter::GetProperty - Retrieves the IDLitData object associated with the visualization object.

IDLParameter::OnDataChangeUpdate - Is called when a data value has been updated or a new data object has been associated with the visualization object.

IDLParameter::QueryParameter - Checks whether a parameter is registered, or retrieves a list of the of all registered parameters.

IDLParameter::DoAction - Is called when an operation is requested by the iTool system, either as the result of a user action such as selection of a menu item or toolbar button, or by another operation.

IDLParameter::OnDataDisconnect - Is called when a data value has been disconnected from a parameter.

IDLParameter::OnDataChangeUpdate - Is called when a data value has been updated or a new data object has been associated with the visualization object.

IDLParameter::QueryParameter - Checks whether a parameter is registered, or retrieves a list of the of all registered parameters.
IDL litParameter::RegisterParameter - Registers a parameter with the visualization object.

IDL litParameter::SetData - Is used to set data in this interface, associating a data object with a given parameter.
Result = Obj->IDLitParameter::SetData(Data [, BY_VALUE] [, PARAMETER_NAME=string] [, NO_UPDATE])

IDL litParameter::SetParameterAttribute - Sets one or more parameter attributes for a registered parameter.
Obj->IDLitParameter::SetParameterAttribute, ParamName [, ATTRIBUTE=variable]

IDL litParameterSet::SetParameterSet - Is used to associate an IDLitParameterSet object with the visualization object’s parameter interface.
Result = Obj->IDLitParameterSet::SetParameterSet(ParamSet)

IDL litParameterSet - A specialized subclass of the IDLitDataContainer class. This class provides the ability to associate names with contained IDLitData objects.

IDL litParameterSet::Add - Is used to add data to the parameter set.
Obj->IDLitParameterSet::Add, Data [, PARAMETER_NAME=string] [, PRESERVE_LOCATION]

IDL litParameterSet::Cleanup - Performs all cleanup on the parameter set object.
OBJ_DESTROY, Obj or Obj->IDLitParameterSet::Cleanup()

IDL litParameterSet::Copy - Returns a copy of the parameter set and its contents.
Result = Obj->IDLitParameterSet::Copy()

IDL litParameterSet::Get - Is used to retrieve one or more IDLitData objects from the parameter set.
Result = Obj->IDLitParameterSet::Get(), /ALL [, COUNT=variable] [, NAME=variable] [, POSITION=integer]

IDL litParameterSet::GetByIndex - Returns the IDLitData object associated with a specified named parameter.
Result = Obj->IDLitParameterSet::GetByIndex(Names [, COUNT=variable] [, NAME=variable])

IDL litParameterSet::GetParameterName - Retrieves the name of a specified parameter using a provided data object.
Result = Obj->IDLitParameterSet::GetParameterName(Data, Name)

IDL litParameterSet::Init - Initializes the IDLitParameter object.
Obj = OBJ_NEW('IDLitParameterSet') or Result = Obj->IDLitParameterSet::Init()

IDL litParameterSet::Remove - Is used to remove a data item from the parameter set.
Obj->IDLitParameterSet::Remove(), Items [, /ALL] [, POSITION=integer]

IDL litReader - The definition of the interface and the process used to construct file readers for the iTools framework. When a new file reader is constructed for the iTools system, a new class is subclassed from this IDLitReader class.

IDL litReader::Cleanup - Performs all cleanup on the object, and should be called by the subclass’ Cleanup method.
OBJ_DESTROY, Obj or Obj->IDLitReader::Cleanup

IDL litReader::GetData - Is called by the system to retrieve the data from the current file.
Result = Obj->IDLitReader::GetData(Data)

IDL litReader::GetFileExtensions - Is called by the system to retrieve the file extensions supported by this particular reader.
Result = Obj->IDLitReader::GetFileExtensions( COUNT=variable)

IDL litReader::GetFilename - Is called by the system to retrieve the current filename associated with this reader.
Result = Obj->IDLitReader::GetFilename()

IDL litReader::GetProperty - Retrieves the value of an IDLitReader property.
Obj->IDLitReader::GetProperty [, PROPERTY=variable]

IDL litReader::Init - Initializes the IDLitReader object.
Obj = OBJ_NEW('IDLitReader', Extensions [, PROPERTY=value]) or Result = Obj->IDLitReader::Init(Extensions [, PROPERTY=value])

IDL litReader::IsA - Is called by the system to determine if the given file is of the type supported by this file reader.
Result = Obj->IDLitReader::IsA(Filename)

IDL litReader::SetFilename - Is called by the system to set the current filename associated with this reader.
Obj->IDLitReader::SetFilename, Filename

IDL litReader::setProperty - Sets the value of an IDLitReader property.
Obj->IDLitReader::setProperty[, PROPERTY=value]

IDL litTool - All the functionality provided by a particular instance of an IDL Intelligent Tool (iTool). This object provides the management systems for the underlying tool functionality.

IDL litTool::ActivateManipulator - Activates a manipulator that has been registered with this tool.
Obj->ActivateManipulator, Identifier[, /DEFAULT]

IDL litTool::Add - Adds any item to the tool.
Obj->IDLitTool::Add, Item
IDLitTool::AddService - Adds a service to the tool.
Obj->[IDLitTool::AddService, Service

IDLitTool::Cleanup - Performs all cleanup on the object
OBJ_DESTROY, Obj or Obj->[IDLitTool::Cleanup

IDLitTool::CommitActions - Commits all pending transactions to the undo-redo buffer and causes a refresh of the current window.
Obj->[IDLitTool::CommitActions

IDLitTool::DisableUpdates - Disables all drawing updates to the current window and UI updates (menu sensitivity) being passed to the user interface.
Obj->[IDLitTool::DisableUpdates ([PREVIOUSLY_DISABLED=variable])

IDLitTool::DoAction - Initiates an operation or action in the tool object.
Result = Obj->[IDLitTool::DoAction(Identifier)

IDLitTool::DoSetProperty - Sets a property on a target component object, and places the change in the undo/redo transaction buffer.
Result = Obj->[IDLitTool::DoSetProperty( TargetIdentifier, PropertyIdentifier, Value)

IDLitTool::DoUIService - Initiates a request for a UI service to execute.
Result = Obj->[IDLitTool::DoUIService( ServiceIdentifier, Requestor)

IDLitTool::EnableUpdates - Re-enables all drawing updates to the current window and UI updates (menu sensitivity) being passed to the user interface.
Obj->[IDLitTool::EnableUpdates

IDLitTool::FindIdentifiers - Retrieve the full identifiers for items within the tool container

IDLitTool::GetCurrentManipulator - Returns the current manipulator in the system.
Result = Obj->[IDLitTool::GetCurrentManipulator()

IDLitTool::GetFileReader - Retrieves a file reader registered with the tool object.
Result = Obj->[IDLitTool::GetFileReaders([Identifier [.ALL] [.COUNTr=variable]])

IDLitTool::GetFileWriter - Retrieves a file writer registered with the tool object.
Result = Obj->[IDLitTool::GetFileWriters([Identifier [.ALL] [.COUNTr=variable]])

IDLitTool::GetManipulators - Retrieves the manipulators registered with the tool object.
Result = Obj->[IDLitTool::GetManipulators([COUNTr=variable])

IDLitTool::GetOperations - Retrieves the operations registered with the tool object.
Result = Obj->[IDLitTool::GetOperations([IDENTIFIER=string] [.COUNTr=variable])

IDLitTool::GetProperty - Retrieves the value of an IDLitTool property.
Obj->[IDLitTool::GetProperty, [PROPERTY=variable]

IDLitTool::GetSelectedItems - Returns a vector of references to the objects currently selected within the current window in the tool.
Result = Obj->[IDLitTool::GetSelectedItems([COUNTr=variable])

IDLitTool::GetService - Retrieves a service that has been registered with the tool.
Result = Obj->[IDLitTool::GetService(IdService)

IDLitTool::GetVisualization - Retrieves a visualization registered with the tool object.
Result = Obj->[IDLitTool::GetVisualizations(Identifier [.ALL] [.COUNTr=variable])

IDLitTool::Init - Initializes the IDLitTool object
Obj = OBJ_NEW(‘IDLitTool’[, /ALL] [.COUNTr=variable])

IDLitTool::RefreshCurrentWindow - Redraws the current window of the tool.
Obj->[IDLitTool::RefreshCurrentWindow

IDLitTool::Register - Registers a generic component with the tool.
Obj->[IDLitTool::Register, Name, ClassName [.DEFAULT] [.DESCRIPTION=string] [.ICON=string] [.IDENTIFIER=string] [.PROXY=string]

IDLitTool::RegisterCustomization - Registers an operation class that represents the graphics customization operation to be associated with this tool.
Obj->[IDLitTool::RegisterCustomization, Name, ClassName

IDLitTool::RegisterFileReader - Registers a file reader component with the tool.
Obj->[IDLitTool::RegisterFileReader, Name, ClassName [.DEFAULT] [.DESCRIPTION=string] [.ICON=string] [.IDENTIFIER=string] [.PROXY=string]

IDLitTool::RegisterFileWriter - Registers a file writer component with the tool.
Obj->[IDLitTool::RegisterFileWriter, Name, ClassName [.DEFAULT] [.DESCRIPTION=string] [.ICON=string] [.IDENTIFIER=string] [.PROXY=string]

IDLitTool::RegisterManipulator - Registers a manipulator component with the tool.
Obj->[IDLitTool::RegisterManipulator, Name, ClassName [.DEFAULT] [.DESCRIPTION=string] [.ICON=string] [.IDENTIFIER=string]
**IDLitTool::SetProperty** - Sets the value of an IDLitTool property

```idl
Obj->[IDLitTool::]SetPropertyPROPERTY=value
```

**IDLitTool::RegisterStatusBarSegment** - Registers a status message bar segment with the tool.

```idl
```

**IDLitTool::RegisterVisualization** - Registers a visualization component with the tool.

```idl
Obj->[IDLitTool::]RegisterVisualizationName, ClassName [, /DEFAULT] [, DESCRIPTION=string] [, ICON=string] [, IDENTIFIER=string] [, PROXY=string]
```

**IDLitTool::UnRegister** - Unregisters a component with the tool.

```idl
Obj->[IDLitTool::]UnRegisterPROPERTY=value
```

**IDLitTool::UnRegisterStatusBarSegment** - Unregisters a status message bar segment with the tool.

```idl
Obj->[IDLitTool::]UnRegisterStatusBarSegmentName
```

**IDLitTool::UnRegisterVisualization** - Unregisters a visualization component with the tool.

```idl
Obj->[IDLitTool::]UnRegisterVisualizationIdentifier
```

**IDLitTool::UnRegisterOperation** - Unregisters an operation component with the tool.

```idl
Obj->[IDLitTool::]UnRegisterOperationIdentifier
```

**IDLitTool::UnRegisterManipulator** - Unregisters a manipulator component with the tool.

```idl
Obj->[IDLitTool::]UnRegisterManipulatorIdentifier
```

**IDLitTool::UnRegisterFileWriter** - Unregisters a file writer component with the tool.

```idl
Obj->[IDLitTool::]UnRegisterFileWriterIdentifier
```

**IDLitTool::UnRegisterFileReader** - Unregisters a file reader component with the tool.

```idl
Obj->[IDLitTool::]UnRegisterFileReaderIdentifier
```

**IDLitTool::UnRegisterCustomization** - Unregisters a customization component with the tool.

```idl
Obj->[IDLitTool::]UnRegisterCustomizationIdentifier
```

**IDLitUI** - A link between the underlying functionality of an iTool and the IDL widget interface.

- **Property:** [ GROUP_LEADER=WidgetID]
  ```idl```
  ```idl```
  ```idl```

**IDLitUI::AddOnNotifyObserver** - Is used to register a specified iTool component object as wishing to receive messages generated by the DoOnNotify method of another specified iTool component object.

```idl
Obj->[IDLitUI::]AddOnNotifyObserver, IdObserver, IdSubject
```

**IDLitUI::Cleanup** - Performs all cleanup on the object

```idl
OBJ_DESTROY, Obj or Obj->[IDLitUI::]Cleanup
```

**IDLitUI::DoAction** - Initiates an operation or action in the tool object associated with this user interface.

```idl
Result = Obj->[IDLitUI::]doAction(Identifier)
```

**IDLitUI::GetProperty** - Retrieves the value of an IDLitUI property.

```idl
Obj->[IDLitUI::]GetProperty, [PROPERTY=variable]
```

**IDLitUI::GetTool** - Returns an object reference to the iTool with which the user interface is associated.

```idl
Result = Obj->[IDLitUI::]GetTool()
```

**IDLitUI::GetWidgetByComponent** - Returns the IDL Widget ID of a widget that has been registered with the user interface object via a call to the IDLitUI::RegisterWidget method.

```idl
Result = Obj->[IDLitUI::]GetWidgetComponent(Name)
```

**IDLitUI::Init** - Initializes the IDLitUI object

```idl
Obj = OBJ_NEW('IDLitUI') or Result = Obj->[IDLitUI::]Init()
```

**IDLitUI::RegisterUIService** - Registers a user interface service with the user interface.

```idl
Result = Obj->[IDLitUI::]RegisterUIServiceServiceIDName, Callback
```

**IDLitUI::RegisterWidget** - Registers an IDL widget hierarchy with the user interface object.

```idl
Result = Obj->[IDLitUI::]RegisterWidget(Name, Callback)
```

**IDLitUI::RemoveOnNotifyObserver** - Is used to un-register a specified iTool component object as wishing to receive messages generated by the DoOnNotify method of another specified iTool component object.

```idl
Obj->[IDLitUI::]RemoveOnNotifyObserver, IdObserver, IdSubject
```

**IDLitUI::SetProperty** - Sets the value of an IDLitUI property.

```idl
Obj->[IDLitUI::]SetPropertyPROPERTY=value
```

**IDLitUI::UnRegisterUIService** - Unregisters a user interface service with the user interface object.

```idl
Obj->[IDLitUI::]UnRegisterUIService
```

**IDLitUI::UnRegisterWidget** - Unregisters a widget with the user interface object.

```idl
Obj->[IDLitUI::]UnRegisterWidget
```
IDLitVisualization - The basis for all iTool visualizations. All visualization components subclass from this class.

Properties: 
- CENTER_OF_ROTATION{Get, Init, Set} = [x, y, z] or [x, y, z] [ /IMPACTS_RANGE{Get, Init, Set}] [ /ISOTROPIC{Get, Init, Set}]
- MANIPULATOR_TARGET{Get, Init, Set}
- PROPERTY_INTERSECTION{Get, Init, Set}
- TYPE{Init, Set} = string or string array

IDLitVisualization::Add - Adds objects to the visualization container.

Obj->IDLitVisualization::Add, Objects

IDLitVisualization::Aggregate - Adds the given object(s) to this visualization's property aggregate.

Obj->IDLitVisualization::Aggregate, Objects

IDLitVisualization::BeginManipulation - Handles notifications that an IDLitManipulator object is about to manipulate this visualization.

Obj->IDLitVisualization::BeginManipulation, Manipulator

IDLitVisualization::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLitVisualization::Cleanup

IDLitVisualization::EndManipulation - Handles notifications that an IDLitManipulator object has finished manipulating this visualization.

Obj->IDLitVisualization::EndManipulation, Manipulator

IDLitVisualization::Get - Retrieves object(s) from the visualization.

Result = Obj->IDLitVisualization::Get[/, ALL]

IDLitVisualization::GetCenterRotation - Returns the center of rotation for this visualization.

Result = Obj->IDLitVisualization::GetCenterRotation

IDLitVisualization::GetCurrentSelectionVisual - Returns the currently active selection visual object for this visualization.

Result = Obj->IDLitVisualization::GetCurrentSelectionVisual()

IDLitVisualization::GetDataSpace - Returns a reference to the parent data space object within the graphics hierarchy that contains this visualization.

Result = Obj->IDLitVisualization::GetDataSpace()

IDLitVisualization::GetDataString - Retrieves a description of this visualization's data at the given x, y, and z location.

Result = Obj->IDLitVisualization::GetDataString( XYZLocation)

IDLitVisualization::GetDefaultSelectionVisual - Returns an object that serves as the default selection visual for this visualization.

Result = Obj->IDLitVisualization::GetDefaultSelectionVisual()

IDLitVisualization::GetManipulatorTarget - Retrieves the manipulator target associated with this visualization.

Result = Obj->IDLitVisualization::GetManipulatorTarget()

IDLitVisualization::GetProperty - Retrieves the value of a property or group of properties for the object.

Result = Obj->IDLitVisualization::GetProperty

IDLitVisualization::GetRequestedAxesStyle - Returns the axes style requested by this visualization.

Result = Obj->IDLitVisualization::GetRequestedAxesStyle()

IDLitVisualization::GetSelectionVisual - Retrieves the selection visual for this visualization that corresponds to the given manipulator.

Result = Obj->IDLitVisualization::GetSelectionVisual(Manipulator)

IDLitVisualization::GetTypes - Identifies the types that this visualization represents, including base types and any specializations.

Result = Obj->IDLitVisualization::GetTypes()

IDLitVisualization::GetXYZRange - Computes the x, y, and z ranges of the overall contents of the visualization, taking into account the IMPACTS_RANGE property setting for itself and its contents.

Result = Obj->IDLitVisualization::GetXYZRange(XRange, YRange, ZRange[, /DATA][/NO_TRANSFORM])

IDLitVisualization::Init - Initializes the visualization object.

Result = Obj->IDLitVisualization::Init([, PROPERTY=value] ) or

IDLitVisualization::Is3D - Determines whether or not this visualization (or any of its contents) is marked as being three-dimensional.

Result = Obj->IDLitVisualization::Is3D()

IDLitVisualization::IsIsotropic - Indicates whether or not this visualization (or any of its contents) is marked as being isotropic.

Result = Obj->IDLitVisualization::IsIsotropic()

IDLitVisualization::IsManipulatorTarget - Determines whether or not this visualization is a manipulator target.

Result = Obj->IDLitVisualization::IsManipulatorTarget()

IDLitVisualization::IsSelected - Determines if this visualization is currently selected or not.

Result = Obj->IDLitVisualization::IsSelected()
IDLitVisualization::Move - Changes the location of an object within the visualization container.

Obj->IDLitVisualization::Move, Source, Destination

IDLitVisualization::On2DRotate - Ensures that objects that use the visualization’s data are notified when the rotation of the parent datasource changes.

Obj->IDLitVisualization::On2DRotate, Notifier, IsRotated

IDLitVisualization::OnAxesRequestChange - Ensures that objects that use the visualization’s data are notified when the axes request for this visualization is changed.

Obj->IDLitVisualization::OnAxesRequestChange, Notifier, AxesRequest

IDLitVisualization::OnAxesStyleRequestChange - Ensures that objects that use the visualization’s data are notified when the axes style of a contained object is changed.

Obj->IDLitVisualization::OnAxesStyleRequestChange, Notifier, StyleRequest

IDLitVisualization::OnDataChange - Ensures that objects that use the visualization’s data are notified when the visualization’s data changes.

Obj->IDLitVisualization::OnDataChange, Notifier

IDLitVisualization::OnDataComplete - Ensures that objects that use the visualization’s data are notified when the visualization’s data are complete.

Obj->IDLitVisualization::OnDataComplete, Notifier

IDLitVisualization::OnDataRangeChange - Ensures that objects that use the visualization’s data are notified when changes to the range of the visualization’s data changes.

Obj->IDLitVisualization::OnDataRangeChange, Notifier, XRange, YRange, Zrange

IDLitVisualization::OnDimensionChange - Ensures that objects that use the visualization’s data are notified when the visualization’s dimensionality changes.

Obj->IDLitVisualization::OnDimensionChange, Notifier, Is3D

IDLitVisualization::OnWorldDimensionChange - Ensures that objects that use the visualization’s data are notified when the visualization’s parent datasource’s dimensionality changes.

Obj->IDLitVisualization::OnWorldDimensionChange, Notifier, Is3D

IDLitVisualization::Remove - Removes the given object(s) from the visualization.

Obj->IDLitVisualization::Remove, Object

[./NO_UPDATE]

IDLitVisualization::RequestsAxes - Indicates whether or not the visualization requests axes. Returns a 1 if the visualization does request axes, or 0 if it does not request axes.

Result = Obj->IDLitVisualization::RequestsAxes()

IDLitVisualization::Restore - Performs any transitional work required after an object has been restored from a SAVE file.

Obj->IDLitVisualization::Restore

IDLitVisualization::Rotate - Rotates this visualization about the given axis by the given angle.

Obj->IDLitVisualization::Rotate, Axis, Angle

[./CENTER_OF_ROTATION=[x, y, z]]

[./PREMULTIPLY]

IDLitVisualization::Scale - Scales the visualization by the given scale factors.

Obj->IDLitVisualization::Scale, SX, SY, SZ

[./CENTER_OF_ROTATION=[x, y, z]]

[./PREMULTIPLY]

IDLitVisualization::Select - Handles notification of mechanisms that key off the current selection (such as the visualization browser) when this visualization has been selected.

Obj->IDLitVisualization::Select[, Mode]

[, /ADDITIVE | /SELECT | /TOGGLE | /UNSELECT] [, /NO_NOTIFY]

IDLitVisualization::Set3D - Sets a flag indicating this visualization is three-dimensional.

Obj->IDLitVisualization::Set3D, Is3D [, /ALWAYS] [, /AUTO_COMPUTE]

IDLitVisualization::SetAxesRequest - Sets the current axes request for this visualization.

Obj->IDLitVisualization::SetAxesRequest, AxesRequest

[, /ALWAYS] [, /AUTO_COMPUTE] [, /NO_NOTIFY]

IDLitVisualization::SetAxesStyleRequest - Sets the current axes style request for this visualization.

Obj->IDLitVisualization::SetAxesStyleRequest, StyleRequest

[, /NO_NOTIFY]

IDLitVisualization::SetCurrentSelectionVisual - Sets the current selection visual for the given manipulator.

Obj->IDLitVisualization::SetCurrentSelectionVisual, Manipulator

IDLitVisualization::SetData - Sets the data parameter of the visualization.

Result = Obj->IDLitVisualization::SetData(Data)

IDLitVisualization::SetDefaultSelectionVisual - Sets the default selection visual to be associated with this visualization.

Obj->IDLitVisualization::SetDefaultSelectionVisual, SelectionVisual

[, POSITION=value]

IDLitVisualization::SetParameterSet - Associates a parameter set with this visualization.

Result = Obj->IDLitVisualization::SetParameterSet(ParameterSet)

IDLitVisualization::SetProperty - Sets the value of a property or group of properties for the object.

Obj->IDLitVisualization::SetProperty

[, PROPERTY=value]

IDLitVisualization::UpdateSelectionVisual - Transforms this visualization’s selection visual to match the visualization’s geometry.

Obj->IDLitVisualization::UpdateSelectionVisual
Alphabetical List of IDL Routines

IDLitWindow::Add - Adds the given object(s) to the window.
Obj->IDLitWindow::Add, Objects [], POSITION=value

IDLitWindow::AddWindowEventObserver - Adds the given object(s) to the list of observers that are to be notified of events that occur within this window.
Obj->IDLitWindow::AddWindowEventObserver, Objects

IDLitWindow::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLitWindow::Cleanup

IDLitWindow::ClearSelections - Clears the window’s list of currently selected items (within its current view).
Obj->IDLitWindow::ClearSelections

IDLitWindow::DoHitTest - Performs a hit test to determine which visualizations within the destination are displayed at a given pixel location.
Result = Obj->DoHitTest(X, Y [, DIMENSIONS=[width, height]], [], /ORDER], [SUB_HIT=variable], [UNITS=[0 1 2 3]])

IDLitWindow::GetEventMask - Returns a bitwise mask representing the events that are enabled for this window.
Result = Obj->IDLitWindow::GetEventMask()
[. BUTTON_EVENTS=variable]
[. KEYBOARD_EVENTS=variable]
[. MOTION_EVENTS=variable]
[. /TRACKING_EVENTS=variable]

IDLitWindow::GetProperty - Retrieves the value of an IDLitWindow property
Obj->IDLitData::GetProperty, PROPERTY=value

IDLitWindow::GetSelectedItems - Returns the currently selected objects within this window’s scene, which represents a container for all of the views (and their corresponding visualization hierarchies) that appear within a window.
Result = Obj->IDLitWindow::GetSelectedItems([, /ALL] [. COUNT=named variable])

IDLitWindow::Init - Initializes the window object.
Obj = OBJ_NEW([IDLitWindow], [ PROPERTY=value]) or
Result = Obj->IDLitWindow::Init([, PROPERTY=value])

IDLitWindow::OnKeyboard - Handles notification (from the native window device) that a keyboard event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnKeyboard, IsASCII, Character, KeySymbol, X, Y, Press, Release, Modifiers

IDLitWindow::OnMouseDown - Handles notification (from the native window device) that a mouse down event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnMouseDown, X, Y, ButtonMask, Modifiers, NumClicks

IDLitWindow::OnMouseMotion - Handles notification (from the native window device) that a mouse motion event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnMouseMotion, X, Y, Modifiers

IDLitWindow::OnMouseUp - Handles notification (from the native window device) that a mouse up event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnMouseUp, X, Y, ButtonMask

IDLitWindow::OnScroll - Handles notification (from the native window device) that a scrolling event has occurred.
Obj->IDLitWindow::OnScroll, X, Y

IDLitWindow::Remove - Removes the given object(s) from the window.
Obj->IDLitWindow::Remove, Object [, /ALL] [, POSITION=index]

IDLitWindow::RemoveWindowEventObserver - Removes the given object(s) from the list of observers that are notified of events that occur within this window.
Obj->IDLitWindow::RemoveWindowEventObserver, Objects

IDLitWindow::SetCurrentZoom - Sets the current zoom factor for this window by changing its virtual dimensions.
Obj->IDLitWindow::SetCurrentZoom, ZoomFactor [, /RESET]

IDLitWindow::SetEventMask - Enables the given events within this window.
Obj->IDLitWindow::SetEventMask([EventMask]
[. BUTTON_EVENTS] [. KEYBOARD_EVENTS]
[. MOTION_EVENTS] [. /TRACKING_EVENTS])

IDL Quick Reference

Objects

IDLitVisualization::VisToWindow - Transforms given points from visualization data space to window device coordinates.
Obj->IDLitVisualization::VisToWindow, InX, InY, InZ, OutX, OutY, OutZ [, /NO_TRANSFORM] or Obj->IDLitVisualization::VisToWindow, InX, InY, OutX, OutY [, /NO_TRANSFORM] or Obj->IDLitVisualization::VisToWindow, InVerts, OutVerts [, /NO_TRANSFORM]

IDLitVisualization::WindowToVis - Transforms given points from window device coordinates to visualization data space.
Obj->IDLitVisualization::WindowToVis, InX, InY, InZ, OutX, OutY, OutZ or Obj->IDLitVisualization::WindowToVis, InX, InY, OutX, OutY or Obj->IDLitVisualization::WindowToVis, InVerts, OutVerts

IDLitWindow - The basis for all iTool visualization windows. All iTool visualization windows subclass from this class.
Properties: IDLitWindow inherits all properties from the IDLgrWindow superclass. See “IDLgrWindow” on page 96 for details.

IDLitWindow::Add - Adds the given object(s) to the window.
Obj->IDLitWindow::Add, Objects [], POSITION=value

IDLitWindow::AddWindowEventObserver - Adds the given object(s) to the list of observers that are to be notified of events that occur within this window.
Obj->IDLitWindow::AddWindowEventObserver, Objects

IDLitWindow::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLitWindow::Cleanup

IDLitWindow::ClearSelections - Clears the window’s list of currently selected items (within its current view).
Obj->IDLitWindow::ClearSelections

IDLitWindow::DoHitTest - Performs a hit test to determine which visualizations within the destination are displayed at a given pixel location.
Result = Obj->DoHitTest(X, Y [, DIMENSIONS=[width, height]], [], /ORDER], [SUB_HIT=variable], [UNITS=[0 1 2 3]])

IDLitWindow::GetEventMask - Returns a bitwise mask representing the events that are enabled for this window.
Result = Obj->IDLitWindow::GetEventMask()
[. BUTTON_EVENTS=variable]
[. KEYBOARD_EVENTS=variable]
[. MOTION_EVENTS=variable]
[. /TRACKING_EVENTS=variable]

IDLitWindow::GetProperty - Retrieves the value of an IDLitWindow property
Obj->IDLitData::GetProperty, PROPERTY=value

IDLitWindow::GetSelectedItems - Returns the currently selected objects within this window’s scene, which represents a container for all of the views (and their corresponding visualization hierarchies) that appear within a window.
Result = Obj->IDLitWindow::GetSelectedItems([, /ALL] [. COUNT=named variable])

IDLitWindow::Init - Initializes the window object.
Obj = OBJ_NEW([IDLitWindow], [ PROPERTY=value]) or
Result = Obj->IDLitWindow::Init([, PROPERTY=value])

IDLitWindow::OnKeyboard - Handles notification (from the native window device) that a keyboard event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnKeyboard, IsASCII, Character, KeySymbol, X, Y, Press, Release, Modifiers

IDLitWindow::OnMouseDown - Handles notification (from the native window device) that a mouse down event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnMouseDown, X, Y, ButtonMask, Modifiers, NumClicks

IDLitWindow::OnMouseMotion - Handles notification (from the native window device) that a mouse motion event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnMouseMotion, X, Y, Modifiers

IDLitWindow::OnMouseUp - Handles notification (from the native window device) that a mouse up event has occurred, and passes along that notification to all observers in the list of window event observers.
Obj->IDLitWindow::OnMouseUp, X, Y, ButtonMask

IDLitWindow::OnScroll - Handles notification (from the native window device) that a scrolling event has occurred.
Obj->IDLitWindow::OnScroll, X, Y

IDLitWindow::Remove - Removes the given object(s) from the window.
Obj->IDLitWindow::Remove, Object [, /ALL] [, POSITION=index]

IDLitWindow::RemoveWindowEventObserver - Removes the given object(s) from the list of observers that are notified of events that occur within this window.
Obj->IDLitWindow::RemoveWindowEventObserver, Objects

IDLitWindow::SetCurrentZoom - Sets the current zoom factor for this window by changing its virtual dimensions.
Obj->IDLitWindow::SetCurrentZoom, ZoomFactor [, /RESET]

IDLitWindow::SetEventMask - Enables the given events within this window.
Obj->IDLitWindow::SetEventMask([EventMask]
[. BUTTON_EVENTS] [. KEYBOARD_EVENTS]
[. MOTION_EVENTS] [. /TRACKING_EVENTS])
Alphabetical List of IDL Routines

IDLitWindow::SetManipulatorManager - Sets the given IDLManipulatorManager object as the current manager of this window's manipulators.

IDLitWindow::SetProperty - Sets the value of an IDLitWindow property.

IDLitWindow::ZoomIn - Causes the current zoom factor for this window to be increased (that is, multiplied by the factor given by the window's ZOOM_BASE property).

IDLitWindow::ZoomOut - Causes the current zoom factor for this window to be decreased (that is, divided by the factor given by the window's ZOOM_BASE property).

IDLitWriter - The definition of the interface and the process used to construct file writers for the iTools framework. When a new file writer is constructed for the iTools system, a new class is subclassed from this IDLitWriter class.

IDLitWriter::Cleanup - Performs all cleanup on the object.

IDLitWriter::GetFileExtensions - Is called by the system to retrieve the file extensions supported by this particular writer.

IDLitWriter::GetFilename - Is called by the system to set the current filename associated with this writer.

IDLitWriter::GetData - Is called by the system to set the data for the current file.

IDLitWriter::IsA - Is called by the system to determine if the given file is of the type supported by this file writer.

IDLitWriter::SetFilename - Is called by the system to set the current filename associated with this writer.

IDLlitWindow::SetManipulatorManager - Sets the given IDLManipulatorManager object as the current manager of this window's manipulators.

IDLlitWindow::SetProperty - Sets the value of an IDLitWindow property.

IDLlitWindow::ZoomIn - Causes the current zoom factor for this window to be increased (that is, multiplied by the factor given by the window's ZOOM_BASE property).

IDLlitWindow::ZoomOut - Causes the current zoom factor for this window to be decreased (that is, divided by the factor given by the window's ZOOM_BASE property).

IDLlitWriter - The definition of the interface and the process used to construct file writers for the iTools framework. When a new file writer is constructed for the iTools system, a new class is subclassed from this IDLitWriter class.

IDLlitWriter::Cleanup - Performs all cleanup on the object.

IDLlitWriter::GetFileExtensions - Is called by the system to retrieve the file extensions supported by this particular writer.

IDLlitWriter::GetFilename - Is called by the system to retrieve the current filename associated with this writer.

IDLlitWriter::GetData - Is called by the system to set the data for the current file.

IDLlitWriter::IsA - Is called by the system to determine if the given file is of the type supported by this file writer.

IDLlitWriter::SetFilename - Is called by the system to set the current filename associated with this writer.

IDLlitWindow::SetManipulatorManager - Sets the given IDLManipulatorManager object as the current manager of this window's manipulators.

IDLlitWindow::SetProperty - Sets the value of an IDLitWindow property.

IDLlitWindow::ZoomIn - Causes the current zoom factor for this window to be increased (that is, multiplied by the factor given by the window's ZOOM_BASE property).

IDLlitWindow::ZoomOut - Causes the current zoom factor for this window to be decreased (that is, divided by the factor given by the window's ZOOM_BASE property).

IDLlitWriter - The definition of the interface and the process used to construct file writers for the iTools framework. When a new file writer is constructed for the iTools system, a new class is subclassed from this IDLitWriter class.

IDLlitWriter::Cleanup - Performs all cleanup on the object.

IDLlitWriter::GetFileExtensions - Is called by the system to retrieve the file extensions supported by this particular writer.

IDLlitWriter::GetFilename - Is called by the system to retrieve the current filename associated with this writer.

IDLlitWriter::GetData - Is called by the system to set the data for the current file.

IDLlitWriter::IsA - Is called by the system to determine if the given file is of the type supported by this file writer.

IDLlitWriter::SetFilename - Is called by the system to set the current filename associated with this writer.

IDLjavaObject - An IDL object encapsulating a Java object. IDL provides data type and other translation services, allowing IDL programs to access the Java object's methods and properties using standard IDL syntax.

IDLjavaObject::GetProperty - Retrieves properties (known as data members in Java) from the Java object that underlies the IDLJavaObject.

IDLjavaObject::SetProperty - Sets properties (known as data members in Java) for the Java object that underlies an instance of IDLJavaObject.

TrackBall - Translates widget events from a draw widget (created with the WIDGET_DRAW function) into transformations that emulate a virtual trackball (for transforming object graphics in three dimensions).

Properties: [AXIS=[0 | 1 | 2]] [CONSTRAN[Init][MOUSE=[1 | 2 | 4]]]

TrackBall::Init - Initializes the TrackBall object.

TrackBall::Reset - Resets the state of the TrackBall object.

TrackBall::Update - Updates the state of the TrackBall object based on the information contained in the input widget event structure.
Statements

Assignment

*variable = expression* - Assigns a value to a variable.

*variable[subscripts] = expression* - Assigns a value to the elements of an array specified by the array subscripts.

*variable[script_range] = expression* - Assigns a value to the elements of an array specified by the array subscript range.

Program Control

Compound Statements

BEGIN...END - Defines a block of statements.

BEGIN
  statements
END | ENDF | ENDELSE | ENDFOR | ENDRP | ENDWHILE

Conditional Statements

IF...THEN...ELSE - Conditionally executes a statement or block of statements.

IF expression THEN statement [ ELSE statement ]

or

IF expression THEN BEGIN
  statements
ENDIF | ELSE BEGIN
  statements
ENDELSE |

CASE - Selects one statement for execution from multiple choices, depending on the value of an expression.

CASE expression OF
  expression: statement
  ...
  expression: statement
  [ ELSE: statement ]
ENDCASE

SWITCH - Selects one statement for execution from multiple choices, depending upon the value of an expression.

SWITCH expression OF
  expression: statement
  ...
  expression: statement
  [ ELSE: statement ]
ENDSWITCH

Loop Statements

FOR...DO - Executes one or more statements repeatedly, while incrementing or decrementing a variable with each repetition, until a condition is met.

FOR Variable = Init, Limit [, Increment] DO statement

or

FOR Variable = Init, Limit [, Increment] DO BEGIN
  statements
ENDFOR

REPEAT...UNTIL - Repeats statement(s) until expression evaluates to true. Subject is always executed at least once.

REPEAT statement UNTIL expression

or

REPEAT BEGIN
  statements
END IF expression

WHILE...DO - Performs statement(s) as long as expression evaluates to true. Subject is never executed if condition is initially false.

WHILE expression DO statement

or

WHILE expression DO BEGIN
  statements
ENDWHILE

Jump Statements

BREAK - Immediately exits from a loop (FOR, WHILE, REPEAT), CASE, or SWITCH statement without resorting to GOTO statements.

BREAK

CONTINUE - Immediately starts the next iteration of the enclosing FOR, WHILE, or REPEAT loop.

CONTINUE

GOTO - Transfers program control to point specified by *label*.

GOTO, *label*
Functions and Procedures

**COMPILE_OPT** - Gives IDL compiler information that changes the default rules for compiling functions or procedures.

```
COMPILE_OPT opt1 [, opt2, ..., optn]
```

*Note: optn can be IDL2, DEFINT32, HIDDEN, LOGICAL_PREDICATE, OBSOLETE, STRICTARR, or STRICTARRSUBS*

**FORWARD_FUNCTION** - Causes argument(s) to be interpreted as functions rather than variables (versions of IDL prior to 5.0 used parentheses to declare arrays).

```
FORWARD_FUNCTION Name1, Name2, ..., Namem
```

**FUNCTION** - Defines a function.

```
FUNCTION Function_Name, parameter1, ..., parameterm
```

**PRO** - Defines a procedure.

```
PRO Procedure_Name, argument1, ..., argumentm
```

**Procedure_Name** - Calls a procedure.

```
Procedure_Name, argument1, ..., argumentm
```

**Result = FUNCTION( arg1, ..., argm )** - Calls a function.

Variable Scope

**COMMON** - Creates a common block.

```
COMMON Block_Name, Variable1, ..., Variablem
```
Executive Commands

Executive commands must be entered at the IDL command prompt. They cannot be used in programs.

**.COMPILE** - Compiles programs without running.

```
.COMPILE [File1,..., File_n]
```

To compile from a temporary file: .COMPILE -File TempFile

**.CONTINUE** - Continues execution of a stopped program.

```
.CONTINUE
```

**.EDIT** - Opens files in editor windows of the IDLDE (Windows and Motif only). Note that filenames are separated by spaces, not commas.

```
.EDIT File1 [File2 File_n]
```

**.FULL_RESET_SESSION** - Does everything .RESET_SESSION does, plus additional reset tasks such as unloading sharable libraries.

```
.FULL_RESET_SESSION
```

**.GO** - Executes previously-compiled main program.

```
.GO
```

**.OUT** - Continues execution until the current routine returns.

```
.OUT
```

**.RESET_SESSION** - Resets much of the state of an IDL session without requiring the user to exit and restart the IDL session.

```
.RESET_SESSION
```

**.RETURN** - Continues execution until RETURN statement.

```
.RETURN
```

**.RNEW** - Erases main program variables and then does .RUN.

```
.RNEW [File1,..., File_n]
```

To save listing in a file:.RNEW -L ListFile.lis File1 [,..., File_n]

To display listing on screen: .RNEW -T File1 [,..., File_n]

**.RUN** - Compiles and executes IDL commands from files or keyboard.

```
.RUN [File1,..., File_n]
```

To save listing in a file:.RUN -L ListFile.lis File1 [,..., File_n]

To display listing on screen: .RUN -T File1 [,..., File_n]

**.SKIP** - Skips over the next n statements and then single steps.

```
.SKIP [n]
```

**.STEP** - Executes one or n statements from the current position.

```
.STEP [n] or .S [n]
```

**.STEPOVER** - Executes a single statement if the statement doesn’t call a routine.

```
.STEPOVER [n] or .SO [n]
```

**.TRACE** - Similar to .CONTINUE, but displays each line of code before execution.

```
.TRACE
```

**.COMPILE**
## Special Characters

The following table lists the characters that have a special meaning in IDL:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ampersand (&amp;)</td>
<td>Separates multiple commands on a single line</td>
</tr>
</tbody>
</table>
| Apostrophe (') | Delimits string constants  
|              | Indicates part of octal or hex constant                                    |
| Asterisk (*) | Multiplication operator  
|              | Array subscript range  
|              | Pointer dereference (if in front of a valid pointer)                       |
| At Sign (@) | Include character: Used at beginning of a line to cause the IDL compiler to substitute the contents of the file whose name appears after the @ symbol for the line.  
|              | In interactive mode, @ is used to execute a batch file.                     |
| Colon (:)   | Ends label identifiers  
|              | Separates start and end subscript ranges                                   |
| Dollar Sign ($) | Continue current command on the next line                                  |
|              | Issue operating system command if entered on a line by itself               |
| Exclamation Point (!) | First character of system variable names and font-positioning commands |
| Period (.)  | First character of executive commands  
|              | Indicates floating-point numbers  
|              | Indicates fields in a structure, such as in mystructure.field1             |
| Question Mark (?) | Invokes online help when entered at the IDL command line  
|                | Part of the ?: ternary operator used in conditional expressions           |
| Semicolon (;) | First character of comment field. Everything after the semicolon is ignored by IDL. Semicolon can be used as the first character or after an IDL command:  
|              | ; This is a comment  
|              | COUNT = 5 ; Set variable COUNT to 5                                        |
Subscripts

Subscripts are used to designate array elements to receive new values, and to retrieve the value of one or more array elements. IDL arrays are zero-based, meaning the first element is element 0.

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array[i, j]</td>
<td>The element stored at column i, row j of an array.</td>
</tr>
<tr>
<td>Vector[i:j]</td>
<td>Elements i through j of a vector.</td>
</tr>
<tr>
<td>Vector[i:*]</td>
<td>Elements from i through the end of a vector.</td>
</tr>
<tr>
<td>Array[i, *]</td>
<td>Column i of a two-dimensional array.</td>
</tr>
<tr>
<td>Array[* , j]</td>
<td>The jth row of a two-dimensional array.</td>
</tr>
<tr>
<td>Array[i:j, m:n]</td>
<td>Subarray of columns i through j, rows m through n.</td>
</tr>
<tr>
<td>Array[Array2]</td>
<td>The elements of Array whose subscripts are the values of Array2.</td>
</tr>
<tr>
<td>(Array_Expression)[i]</td>
<td>Element i of an array-valued expression.</td>
</tr>
</tbody>
</table>
# Operators

## Mathematical Operators
- `+`: Addition, String Concatenation
- `-`: Subtraction and Negation
- `*`: Multiplication, Pointer dereference
- `/`: Division
- `^`: Exponentiation
-`++`: Increment
-`--`: Decrement
- `MOD`: Modulo

## Minimum/Maximum Operators
- `<`: The Minimum Operator
- `>`: The Maximum Operator

## Matrix Operators
- `#` and `##`: Matrix Multiplication

## Logical Operators
- `&&`: Logical AND
- `||`: Logical OR
- `~`: Logical NOT
- `AND`: Bitwise AND
- `NOT`: Bitwise complement
- `OR`: Bitwise OR
- `XOR`: Bitwise exclusive OR

## Relational Operators
- `EQ`: Equal to
- `GE`: Greater than or equal to
- `GT`: Greater than
- `LE`: Less than or equal to
- `LT`: Less than
- `NE`: Not equal to

## Other Operators
- `[ ]`: Array concatenation, enclose array subscripts
- `( )`: Group expressions to control order of evaluation
- `=`: Assignment
- `op=`: Compound assignment
- `?:`: Conditional expression
**Operator Precedence**

Operators with the highest precedence are evaluated first. Operators with equal precedence are evaluated from left to right.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>First (highest)</td>
<td>( ) (parentheses, to group expressions)</td>
</tr>
<tr>
<td></td>
<td>[ ] (brackets, to concatenate arrays)</td>
</tr>
<tr>
<td>Second</td>
<td>. (structure field dereference)</td>
</tr>
<tr>
<td></td>
<td>[ ] (brackets, to subscript an array)</td>
</tr>
<tr>
<td></td>
<td>( ) (parentheses, used in a function call)</td>
</tr>
<tr>
<td>Third</td>
<td>* (pointer dereference)</td>
</tr>
<tr>
<td></td>
<td>^ (exponentiation)</td>
</tr>
<tr>
<td></td>
<td>++ (increment)</td>
</tr>
<tr>
<td></td>
<td>-- (decrement)</td>
</tr>
<tr>
<td>Fourth</td>
<td>* (multiplication)</td>
</tr>
<tr>
<td></td>
<td># and ## (matrix multiplication)</td>
</tr>
<tr>
<td></td>
<td>/ (division)</td>
</tr>
<tr>
<td></td>
<td>MOD (modulus)</td>
</tr>
<tr>
<td>Fifth</td>
<td>+ (addition)</td>
</tr>
<tr>
<td></td>
<td>- (subtraction and negation)</td>
</tr>
<tr>
<td></td>
<td>&lt; (minimum)</td>
</tr>
<tr>
<td></td>
<td>&gt; (maximum)</td>
</tr>
<tr>
<td></td>
<td>NOT (bitwise negation)</td>
</tr>
<tr>
<td>Sixth</td>
<td>EQ (equality)</td>
</tr>
<tr>
<td></td>
<td>NE (not equal)</td>
</tr>
<tr>
<td></td>
<td>LE (less than or equal)</td>
</tr>
<tr>
<td></td>
<td>LT (less than)</td>
</tr>
<tr>
<td></td>
<td>GE (greater than or equal)</td>
</tr>
<tr>
<td></td>
<td>GT (greater than)</td>
</tr>
<tr>
<td>Seventh</td>
<td>AND (bitwise AND)</td>
</tr>
<tr>
<td></td>
<td>OR (bitwise OR)</td>
</tr>
<tr>
<td></td>
<td>XOR (bitwise exclusive OR)</td>
</tr>
<tr>
<td>Eighth</td>
<td>&amp; &amp; (logical AND)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>~ (logical negation)</td>
</tr>
<tr>
<td>Ninth</td>
<td>? : (conditional expression)</td>
</tr>
</tbody>
</table>
System Variables

IDL system variables contain useful constants, control plotting defaults, and store information about the current IDL session.

Constant System Variables

| IDPI | Double-precision pi (p). |
| IDTOR | Degrees to radians, pi/180 = 0.01745. |
| IMAP | Read-only system variable used by MAP_SET. |
| IP | Single-precision pi (p). |
| IRADEG | Radians to degrees, 180/pi ≈ 57.2958. |
| VALUES | Single- and double-precision NaN and Infinity values. |

Graphics System Variables

| ID | Information about current graphics device. |
| ID | Fields: FILL_DIST - line interval, in device coordinates |
| ID | FLAGS - longword of flags |
| ID | N_COLORS - number of simultaneously available colors |
| ID | NAME - string containing name of device |
| ID | ORIGIN - pan/scroll offset (pan, scroll) |
| ID | TABLE_SIZE - number of color table indices |
| ID | UNIT - logical number of file open for output |
| ID | WINDOW - index of currently open window |
| ID | X_CH_SIZE, Y_CHAR_SIZE - width/height of rectangle that encloses the average character in current font, in device units (usually pixels) |
| ID | X_PIXELS_CM, Y_PIXELS_CM - approx. number of pixels/cm |
| ID | X_SIZE, Y_SIZE - total size of the display or window, in device units |
| ID | X_VSIZE, Y_VSIZE - size of visible area of display or window |
| ID | ZOOM - X and Y zoom factors |

| !ORDER | Direction of image transfer: 0=bottom up, 1=top down. |
| !P | Information for plotting procedures. |

Fields:

| BACKGROUND | background color index |
| CHANNEL | default source or destination channel |
| CHARSIZE | character size of annotation when Hershey fonts are selected |
| CHARTHICK | integer specifying thickness of vector fonts |
| CLIP | device coords of clipping window ( [x₀, y₀, z₀], (x₁, y₁, z₁) ) |
| COLOR | default color index |
| FONT | integer specifying graphics text font system to use (-1 for Hershey, 0 for output device font, 1 for TrueType) |
| LINESTYLE | style of lines that connect points (see "Line Styles" on page 121) |
| MULTI | integer array: [plots remaining on page, columns per page, rows per page, plots in Z direction, 0 for left to right or 1 for top to bottom] |
| NOCLIP | if set, inhibits clipping of graphic vectors |
| NOERASE | set to nonzero value to prevent erasing |
| NSUM | number of adjacent points to average |
| POSITION | normalized coords of plot window (x₀, y₀, x₁, y₁) |
| PSYM | plotting symbol index (see "Plotting Symbols" on page 121) |
| REGION | normalized coords of plot region (x₀, y₀, x₁, y₁) |
| SUBTITLE | plot subtitle (under X axis label) |
| T | homogeneous 4 x 4 transformation matrix |
| T3D | enables 3D to 2D transformation |
| THICK | thickness of lines connecting points |
| TITLE | main plot title |
| TICKLEN | tick mark length (0.0 to 1.0) |
Error Handling/Informational System Variables

!ERROR_STATE - Structure containing all error information.
  Fields: NAME - string containing error name of IDL-generated component of last error message (read-only).
  BLOCK - string containing name of message block for IDL-generated component of last error message.
  CODE - long-integer containing error code of IDL-generated component of last error message.
  SYS_CODE - long-integer containing error code of operating system component of last error message.
  SYS_CODE_TYPE - A string describing the type of system code contained in SYS_CODE.
  MSG - string containing text of IDL-generated component of last error message (read-only).
  MSG_PREFIX - string containing prefix string used for error messages.
  SYS_MSG - string containing text of operating system generated component of last error message (read-only).

!EXCEPT - Controls when IDL checks for math error conditions
  (0=never report exceptions, 1=report exceptions when interpreter is returning to interactive prompt, 2=report exceptions at end of each IDL statement).

!MOUSE - Status from the last cursor read operation.
  Fields: X, Y - location (in device coords) of cursor when mouse button was pressed
  BUTTON - specifies which mouse button was pressed (1 if left, 2 if middle, 4 if right)
  TIME - number of milliseconds since a base time

!WARN - Report use of obsolete routines.
  Fields: OBS_ROUTINES - if set to 1, IDL generates warnings when it encounters use of obsolete routines
  OBS_SYSVARS - if set to 1, IDL generates warnings when it encounters use of obsolete system variables
  PARENS - if set to 1, IDL generates warnings when it encounters use parentheses to index array
IDL Environment System Variables

!CPU - Read-only variable that supplies information about the state of the system processor, and of IDL's use of it.

**Fields:**
- HW_VECTOR - True (1) if the system supports a vector unit (e.g. Macintosh Altivec/Velocity Engine) or False (0) otherwise.
- VECTOR_ENABLE - True (1) if IDL will use a vector unit, if such a unit is available on the current system, and False (0) otherwise.
- HW_NCPU - The number of CPUs contained in the system on which IDL is currently running.
- TPOOL_NTHREADS - The number of threads that IDL will use in thread pool computations.
- TPOOL_MIN_ELTS - The number of elements in a computation that are necessary before IDL will use the thread pool to perform the work
- TPOOL_MAX_ELTS - The maximum number of elements in a computation for which IDL will use the thread pool.

!DIR - Location of the main IDL directory.

!DLM_PATH - Indicates where IDL looks for Dynamically Loadable Modules when started. Read-only.

!EDIT_INPUT - Enables/disables keyboard line editing.

!HELP_PATH - Lists directories IDL will search for online help files

!JOURNAL - Logical unit number of journal output, or 0.

!MORE - Set to 0 to prevent paginating help text.

!MAKE_DLL - Used to configure how IDL uses the CALL_EXTERNAL, DLMs, and LINKIMAGE for the current platform.

!PATH - Search path for IDL routines.
- UNIX: colon-separated list of directories.
- Windows: semicolon-separated list of directories.

!PROMPT - String to be used for IDL prompt.

!QUIET - Suppresses informational messages if set to nonzero.

!VERSION - Type, architecture, and version of IDL.

**Fields:**
- ARCH - CPU hardware architecture of the system.
- OS - The name of the underlying operating system kernel (e.g. AIX, sunos, Win32).
- OS_FAMILY - The generic name of the operating system (e.g. UNIX, Windows).
- OS_NAME - The vendor's name for the operating environment (e.g. Solaris, Microsoft Windows).
- RELEASE - The IDL version number.
- BUILD_DATE - Date the IDL executable was compiled.
Graphics Information

Direct Graphics Devices

CGM - The CGM Device
HP - The HP-GL Device
NULL - The Null Display Device
PCL - The PCL Device
PRINTER - The Printer Device
PS - The PostScript Device
REGIS - The Regis Terminal Device
TEK - The Tektronix Device
WIN - The Microsoft Windows Device
X - The X Windows Device
Z - The Z-Buffer Device

Graphics Keywords

The following keywords are used with IDL plotting routines (AXIS, CONTOUR, PLOT, O PLOT, SHADE_SURF, and SURFACE) and graphics routines (CURSOR, ERASE, PLOTS, POLYFILL, TV, TVCRS, TVRD, and XYOUTS). Many have system variable equivalents. Not all keywords work with all routines. Listings such as (XYZ)KEYWORD indicate that there are 3 keywords, one for each axis (e.g., XCHARSIZE, YCHARSIZE, ZCHARSIZE).

BACKGROUND - Background color index when erasing.
CHANNEL - Channel index or mask for multi-channel displays.
CHARSIZE - Overall character size.
{XYZ}CHARSIZE - Character size for axes.
CHARTHICK - Overall thickness for vector fonts.
CLIP - Coordinates of clipping window.
COLOR - Color index for data, text, line, or polygon fill.
DATA - Set to plot in data coordinates.
DEVICE - Set to plot in device coordinates.

FONT - Text font index: -1 for vector, 0 for hardware fonts.
{XYZ}GRIDSTYLE - Linestyle index for tickmarks and grids.
LINESTYLE - Linestyle used to connect data points.
{XYZ}MARGIN - Margin of plot window in character units.
{XYZ}MINOR - Number of minor tick marks.
NOCLIP - Set to disable clipping of plot.
NODATA - Set to plot only axes, titles, and annotation w/o data.
NOERASE - Set to inhibit erasing before new plot.
NORMAL - Set to plot in normal coordinates.
ORIENTATION - Angle (in degrees counter-clockwise) for text.
POSITION - Position of plot window.
PSYM - Use plotting symbols to plot data points.
{XYZ}RANGE - Axis range.
{XYZ}STYLE - Axis type.
SUBTITLE - String for subtitle.
SYMSIZE - Size of PSYM plotting symbols.
T3D - Set to use 3D transformation store in !P.T.
THICK - Overall line thickness.
{XYZ}THICK - Thickness of axis and tickmark lines.
{XYZ}TICKFORMAT - Allows advanced formatting of tick labels.
{XYZ}TICKINTERVAL - Set to indicate the interval between major tick marks for the first axis level.
{XYZ}TICKLAYOUT - Set to indicate the tick layout style to be used to draw each level of the axes.
TICKLEN - Length of tickmarks in normal coordinates. 1.0 produces a grid. Negative values extend outside window.
{XYZ}TICKLEN - Tickmark lengths for individual axes.
{XYZ}TICKNAME - String array of up to 30 elements for tick mark annotation.
{XYZ}TICKS - Number of major tick intervals for axes.
{XYZ}TICKUNITS - Set to indicate the units to be used for axis tick labeling.
{XYZ}TICKV - Array of up to 30 elements for tick mark values.
{XYZ}TICK_GET - Variable in which to return values of tick mark.
TITLE - String for plot title.
{XYZ}TITLE - String for specified axis title.
ZVALUE - The Z coordinate for a 2D plot in 3D space.
Z - Z coordinate if Z argument not specified in 3D plot call.
### Line Styles

The LINESTYLE keyword to the Direct Graphics plotting routines O/PLOT, PLOT, PLOTS, and SURFACE accepts the following values:

<table>
<thead>
<tr>
<th>Index</th>
<th>Linestyle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Solid</td>
</tr>
<tr>
<td>1</td>
<td>Dotted</td>
</tr>
<tr>
<td>2</td>
<td>Dashed</td>
</tr>
<tr>
<td>3</td>
<td>Dash Dot</td>
</tr>
<tr>
<td>4</td>
<td>Dash Dot Dot Dot</td>
</tr>
<tr>
<td>5</td>
<td>Long Dashes</td>
</tr>
</tbody>
</table>

### Plotting Symbols

The PSYM keyword to Direct Graphics plotting routines O/PLOT, PLOT, and PLOTS accepts the following values:

<table>
<thead>
<tr>
<th>PSYM Value</th>
<th>Plotting Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Plus sign (+)</td>
</tr>
<tr>
<td>2</td>
<td>Asterisk (*)</td>
</tr>
<tr>
<td>3</td>
<td>Period (.)</td>
</tr>
<tr>
<td>4</td>
<td>Diamond</td>
</tr>
<tr>
<td>5</td>
<td>Triangle</td>
</tr>
<tr>
<td>6</td>
<td>Square</td>
</tr>
<tr>
<td>7</td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td>User-defined. See USERSYM procedure.</td>
</tr>
<tr>
<td>9</td>
<td>Undefined</td>
</tr>
<tr>
<td>10</td>
<td>Histogram mode.</td>
</tr>
</tbody>
</table>
This quick reference guide contains an alphabetical listing of all scientific data format routines including CDF*, EOS* and HDF* routines. The alphabetical listing contains all functions, procedures, and statements including the syntax of each.
CDF Routines

CDF_ATTCREATE - Creates a new attribute.
   Result = CDF_ATTCREATE(Id, Attribute_Name [, /GLOBAL_SCOPE] [, /VARIABLE_SCOPE] )

CDF_ATDELETE - Deletes attribute from specified CDF file.
   CDF_ATDELETE, Id, Attribute [, EntryNum]
   [, /ZVARIABLE]

CDF_ATTEXIST - Determines whether specified attribute exists.
   Result = CDF_ATTEXIST(Id, Attribute [, EntryNum]
   [, /ZVARIABLE]

CDF_ATTGET - Reads an attribute entry from a CDF file.
   CDF_ATTGET, Id, Attribute, EntryNum, Value
   [, CDF_TYPE=variable] [, /ZVARIABLE]

CDF_ATTINFO - Obtains information about specified attribute.
   CDF_ATTINFO, Id, Attribute, Name, Scope, MaxEntry
   [, MaxZEntry]

CDF_ATTNUM - Returns an attribute number.
   Result = CDF_ATTNUM(Id, Attribute_Name)

CDF_ATTPUT - Writes an attribute entry to a CDF file.
   CDF_ATTPUT, Id, Attribute, EntryNum, Value
   [, /ZVARIABLE]

CDF_ATTREN - Renames an existing attribute.
   CDF_ATTREN, Id, OldAttr, NewName

CDF_CLOSE - Closes specified Common Data Format file.
   CDF_CLOSE, Id

CDF_COMPRESSION - Sets or returns the compression mode for a CDF file and/or variables.
   CDF_COMPRESSION, Id
   [, GET_COMPRESSION=variable]
   [, GET_GZIP_LEVEL=variable]
   [, GET_VAR_COMPRESSION=variable]
   [, GET_VAR_GZIP_LEVEL=variable]
   [, SET_COMPRESSION={0 | 1 | 2 | 3 | 5}]
   [, SET_GZIP_LEVEL=integer {1 to 9}]
   [, SET_VAR_COMPRESSION={0 | 1 | 2 | 3 | 5}]
   [, SET_VAR_GZIP_LEVEL=integer {1 to 9}]
   [, VARIABLE=variable name or index] [, /ZVARIABLE]

CDF_CONTROL - Obtains or sets information for a CDF file.
   CDF_CONTROL, Id [, ATTRIBUTE=name or number]
   [, GET_ATTRINFO=variable]
   [, GET_COPYRIGHT=variable]
   [, GET_FILENAME=variable]
   [, GET_FORMAT=variable]
   [, GET_NEGTOPOSFP0_MODE=variable]
   [, GET_NUMATTRS=variable]
   [, GET_READONLY_MODE=variable]
   [, GET_RVAR_CACHESIZE=variable]
   [, GET_VAR_INFO=variable]
   [, GET_ZMODE=variable]
   [, GET_ZVAR_CACHESIZE=variable]
   [, SET_CACHESIZE=value]
   [, SET_EXTENDRECS=records]
   [, SET_INITIALRECS=records]
   [, SET_NEGTOPOSFP0_MODE]
   [, SET_READONLY_MODE]
   [, SET_RVAR_CACHESIZE=value[See Note]]
   [, SET_RVARS_CACHESIZE=value[See Note]]
   [, SET_ZMODE={0 | 1 | 2}]
   [, SET_ZVAR_CACHESIZE=value[See Note]]
   [, SET_ZVARS_CACHESIZE=value[See Note]]
   [, VARIABLE=variable name or index] [, /ZVARIABLE]

Note: Use only with MULTI_FILE CDF files

CDF_CREATE - Creates a new Common Data Format file.
   Result = CDF_CREATE( Filename, [Dimensions]
   [, /CLOBBER] [, /MULTI_FILE] [, /SINGLE_FILE]
   [, /COL_MAJOR] [, /ROW_MAJOR] )

Encoding Keywords (pick one):
   [, /ALPHAOSF1_ENCODING]
   [, /ALPHAVMSD_ENCODING]
   [, /ALPHAVMSG_ENCODING]
   [, /DECSTATION_ENCODING]
   [, /HOST_ENCODING]
   [, /HP_ENCODING]
   [, /IBMPC_ENCODING]
   [, /MAC_ENCODING]
   [, /NETWORK_ENCODING]
   [, /NEXT_ENCODING]
   [, /SGI_ENCODING]
   [, /SUN_ENCODING]

Decoding Keywords (pick one):
   [, /ALPHAOSF1_DECODING]
   [, /ALPHAVMSD_DECODING]
   [, /ALPHAVMSG_DECODING]
   [, /DECSTATION_DECODING]
   [, /HOST_DECODING]
   [, /IBMPC_DECODING]
   [, /IBMRS_DECODING]
   [, /MAC_DECODING]
   [, /NETWORK_DECODING]
   [, /NEXT_DECODING]
   [, /SGI_DECODING]
   [, /SUN_DECODING]

CDF_DELETE - Deletes specified Common Data Format file.
   CDF_DELETE, Id
CDF_DOC - Gets documentation information about a CDF file.
CDF_DOC, Id, Version, Release, Copyright
[, INCREMENT=variable]

CDF_ENCODE_EPOCH - Encodes CDF_EPOCH variable into a string.
Result = CDF_ENCODE_EPOCH(Epoch, [ EPOCH=[0 | 1 | 2 | 3]])

CDF_EPOCH - Computes/breaks down CDF_EPOCH values.
CDF_EPOCH, Epoch, Year [, Month, Day, Hour, Minute, Second, Milli] [, BREAKDOWN_EPOCH]
[ , COMPUTE_EPOCH]

CDF_ERROR - Returns explanation of a given status code.
Result = CDF_ERROR(Status)

CDF_EXISTS - Returns True if CDF data format library is supported on the current IDL platform.
Result = CDF_EXISTS()

CDF_INQUIRE - Returns global information about CDF file.
Result = CDF_INQUIRE(Id)

CDF_LIB_INFO - Returns information about the CDF Library being used.
CDF_LIB_INFO [, COPYRIGHT=variable] [, INCREMENT=variable] [, SUBINCREMENT=variable] [, VERSION=variable]

CDF_OPEN - Opens an existing Common Data Format file.
Result = CDF_OPEN(Filename)

CDF_PARSE_EPOCH - Parses input string into a double precision value properly formatted for use as CDF_EPOCH variable.
Result = CDF_PARSE_EPOCH(Epoch_string)

CDF_VARCREATE - Creates new variable in CDF file.
Result = CDF_VARCREATE(Id, Name [, DimVar],
[, CDF_BYTE | CDF_CHAR | CDF_DOUBLE | CDF_EPOCH | CDF_FLOAT | CDF_INT1 | CDF_INT2 | CDF_INT4 | CDF_REAL4 | CDF_REAL8 | CDF_UINT1 | CDF_UINT2 | CDF_UINT4]
[, ALLOCATERECs=record] [, DIMENSIONS=array] [, NUMELEM=characters] [, REC_VOry] [, ZVARiABLE])

CDF_VARDELETE - Deletes variable from a SINGLE_FILE CDF file.
CDF_VARDELETE, Id, Variable [, ZVARiABLE]

CDF_VARGET - Reads multiple values from CDF file variable.
CDF_VARGET, Id, Variable, Value [, COUNT=vector]
[, INTERVAL=vector] [, OFFSET=vector]
[, REC_COUNT=records] [, REC_INTERVAL=vector]
[, REC_START=record] [, STRING] [data in CDF file must be type CDF_CHAR or CDF_UCHAR]
[ , ZVARIABLE]

CDF_VARGET1 - Reads one value from a CDF file variable.
CDF_VARGET1, Id, Variable, Value [, OFFSET=vector]
[, REC_COUNT=record] [, STRING] [data in CDF file must be type CDF_CHAR or CDF_UCHAR]
[, ZVARIABLE]

CDF_VARINQ - Returns structure containing information about specified variable.
Result = CDF_VARINQ( Id, Variable [, ZVARIABLE] )

CDF_VARNUM - Returns variable number associated with given variable name.
Result = CDF_VARNUM( Id, VarName [, IsVar] )

CDF_VARPUT - Writes value to a variable.
CDF_VARPUT, Id, Variable, Value [, COUNT=vector]
[, INTERVAL=vector] [, OFFSET=vector]
[, REC_INTERVAL=vector] [, REC_START=record]
[, ZVARIABLE]

CDF_VARRENAME - Renames existing variable.
CDF_VARRENAME, Id, OldVariable, NewName
[, ZVARIABLE]

EOS Routines

EOS_EH_CONVANG - Converts angles between decimal degrees, radians, and packed degrees-minutes-seconds.
Result = EOS_EH_CONVANG(inAngle, code)

EOS_EH_GETVERSION - Retrieves the HDF-EOS version string of an HDF-EOS file.
Result = EOS_EH_GETVERSION(fid, version)

EOS_EH_IDINFO - Returns the HDF file IDs corresponding to the HDF-EOS file ID returned by EOS_SW_OPEN, EOS_GD_OPEN, or EOS_PT_OPEN.
Result = EOS_EH_IDINFO(fid, HDFfid, sdInterfaceID)

EOS_EXISTS - Returns True if HDF EOS format library is supported on the current IDL platform.
Result = EOS_EXISTS()

EOS_GD_ATTACH - Attaches to the grid using the gridname parameter as the identifier.
Result = EOS_GD_ATTACH(fid, gridname)

EOS_GD_ATTRINFO - Returns number type and number of elements (count) of a grid attribute.
Result = EOS_GD_ATTRINFO(gridID, attname, numertype, count)

EOS_GD_CLOSE - Closes the HDF grid file.
Result = EOS_GD_CLOSE(fid)

EOS_GD_COMPINFO - Returns the compression code and compression parameters for a given field.
Result = EOS_GD_COMPINFO(gridID, fieldName, compcode, compparm)

IDL Quick Reference
EOS_GD_CREATE - Creates a grid within the file.
Result = EOS_GD_CREATE(gridID, gridname, xdimsize, ydimsize, upleftpt, lowrightpt)

EOS_GD_DEFBOXREGION - Defines a longitude-latitude box region for a grid.
Result = EOS_GD_DEFBOXREGION(gridID, cornerlon, cornerlat)

EOS_GD_DEFDEORIGIN - Defines the origin of the grid data.
Result = EOS_GD_DEFDEORIGIN(gridID, origincode)

EOS_GD_DEFPPIXREG - Defines whether the pixel center or pixel corner is used when requesting the location of a given pixel.
Result = EOS_GD_DEFPPIXREG(gridID, pixreg)

EOS_GD_DEFBLOCK - Defines dimensions used by field definition routines to establish size of the field.
Result = EOS_GD_DEFBLOCK(gridID, dimname, dimsize)

EOS_GD_DEFDEFFIELD - Defines data fields to be stored in the grid.
Result = EOS_GD_DEFDEFFIELD(gridID, fieldname, dimlist, numbertype, [. /MERGE])

EOS_GD_DEFDEFRIGIN - Defines the pixel registration code.
Result = EOS_GD_DEFDEFRIGIN(gridID, pixregcode)

EOS_GD_DEFINF - Defines data fields to be stored in the grid.
Result = EOS_GD_DEFINF(gridID, fieldname, fieldlist)

EOS_GD_DEFGGRIDINFO - Retrieves information on a specific data field.
Result = EOS_GD_DEFGGRIDINFO(gridID, fieldname, rank, dims, numbertype, dimlist)

EOS_GD_GETFILLVALUE - Retrieves fill value for specified field.
Result = EOS_GD_GETFILLVALUE(gridID, fieldname, fillvalue)

EOS_GD_GETPIXVALUES - Reads data from a data field for the specified longitude/latitude pairs.
Result = EOS_GD_GETPIXVALUES(gridID, nPixels, pixRow, pixCol)

EOS_GD_GETPIXVALUES - Reads data from a data field for the specified pixels.
Result = EOS_GD_GETPIXVALUES(gridID, nPixels, pixRow, pixCol, fieldname, buffer)

EOS_GD_GRIDINFO - Returns number of rows, columns, and the location of the upper left and lower right corners of the grid image.
Result = EOS_GD_GRIDINFO(gridID, xdimsize, ydimsize, upleft, lowright)

EOS_GD_INQATTRS - Retrieves information about attributes defined in grid.
Result = EOS_GD_INQATTRS(gridID, attrlist, [. LENGTH (OUT)=value ])

EOS_GD_INQDIM - Retrieves information about dimensions defined in grid.
Result = EOS_GD_INQDIM(gridID, dimname, dims)

EOS_GD_INQFIELDS - Retrieves information about the data fields defined in grid.
Result = EOS_GD_INQFIELDS(gridID, fieldlist, rank, numbertype)

EOS_GD_INQGRID - Retrieves number and names of grids defined in HDF-EOS file.
Result = EOS_GD_INQGRID(filename, gridlist, [. LENGTH (OUT)=value ])

EOS_GD_INTERPOLATE - Performs bilinear interpolation on a grid field.
Result = EOS_GD_INTERPOLATE(gridID, Interp, lonVal, latVal, interpVal)

EOS_GD_NENTRIES - Returns number of entries and descriptive string buffer size for a specified entity.
Result = EOS_GD_NENTRIES(gridID, entrycode, [. LENGTH (OUT)=value ])

EOS_GD_OPEN - Opens an existing file or creates a new file.
Result = EOS_GD_OPEN(filename, access [. /CREATE] [. /RDWR | /READ])

EOS_GD_ORIGININFO - Retrieves origin code.
Result = EOS_GD_ORIGININFO(gridID, origincode)

EOS_GD_PIXREGINFO - Retrieves the pixel registration code.
Result = EOS_GD_PIXREGINFO(gridID, pixregcode)
EOS_GD_PROJINFO - Retrieves GCTP projection code, zone code, spheroid code, and projection parameters of the grid.  
Result = EOS_GD_PROJINFO(gridID, projcode, zonecode, spheroidcode, projparm)

EOS_GD_QUERY - Returns information about a specified grid.  
Result = EOS_GD_QUERY( Filename, GridName, [Info])

EOS_GD_READATTR - Reads attribute from a grid.  
Result = EOS_GD_READATTR(gridID, attname, datbuf)

EOS_GD_READFIELD - Reads data from a grid field.  

EOS_GD_READTILE - Reads from tile within field.  
Result = EOS_GD_READTILE(gridID, fieldname, tilecoords, buffer)

EOS_GD_REGIONINFO - Returns information about a subsetted region for a particular field.  
Result = EOS_GD_REGIONINFO(gridID, regionID, fieldname, ntype, rank, dims, size, upleftpt, lowrightpt)

EOS_GD_SETFILLVALUE - Sets fill value for the specified field.  
Result = EOS_GD_SETFILLVALUE(gridID, fieldname, fillvalue)

EOS_GD_SETTILECACHE - Sets tile cache parameters.  
Result = EOS_GD_SETTILECACHE(gridID, fieldname, maxcache, cachecode)

EOS_GD_TILEINFO - Returns tiling code, tiling rank, and tiling dimensions for a given field.  
Result = EOS_GD_TILEINFO(gridID, fieldname, tilecode, tilerank, tiledims)

EOS_GD_WRITEATTR - Writes/updates attribute in a grid.  
Result = EOS_GD_WRITEATTR( gridID, attname, datbuf [, COUNT=value ] [, HDF_TYPE=value ] )

EOS_GD_WRITEFIELD - Writes data to a grid field.  
Result = EOS_GD_WRITEFIELD(gridID, fieldname, data [, EDGE=array] [, START=array] [, STRIDE=array])

EOS_GD_WRITEFILEMETA - Writes field metadata for a grid field not defined by the Grid API.  
Result = EOS_GD_WRITEFILEMETA(gridID, fieldname, dimlist, numbertype)

EOS_GD_WRITE TILE - Writes a single tile of data to a field.  
Result = EOS_GD_WRITE TILE(gridID, fieldname, tilecoords, data)

EOS_PT_ATTACH - Attaches to point using the pointname parameter as the identifier.  
Result = EOS_PT_ATTACH(fid, pointname)

EOS_PT_ATTRINFO - Returns number type and number of elements of a point attribute.  
Result = EOS_PT_ATTRINFO( pointID, attname, numbertype, count)

EOS_PT_BCKLINKINFO - Returns linkfield to the previous level.  
Result = EOS_PT_BCKLINKINFO(pointID, level, linkfield)

EOS_PT_CLOSE - Closes the HDF point file.  
Result = EOS_PT_CLOSE(fid)

EOS_PT_CREATE - Creates point as a Vgroup within the HDF file.  
Result = EOS_PT_CREATE(fid, pointname)

EOS_PT_DEFBOXREGION - Defines area of interest for a point.  
Result = EOS_PT_DEFBOXREGION(pointID, cornerlon, cornerlat)

EOS_PT_DEFLEVEL - Defines a level within a point.  
Result = EOS_PT_DEFLEVEL(pointID, levelname, fieldlist, fieldtype, fieldorder)

EOS_PT_DEFLINKAGE - Defines linkfield between two levels.  
Result = EOS_PT_DEFLINKAGE(pointID, parent, child, linkfield)

EOS_PT_DEFTIMEPERIOD - Defines a time period for a point.  
Result = EOS_PT_DEFTIMEPERIOD(pointID, starttime, stoptime)

EOS_PT_DEFVRTREGION - Selects records within a point whose field values are within a given range.  
Result = EOS_PT_DEFVRTREGION( pointID, regionID, vertObj, range)

EOS_PT_DETACH - Detaches from a point data set.  
Result = EOS_PT_DETACH(pointID)

EOS_PT_EXTRACTPERIOD - Reads data from the designated level fields into the data buffer from the subsetted time period.  
Result = EOS_PT_EXTRACTPERIOD(pointID, periodID, level, fieldlist, buffer)

EOS_PT_EXTRACTREGION - Reads data from the designated level fields into the data buffer from the subsetted area of interest.  
Result = EOS_PT_EXTRACTREGION( pointID, regionID, level, fieldlist, buffer)

EOS_PT_FWDLINKINFO - Returns linkfield to the given level.  
Result = EOS_PT_FWDLINKINFO(pointID, level, linkfield)

EOS_PT_GETLEVELNAME - Returns the name of a level given the level number (0-based).  
Result = EOS_PT_GETLEVELNAME( pointID, level, levelname [, LENGTH=OUT=variable] )

EOS_PT_GETRECNUMS - Returns record numbers in one level that are connected to a given set of records in a different level.  
Result = EOS_PT_GETRECNUMS( pointID, inlevel, outlevel, inRecs, inRecs, outRecs)

EOS_PT_INQATTRS - Returns attribute list as a comma-separated string.  
Result = EOS_PT_INQATTRS( pointID, attlist [, LENGTH=value] )
EOS_PT_INQPOINT - Retrieves number and names of points defined in HDF-EOS file.
   Result = EOS_PT_INQPOINT( filename, pointlist [, LENGTH(OUT)=value] )

EOS_PT_LEVELINDEX - Returns the level index for a given level.
   Result = EOS_PT_LEVELINDEX( pointID, levelname )

EOS_PT_LEVELINFO - Returns information about the fields in a given level.
   Result = EOS_PT_LEVELINFO( pointID, level, fieldlist, fldtype, fldorder )

EOS_PT_NFIELDS - Returns the number of fields in a level.
   Result = EOS_PT_NFIELDS( pointID, level )

EOS_PT_NLEVELS - Returns the number of levels in a point.
   Result = EOS_PT_NLEVELS( pointID )

EOS_PT_OPEN - Creates a new file or opens an existing one.
   Result = EOS_PT_OPEN( filename [, /CREATE] [, LENGTH=length] )

EOS_PT_PERIODINFO - Returns information about a subsetted time period for a given level.
   Result = EOS_PT_PERIODINFO( pointID, periodID, level, fieldlist, size )

EOS_PT_PERIODRECS - Returns record numbers within a subsetted time period for a given level.
   Result = EOS_PT_PERIODRECS( pointID, periodID )

EOS_PT_QUERY - Returns information about a specified point.
   Result = EOS_PT_QUERY( Filename, PointName [, Info] )

EOS_PT_READATTR - Reads attributes.
   Result = EOS_PT_READATTR( pointID, attrname, datbuf )

EOS_PT_READLEVEL - Reads data from the specified fields and records of a single level.
   Result = EOS_PT_READLEVEL( pointID, level, fieldlist, nrec, recs, buffer )

EOS_PT_REGIONINFO - Returns information about a subsetted area of interest for a given fieldlist.
   Result = EOS_PT_REGIONINFO( pointID, regionID, level, fieldlist, size )

EOS_PT_REGIONRECS - Returns the record numbers within a subsetted geographic region for a given level.
   Result = EOS_PT_REGIONRECS( pointID, regionID, level, nrec, recs )

EOS_PT_SIZEOF - Returns information about specified fields in a point regardless of level.
   Result = EOS_PT_SIZEOF( pointID, fieldlist, fldlevel )

EOS_PT_UPDATELEVEL - Updates the specified fields and records of a single level.
   Result = EOS_PT_UPDATELEVEL( pointID, level, fieldlist, nrec, recs, data )

EOS_PT_WRITEATTR - Writes/updates an attribute in a point.
   Result = EOS_PT_WRITEATTR( pointID, attrname, datbuf [, COUNT=value] [, HDF_TYPE=value] )

EOS_PT_WRITELEVEL - Writes (appends) full records to a level.
   Result = EOS_PT_WRITELEVEL( pointID, level, nrec, data )

EOS_QUERY - Returns information about the makeup of an HDF-EOS file.
   Result = EOS_QUERY( Filename, [Info] )

EOS_SW_ATTACH - Attaches to the swath using the swathname parameter as the identifier.
   Result = EOS_SW_ATTACH( fid, swathname )

EOS_SW_ATTRINFO - Returns number type and number of elements of a swath attribute.
   Result = EOS_SW_ATTRINFO( swathID, attrname, numbertype, count )

EOS_SW_CLOSE - Closes the HDF swath file.
   Result = EOS_SW_CLOSE( fid )

EOS_SW_COMPINFO - Returns compression code and compression parameters for a given field.
   Result = EOS_SW_COMPINFO( swathID,fieldname, compcode, compparm )

EOS_SW_CREATE - Creates a swath within the file.
   Result = EOS_SW_CREATE( fid, swathname )

EOS_SW_DEFBOXREGION - Defines a longitude-latitude box region for a swath.
   Result = EOS_SW_DEFBOXREGION( swathID, cornerlon, cornerlat, mode )

EOS_SW_DEFCOMP - Sets HDF field compression for subsequent swath field definitions.
   Result = EOS_SW_DEFCOMP( swathID, compcode [, compparm] )

EOS_SW_DEFDATAFIELD - Defines geolocation fields to be stored in the swath.
   Result = EOS_SW_DEFDATAFIELD( swathID, fieldname, dimlist, numbertype [, /MERGE] )

EOS_SW_DEFDIM - Defines dimensions that are used by the field definition routines to establish the size of the field.
   Result = EOS_SW_DEFDIM( swathID, fieldname, dim )

EOS_SW_DEFDIMMAP - Defines monotonic mapping between the geolocation and data dimensions.
   Result = EOS_SW_DEFDIMMAP( swathID, geodim, datadim, offset, increment )

IDL Quick Reference
EOS_SW_DEFGEOFIELD - Defines geolocation fields to be stored in the swath.
Result = EOS_SW_DEFGEOFIELD(swathID, fieldname, dimlist, numbertype [,/MERGE])

EOS_SW_DEFIDXMAP - Defines mapping between a geolocation and data dimension.
Result = EOS_SW_DEFIDXMAP(swathID, geodim, datadim, index)

EOS_SW_DEFINPERIOD - Defines a time period for a swath.
Result = EOS_SW_DEFINPERIOD(swathID, starttime, stoptime, mode)

EOS_SW_DEFVRTREGION - Subsets along any dimension.
Result = EOS_SW_DEFVRTREGION(swathID, regionID, vertObj, range)

EOS_SW_DETACH - Detaches from the swath interface.
Result = EOS_SW_DETACH(swathID)

EOS_SW_DIMINFO - Retrieves the size of the specified dimension.
Result = EOS_SW_DIMINFO(swathID, dimname)

EOS_SW_DUPEGendregion - Copies information stored in a current region or period to a new region or period.
Result = EOS_SW_DUPEGendregion(regionID)

EOS_SW_EXTRACTPERIOD - Reads data into the data buffer from the subsetted time period.
Result = EOS_SW_EXTRACTPERIOD(swathID, periodID, fieldname, external_mode, buffer)

EOS_SW_EXTRACTREGION - Reads data into the data buffer from the subsetted region.
Result = EOS_SW_EXTRACTREGION(swathID, regionID, fieldname, external_mode, buffer)

EOS_SW_FIELDINFO - Retrieves information on a specific data field.
Result = EOS_SW_FIELDINFO(swathID, fieldname, rank, dims, numbertype, dimlist)

EOS_SW_GETFILLVALUE - Retrieves fill value for given field.
Result = EOS_SW_GETFILLVALUE(swathID, fieldname, fillvalue)

EOS_SW_IDXMAPINFO - Retrieves size of the indexed array and the array of indexed elements of the specified geolocation mapping.
Result = EOS_SW_IDXMAPINFO(swathID, geodim, datadim, index)

EOS_SW_INQATRTRS - Retrieves information about attributes defined in swath.
Result = EOS_SW_INQATRTRS(swathID, attrlist [,LENGTH(OUT)=value])

EOS_SW_INQDATAFIELDS - Retrieves information about all of the data fields defined in swath.
Result = EOS_SW_INQDATAFIELDS(swathID, fieldlist, rank, numbertype)

EOS_SW_INQDIM - Retrieves information about all of the dimensions defined in swath.
Result = EOS_SW_INQDIM(swathID, dimname, dim)

EOS_SW_INQGEOFIELD - Retrieves information about all of the geolocation fields defined in swath.
Result = EOS_SW_INQGEOFIELD(swathID, fieldlist, rank, numbertype)

EOS_SW_INQIDXMAPS - Retrieves information about all indexed geolocation/data mappings in swath.
Result = EOS_SW_INQIDXMAPS(swathID, idxmap, idxsizes)

EOS_SW_INQMAPS - Retrieves information about all non-indexed geolocation relations in swath.
Result = EOS_SW_INQMAPS(swathID, dimmap, offset, increment)

EOS_SW_INQSWATH - Retrieves number and names of swaths defined in HDF-EOS file.
Result = EOS_SW_INQSWATH(filename, swathlist [,LENGTH=value])

EOS_SW_MAPINFO - Retrieves offset and increment of the specified geolocation mapping.
Result = EOS_SW_MAPINFO(swathID, geodim, datadim, offset, increment)

EOS_SW_NENTRIES - Returns number of entries and descriptive string buffer size for specified entity.
Result = EOS_SW_NENTRIES(swathID, entrycode [,LENGTH(OUT)=value])

EOS_SW_OPEN - Opens an existing file, or creates a new file.
Result = EOS_SW_OPEN(filename [,CREATE] [,RDWR | ,READ])

EOS_SW_PERIODINFO - Returns information about a subsetted time period for a given field.
Result = EOS_SW_PERIODINFO(swathID, periodID, fieldname, ntype, rank, dims, size)

EOS_SW_QUERY - Returns information about a specified swath.
Result = EOS_SW_QUERY(Filename, SwathName, [Info])

EOS_SW_READATTR - Reads attribute from a swath field.
Result = EOS_SW_READATTR(swathID, attrname, datbuf)

EOS_SW_READFIELD - Reads data from a swath field.
Result = EOS_SW_READFIELD(swathID, fieldname, buffer [,EDGE=]array [,START=]array [,STRIDE=]array)

EOS_SW_REGIONINFO - Returns information about a subsetted region for a given field.
Result = EOS_SW_REGIONINFO(swathID, regionID, fieldname, ntype, rank, dims, size)

EOS_SW_SETFILLVALUE - Sets fill value for the specified field.
Result = EOS_SW_SETFILLVALUE(swathID, fieldname, fillvalue)

IDL Quick Reference
EOS_SW_WRITEATTR - Writes/updates attribute in a swath.
Result = EOS_SW_WRITEATTR(swathID, attrname, datbuf [, COUNT=value] [, HDF_TYPE=value] )

EOS_SW_WRITEDATAMETA - Writes field metadata for an existing data field.
Result = EOS_SW_WRITEDATAMETA(swathID, fieldname, dimlist, numbertype)

EOS_SW_WRITEFIELD - Writes data to a swath field.
Result = EOS_SW_WRITEFIELD(swathID, fieldname, cut, data [, EDGE=arryray] [, START=array] [, STRIDE=arryray] )

EOS_SW_WRITEGEOMETA - Writes field metadata for an existing geolocation field.
Result = EOS_SW_WRITEGEOMETA(swathID, fieldname, dimlist, numbertype)

HDF Routines

HDF_AN_NUMANN - Returns total number of annotations of a given type.
Result = HDF_AN_NUMANN(ann_id, annot_type, obj_tag, obj_ref)

HDF_AN_GET_TAGREF - Retrieves HDF tag and reference number of annotation.
Result = HDF_AN_GET_TAGREF(an_id, index, annot_type, ann_tag, ann_ref)

HDF_AN_ID2TAGREF - Retrieves HDF tag/reference number pair of annotation.
Result = HDF_AN_ID2TAGREF(ann_id, ann_tag, ann_ref)

HDF_AN_WRITEANN - Writes/updates annotation.
Result = HDF_AN_WRITEANN(ann_id, annot_type, obj_ref, ann_list)

HDF_AN_WRITEATTR - Writes/updates attribute in a swath.
Result = HDF_AN_WRITEATTR(swathID, attrname, datbuf [, COUNT=value] [, HDF_TYPE=value] )

HDF_AN_READANN - Reads specified annotation.
Result = HDF_AN_READANN(ann_id, annotation [, LENGTH=characters] )

HDF_AN_TAGREF2ID - Returns ID of annotation with given tag.
Result = HDF_AN_TAGREF2ID(ann_id, ann_tag)

HDF_AN_TAG2ATYPE - Returns annotation type of corresponding HDF tag.
Result = HDF_AN_TAG2ATYPE(ann_tag)

HDF_AN_NUMANN - Returns total number of annotations of a given type.
Result = HDF_AN_NUMANN(ann_id, annot_type, obj_tag, obj_ref)

HDF_AN_READANN - Reads specified annotation.
Result = HDF_AN_READANN(ann_id, annotation [, LENGTH=characters] )

HDF_AN_SELECT - Obtains identifier of specified annotation.
Result = HDF_AN_SELECT(ann_id, index, annot_type)

HDF_AN_START - Initializes interface for specified file.
Result = HDF_AN_START(file_id)

HDF_AN_TAG2ATYPE - Returns annotation type of corresponding HDF tag.
Result = HDF_AN_TAG2ATYPE(ann_tag)

HDF_AN_TAGREF2ID - Returns ID of annotation with given tag.
Result = HDF_AN_TAGREF2ID(ann_id, ann_tag, ann_ref)

HDF_AN_WRITEANN - Writes annotation text.
Result = HDF_AN_WRITEANN(ann_id, annotation [, LENGTH=characters] )

HDF_BROWSER - See “HDF_BROWSER” on page 41.
HDF_CLOSE - Closes HDF file associated with the given file handle.
HDF_CLOSE, FileHandle

HDF_DELDD - Deletes tag or reference from list of data descriptors.
HDF_DELDD, FileHandle, Tag, Ref

HDF_DF24_ADDIMAGE - Writes 24-bit raster image to HDF file.
HDF_DF24_ADDIMAGE, Filename, Image [], /FORCE_BASELINE{useful only if QUALITY<25}] [], /JPEG | /RLE [ [, QUALITY=value(0 to 100)]

HDF_DF24_GETIMAGE - Reads 24-bit raster image from HDF file.
HDF_DF24_GETIMAGE, Filename, Image [], /LINE | /PIXEL | /PLANE

HDF_DF24_GETINFO - Retrieves information about the current 24-bit HDF image.
HDF_DF24_GETINFO, Filename, Width, Height, Interlace

HDF_DF24_LASTREF - Returns reference number of most recently read or written 24-bit image in an HDF file.
Result = HDF_DF24_LASTREF( )

HDF_DF24_NIMAGES - Returns the number of 24-bit images in an HDF file.
Result = HDF_DF24_NIMAGES(Filename)

HDF_DF24_READREF - Sets reference number of image in an HDF file.
HDF_DF24_READREF, Filename, Refno

HDF_DF24_RESTART - Causes next call to HDF_DF24_GETIMAGE to read first 24-bit image in the HDF file.
HDF_DF24_RESTART
HDF_DFAN_ADDFDS - Adds file description to HDF file.
HDF_DFAN_ADDFDS, Filename, Description

HDF_DFAN_ADDFID - Adds file annotation to HDF file.
HDF_DFAN_ADDFID, Filename, Label

HDF_DFAN_GETDESC - Reads description for given tag and reference number in HDF file.
HDF_DFAN_GETDESC, Filename, Tag, Ref, Description

HDF_DFAN_GETFDS - Reads next available file description.
HDF_DFAN_GETFDS, Filename, Description [], /FIRST

HDF_DFAN_GETFID - Reads next available file annotation.
HDF_DFAN_GETFID, Filename, Label [], /FIRST

HDF_DFAN_GETLABEL - Reads label for given tag-reference pair.
HDF_DFAN_GETLABEL, Filename, Tag, Ref, Label

HDF_DFAN_LABLIST - Retrieves list of reference numbers and labels for given tag.
Result = HDF_DFAN_LABLIST(Filename, Tag, Reflist, Labellist [], LISTSIZE=value] [ , MAXLABEL=value] [ , STARTPOS=value] [, /STRING] )

HDF_DFAN_LASTREF - Returns reference number of most recently read or written annotation.
Result = HDF_DFAN_LASTREF( )

HDF_DFAN_PUTDESC - Writes description for given tag and reference number.
HDF_DFAN_PUTDESC, Filename, Tag, Ref, Description

HDF_DFAN_PUTLABEL - Writes label for given tag and reference number.
HDF_DFAN_PUTLABEL, Filename, Tag, Ref, Label

HDF_DFP_ADDPAL - Appends palette to a HDF file.
HDF_DFP_ADDPAL, Filename, Palette

HDF_DFP_GETPAL - Reads next available palette from HDF file.
HDF_DFP_GETPAL, Filename, Palette

HDF_DFP_LASTREF - Returns reference number of most recently read or written palette in HDF file.
Result = HDF_DFP_LASTREF( )

HDF_DFP_NPALS - Returns number of palettes present in HDF file.
Result = HDF_DFP_NPALS(Filename)

HDF_DFP_RESTART - Causes next call to HDF_DFP_GETPAL to read from the first palette in HDF file.
HDF_DFP_RESTART

HDF_DFP_WRITEREF - Sets reference number for next palette to be written to a HDF file.
HDF_DFP_WRITEREF, Filename, Refno

HDF_DFR8_ADDIMAGE - Appends 8-bit raster image to the specified HDF file.
HDF_DFR8_ADDIMAGE, Filename, Image

HDF_DFR8_READREF - Reads next available 8-bit image in HDF file.
Result = HDF_DFR8_READREF(Filename)

HDF_DFR8_SETPALETTE - Sets current palette to be used for subsequent images in a HDF file.
HDF_DFR8_SETPALETTE, Palette

HDF_DUPDD - Generates new references to existing data in HDF file.
HDF_DUPDD, FileHandle, NewTag, NewRef, OldTag, OldRef

HDF_EXISTS - Returns True if HDF format library is supported on the current IDL platform.
Result = HDF_EXISTS( )

HDF_GR_ATTRIBINFO - Retrieves information about specified HDF data object.
Result = HDF_GR_ATTRIBINFO(obj_id, attr_index, name, data_type, count)

HDF_GR_CREATE - Creates HDF GR raster image.
Result = HDF_GR_CREATE(gr_id, name, ncomp, data_type, interlace_mode, dim_sizes)

IDL Quick Reference
IDL Quick Reference
HDF_SD_ENDACCESS - Closes SD dataset interface.
HDF_SD_ENDACCESS, SD_ID

HDF_SD_FILEINFO - Retrieves the number of datasets and global attributes in HDF file.
HDF_SD_FILEINFO, SD_ID, Datasets, Attributes

HDF_SD_GETDATA - Retrieves a hyperslab of values from SD dataset.
HDF_SD_GETDATA, SDS_ID, Data [, COUNT=vector] [, /NOREVERSE] [, START=vector] [, STRIDE=vector]

HDF_SD_GETINFO - Retrieves information about SD dataset.

HDF_SD_IDTOREF - Converts SD data set ID into SD data set reference number.
Result = HDF_SD_IDTOREF(SD_ID)

HDF_SD_ISCOORDVAR - Determines whether supplied dataset ID represents NetCDF “coordinate” variable.
Result = HDF_SD_ISCOORDVAR(SDS_ID)

HDF_SD_NAMETOINDEX - Returns SD dataset index given its name and SD interface ID.
Result = HDF_SD_NAMETOINDEX(SD_ID, SDS_Name)

HDF_SD_REFTOINDEX - Returns SD dataset index given its reference number and SD interface ID.
Result = HDF_SD_REFTOINDEX(SD_ID, Refno)

HDF_SD_SELECT - Returns SD dataset ID.
Result = HDF_SD_SELECT(SD_ID, Number)

HDF_SD_SETCOMPRESS - Compresses an existing HDF SD dataset or sets the compression method of a new HDF SD dataset.
HDF_SD_SETCOMPRESS, SDS_ID, comptype [, EFFORT=integer [1 to 9]]

HDF_SD_SETTEXTFILE - Moves data values from a dataset into an external file.
HDF_SD_SETTEXTFILE, SDS_ID, Filename [, OFFSET=bytes]

HDF_SD_SETCOMPRESS - Sets information about SD dataset.
HDF_SD_SETCOMPRESS, SDS_ID [, FILL=variable] [, FORMAT=string] [, LABEL=string] [, RANGE=max, min] [, UNIT=string] [, COORDSYS=string] [, CALDATA=structure]

HDF_SD_SETSTART - Opens or creates HDF file and initializes SD interface.
Result = HDF_SD_SETSTART( Filename [, /READ] [, /RDWR] [, /CREATE])

HDF_SD_READ - See “HDF_READ” on page 41.

HDF_SD_ADDDDATA - Writes hyperslab of values to an SD dataset.
HDF_SD_ADDDDATA, SDS_ID, Data [, COUNT=vector] [, /NOREVERSE] [, START=vector] [, STRIDE=vector]

HDF_SD_ATTRFIND - Locates index of HDF attribute given its name.
Result = HDF_SD_ATTRFIND(S_ID, Name)

HDF_SDATTRINFO - Reads or retrieves information about SD attribute.
HDF_SDATTRINFO, S_ID, Attr_Index [, COUNT=variable] [, DATA=variable] [, HDF_TYPE=variable] [, NAME=variable] [, TYPE=variable]

HDF_SD_ATTRSET - Writes attributes to an open HDF SD dataset.

HDF_SD_DIMGET - Retrieves info. about SD dataset dimension.

HDF_SD_DIMGETID - Returns dimension ID given a dataset “SDS_ID” and dimension number.
Result = HDF_SD_DIMGETID(SDS_ID, Dimension_Number)

HDF_SD_DIMSET - Sets scale and data strings for SD dimension.

HDF_SD_END - Closes SD interface to an HDF file.
HDF_SD_END, SD_ID

HDF_SD_ENDACCESS - Closes SD dataset interface.

HDF_SD_REFTOINDEX(SD_ID)

HDF_SDATTRINFO(S_ID, Name)

HDF_SDATTRSET(S_ID, Attr_Name)

HDF_SD_DIMGET(SDS_ID, Dimension_Number)

HDF_SD_DIMSET(Dims, Attr_Name)

HDF_SD_END(SD_ID)

HDF PACKDATA - Packs a set IDL variable into an array of raw byte data.

HDF_SD_ADDDDATA(data1 [, data2 [, data3 [, data4 [, data5 [, data6 [, data7 [, data8]]]]]]])

HDF_SD_ATTRFIND(S_ID)

HDF_SD_ATTRSET(Attr_Name)

HDF_SD_DIMGET(Dim_ID)

HDF_SD_END(SD_ID)

HDF_SD_READ()

HDF_SD_ADDDDATA()

HDF_SD_ATTRFIND()

HDF_SD_ATTRSET()

HDF_SD_DIMGET()

HDF_SD_END()
HDF_VD_ATTRSET - Writes a vdata attribute or a vdata field attribute to the currently attached HDF VData structure.

HDF_VD_ATTACH - Accesses a VData with the given ID.

HDF_VD_ATTRFIND - Returns an attribute's index number given the name of an attribute.

HDF_VD_ATTRINFO - Retrieves information about a VData attribute.

HDF_VD_ATTRINFO, VData, FieldID, AttrID, Values
[. COUNT=variable] [. DATA=variable]
[. HDF_TYPE=variable] [. NAME=variable]
[. TYPE=variable]

HDF_VD_ATTRSET - Writes a vdata attribute or a vdata field attribute to the currently attached HDF VData structure.

HDF_VD_ATTACH, VData, FieldID, Attr_Name, Values
[. Count] [. BYTE=variable] [. DFNT_CHAR8]
[. FLOAT=variable] [. DFNT_FLOAT32]
[. INT=variable] [. DFNT_INT16]
[. UINT=variable] [. DFNT_UINT16]
[. LONG=variable] [. DFNT_INT32]
[. ULONG=variable] [. DFNT_UINT32]

HDF_VD_DETACH - Called when done accessing a VData.

HDF_VD_DETACH, VData

HDF_VD_FDEFINE - Adds new field specification for VData.

HDF_VD_FDEFINE, VData, FieldName [. BYTE=variable]
[. DFNT_INT16=variable]
[. DFNT_UINT16=variable]
[. DFNT_FLOAT16=variable]

HDF_VD_FEXIST - Returns true if specified fields exist in HDF file.

HDF_VD_FEXIST(VData, Fieldnames)

HDF_VD_FIND - Returns reference number of specified VData.

HDF_VD_FIND( FileHandle, Name)

HDF_VD_GET - Returns information about a VData.

HDF_VD_GET, VData [, . CLASS=variable]
[. COUNT=variable] [. FIELDS=variable]
[. INTERLACE=variable] [. NAME=variable]
[. NFIELDS=variable] [. REF=variable]
[. SIZE=variable] [. TAG=variable]

HDF_VD_GETID - Returns VData reference number for next VData.

HDF_VD_GETID( FileHandle, VDataSet)

HDF_VD_GETINFO - Returns information about each Vdata field.

HDF_VD_GETINFO, VData, Index [. NAME=variable]
[. ORDER=variable] [. SIZE=variable]
[. TYPE=variable]

HDF_VD_INSERT - Adds VData or VGroup to contents of VGroup.

HDF_VD_INSERT, VGroup, VData(or Vgroup), Position=variable

HDF_VD_ISATTR - Returns True (1) if the VData is storing an attribute, False (0) otherwise.

HDF_VD_ISATTR(VData)

HDF_VD_ISVD - Returns True (1) if an object is a VData.

HDF_VD_ISVD(VGroup, Id)

HDF_VD_ISVG - Returns True (1) if an object is a VGroup.

HDF_VD_ISVG(VGroup, Id)

HDF_VD_LONE - Returns array containing all VDatas that are not contained in another VData.

HDF_VD_LONE( FileHandle [. MAXSIZE=value] )

HDF_VD_NATTRS - Returns the number of attributes associated with the specified VData.

HDF_VD_NATTRS( VData, FieldId )

HDF_VD_READ - Reads data from a VData.

HDF_VD_READ( VData, Data [. FIELDS=string] [. FULL_INTERLACE=variable]
[. NO_INTERLACE=variable] [. NRECORDS=records] )

HDF_VD_SEEK - Moves read pointer in specified VData to specific record number.

HDF_VD_SEEK, VData, Record

HDF_VD_SETINFO - Specifies general information about a VData.

HDF_VD_SETINFO, VData [. CLASS=string]
[. FULL_INTERLACE=variable] [. NO_INTERLACE=variable]
[. NAME=string]

HDF_VD_WRITE - Stores data in a VData.

HDF_VD_WRITE, VData, Fields, Data [. FULL_INTERLACE=variable]
[. NO_INTERLACE=variable] [. NRECORDS=records]

HDF_VG_ADDTR - Adds tag and reference to specified VGroup.

HDF_VG_ADDTR, VGroup, Tag, Ref

HDF_VG_ATTACH - Attaches (opens) a VGroup.

HDF_VG_ATTACH( FileHandle, VGroupId [. READ=variable] [. WRITE=variable] )

HDF_VG_DETACH - Called when finished accessing a VGroup.

HDF_VG_DETACH, VGroup

HDF_VG_GETID - Returns VGroup ID for specified VGroup.

HDF_VG_GETID( FileHandle, VGroupId)
HDF_VG_GETINFO - Returns information about a VGroup.
   HDF_VG_GETINFO, VGroup [, , CLASS=variable] 
   [, NAME=variable] [, NENTRIES=variable] 
   [, REF=variable] [, TAG=variable]

HDF_VG_GETNEXT - Returns reference number of the next object 
in a VGroup.
   Result = HDF_VG_GETNEXT(VGroup, Id)

HDF_VG_GETTR - Returns tag/reference pair at specified position 
within a VGroup.
   HDF_VG_GETTR, VGroup, Index, Tags, Refs

HDF_VG_GETTRS - Returns tag/reference pairs of HDF file objects 
belonging to the specified VGroup.
   HDF_VG_GETTRS, VGroup, Tags, Refs 
   [, MAXSIZE=value]

HDF_VG_INQTR - Returns true if specified tag/reference pair is 
linked to the specified Vgroup.
   Result = HDF_VG_INQTR(VGroup, Tag, Ref)

HDF_VG_INSERT - Adds VData or VGroup to contents of VGroup.
   HDF_VG_INSERT, VGroup, VData(or 
   VGroup)[, POSITION=variable]

HDF_VG_ISVD - Returns true if object is a VData.
   Result = HDF_VG_ISVD(VGroup, Id)

HDF_VG_ISVG - Returns true if object is a VGroup.
   Result = HDF_VG_ISVG(VGroup)

HDF_VG_LONE - Returns array containing Ids of all VGroups that 
are not contained in another VGroup.
   Result = HDF_VG_LONE( [FileHandle 
   [, MAXSIZE=value] )

HDF_VG_NUMBER - Returns number of HDF file objects in specified 
VGroup.
   Result = HDF_VG_NUMBER(VGroup)

HDF_VG_SETINFO - Sets the name and class of a VGroup.
   HDF_VG_SETINFO, VGroup [, CLASSNAME=string] 
   [, NAME=variable]

HDF5 Routines

H5_BROWSER - Presents a graphical user interface for viewing and 
reading HDF5 files.
   Result = H5_BROWSER([Files [, /DIALOG_READ] ])

H5_CLOSE - Flushes all data to disk, closes file identifiers, and cleans 
up memory.
   H5_CLOSE

H5_GET_LIBVERSION - Returns the current version of the HDF5 
library used by IDL.
   Result = H5_GET_LIBVERSION( )

H5_OPEN - Initializes IDL’s HDF5 library.
   H5_OPEN

H5_PARSE - Recursively descends through an HDF5 file or group and 
creates an IDL structure containing object information and data.
   Result = H5_PARSE( File [, /READ_DATA] ) 
   or
   Result = H5_PARSE( Loc_id, Name [, FILE=string] 
   [, PATH=string] [, /READ_DATA] )

H5A_CLOSE - Closes the specified attribute and releases resources 
used by it.
   H5A_CLOSE, Attribute_id

H5A_GET_NAME - Retrieves an attribute name given the attribute 
identifier number.
   Result = H5A_GET_NAME(Attribute_id)

H5A_GET_NUM_ATTRS - Returns the number of attributes 
attached to a group, dataset, or a named datatype.
   Result = H5A_GET_NUM_ATTRS(Loc_id)

H5A_GET_SPACE - Returns the identifier number of a copy of the 
dataspace for an attribute.
   Result = H5A_GET_SPACE(Attribute_id)

H5A_GET_TYPE - Returns the identifier number of a copy of the 
datatype for an attribute.
   Result = H5A_GET_TYPE(Attribute_id)

H5A_OPEN_IDX - Opens an existing attribute by the index of that 
attribute.
   Result = H5A_OPEN_IDX(Loc_id, Index)

H5A_OPEN_NAME - Opens an existing attribute by the name of 
that attribute.
   Result = H5A_OPEN_NAME(Loc_id, Name)

H5A_READ - Reads the data within an attribute, converting from the 
HDF5 file datatype into the HDF5 memory datatype, and finally 
into the corresponding IDL datatype.
   Result = H5A_READ(Attribute_id)

H5D_CLOSE - Closes the specified dataset and releases its used 
routines.
   H5D_CLOSE, Dataset_id

H5D_GET_SPACE - Returns an identifier number for a copy of the 
dataspace for a dataset.
   Result = H5D_GET_SPACE(Dataset_id)

H5D_GET_STORAGE_SIZE - Returns the amount of storage in 
bytes required for a dataset.
   Result = H5D_GET_STORAGE_SIZE(Dataset_id)

H5D_GET_TYPE - Returns an identifier number for a copy of the 
datatype for a dataset.
   Result = H5D_GET_TYPE(Dataset_id)

H5D_OPEN - Opens an existing dataset within an HDF5 file.
   Result = H5D_OPEN(Loc_id, Name)

H5D_READ - Reads the data within a dataset, converting from the 
HDF5 file datatype into the HDF5 memory datatype, and finally 
into the corresponding IDL datatype.
   Result = H5D_READ(Dataset_id [, FILE_SPACE=id] 
   [, MEMORY_SPACE=id] )
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>H5F_CLOSE</td>
<td>Closes the specified file and releases resources used by it.</td>
<td>Result = H5F_CLOSE(File_id)</td>
</tr>
<tr>
<td>H5F_IS_HDF5</td>
<td>Determines if a file is in the HDF5 format.</td>
<td>Result = H5F_IS_HDF5(Filename)</td>
</tr>
<tr>
<td>H5F_OPEN</td>
<td>Opens an existing HDF5 file.</td>
<td>Result = H5F_OPEN(Filename)</td>
</tr>
<tr>
<td>H5G_CLOSE</td>
<td>Closes the specified group and releases resources used by it.</td>
<td>Result = H5G_CLOSE(Group_id)</td>
</tr>
<tr>
<td>H5G_GET_COMMENT</td>
<td>Retrieves a comment string from a specified object.</td>
<td>Result = H5G_GET_COMMENT(Loc_id, Name)</td>
</tr>
<tr>
<td>H5G_GET_LINKVAL</td>
<td>Returns the name of the object pointed to by a symbolic link.</td>
<td>Result = H5G_GET_LINKVAL(Loc_id, Name)</td>
</tr>
<tr>
<td>H5G_GET_MEMBER_NAME</td>
<td>Retrieves the name of an object within a group, by its zero-based index.</td>
<td>Result = H5G_GET_MEMBER_NAME(Loc_id, Name, Index)</td>
</tr>
<tr>
<td>H5G_GET_NMEMBERS</td>
<td>Returns the number of objects within a group.</td>
<td>Result = H5G_GET_NMEMBERS(Loc_id, Name)</td>
</tr>
<tr>
<td>H5G_GET_OBJINFO</td>
<td>Retrieves information from a specified object.</td>
<td>Result = H5G_GET_OBJINFO(Loc_id, Name [, /FOLLOW_LINK])</td>
</tr>
<tr>
<td>H5G_OPEN</td>
<td>Opens an existing group within an HDF5 file.</td>
<td>Result = H5G_OPEN(Loc_id, Name)</td>
</tr>
<tr>
<td>H5I_GET_TYPE</td>
<td>Returns the object's type.</td>
<td>Result = H5I_GET_TYPE(Obj_id)</td>
</tr>
<tr>
<td>H5R_DEREFERENCE</td>
<td>Opens a reference and returns the object identifier.</td>
<td>Result = H5R_DEREFERENCE(Loc_id, Reference)</td>
</tr>
<tr>
<td>H5R_GET_OBJECT_TYPE</td>
<td>Returns the type of object that an object reference points to.</td>
<td>Result = H5R_GET_OBJECT_TYPE(Loc_id, Reference)</td>
</tr>
<tr>
<td>H5S_CLOSE</td>
<td>Releases and terminates access to a dataspace.</td>
<td>Result = H5S_CLOSE(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_COPY</td>
<td>Copies an existing dataspace.</td>
<td>Result = H5S_COPY(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_CREATE_SIMPLE</td>
<td>Creates a simple dataspace.</td>
<td>Result = H5S_CREATE_SIMPLE(Dimensions [, MAX_DIMENSIONS=vector])</td>
</tr>
<tr>
<td>H5S_GET_SELECT_BOUNDS</td>
<td>Retrieves the coordinates of the bounding box containing the current dataspace selection.</td>
<td>Result = H5S_GET_SELECT_BOUNDS(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_GET_SELECT_ELEM_NPOINTS</td>
<td>Determines the number of element points in the current dataspace selection.</td>
<td>Result = H5S_GET_SELECT_ELEM_NPOINTS(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_GET_SELECT_ELEM_POINTLIST</td>
<td>Returns a list of the element points in the current dataspace selection.</td>
<td>Result = H5S_GET_SELECT_ELEM_POINTLIST(Dataspace_id [, START=value] [, NUMBER=value])</td>
</tr>
<tr>
<td>H5S_GET_SELECT_HYPER_BLOCKLIST</td>
<td>Returns a list of the hyperslab blocks in the current dataspace selection.</td>
<td>Result = H5S_GET_SELECT_HYPER_BLOCKLIST(Dataspace_id [, START=value] [, NUMBER=value])</td>
</tr>
<tr>
<td>H5S_GET_SELECT_HYPER_NBLOCKS</td>
<td>Determines the number of hyperslab blocks in the current dataspace selection.</td>
<td>Result = H5S_GET_SELECT_HYPER_NBLOCKS(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_GET_SELECT_NPOINTS</td>
<td>Determines the number of elements in a dataspace selection.</td>
<td>Result = H5S_GET_SELECT_NPOINTS(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_GET_SIMPLE_EXTENT_DIMS</td>
<td>Returns the dimension sizes for a dataspace.</td>
<td>Result = H5S_GET_SIMPLE_EXTENT_DIMS(Dataspace_id [, MAX_DIMENSIONS=variable])</td>
</tr>
<tr>
<td>H5S_GET_SIMPLE_EXTENT_NDIMS</td>
<td>Determines the number of dimensions (or rank) of a dataspace.</td>
<td>Result = H5S_GET_SIMPLE_EXTENT_NDIMS(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_GET_SIMPLE_EXTENT_NPOINTS</td>
<td>Determines the number of elements in a dataspace.</td>
<td>Result = H5S_GET_SIMPLE_EXTENT_NPOINTS(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_GET_SIMPLE_EXTENT_TYPE</td>
<td>Returns the current class of a dataspace.</td>
<td>Result = H5S_GET_SIMPLE_EXTENT_TYPE(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_IS_SIMPLE</td>
<td>Determines whether a dataspace is a simple dataspace.</td>
<td>Result = H5S_IS_SIMPLE(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_OFFSET_SIMPLE</td>
<td>Sets the selection offset for a simple dataspace.</td>
<td>Result = H5S_OFFSET_SIMPLE(Dataspace_id, Offset)</td>
</tr>
<tr>
<td>H5S_SELECT_ALL</td>
<td>Selects the entire extent of a dataspace.</td>
<td>Result = H5S_SELECT_ALL(Dataspace_id)</td>
</tr>
<tr>
<td>H5S_SELECT_ELEMENTS</td>
<td>Selects array elements to be included in the selection for a dataspace.</td>
<td>Result = H5S_SELECT_ELEMENTS(Dataspace_id, Coordinates, /RESET)</td>
</tr>
</tbody>
</table>
H5S_SELECT_HYPERSLAB - Selects a hyperslab region to be included in the selection for a dataspace.

H5S_SELECT_NONE - Resets the dataspace selection region to include no elements.
H5S_SELECT_NONE, Dataspac_e_id

H5S_SELECT_VALID - Verifies that the selection is within the extent of a dataspace.
Result = H5S_SELECT_VALID(Dataspac_e_id)

H5T_CLOSE - Releases the specified datatype’s identifier and releases resources used by it.
H5T_CLOSE, Datatype_id

H5T_COMMITTED - Determines whether a datatype is a named datatype or a transient type.
Result = H5T_COMMITTED(Datatype_id)

H5T_COPY - Copies an existing datatype.
Result = H5T_COPY(Datatype_id)

H5T_EQUAL - Determines whether two datatype identifiers refer to the same datatype.
Result = H5T_EQUAL(Datatype_id1, Datatype_id2)

H5T_GET_ARRAY_DIMS - Returns the dimension sizes for an array datatype object.
Result = H5T_GET_ARRAY_DIMS(Datatype_id [, PERMUTATIONS=variable] )

H5T_GET_ARRAY_NDIMS - Determines the number of dimensions (or rank) of an array datatype object.
Result = H5T_GET_ARRAY_NDIMS(Datatype_id)

H5T_GET_CLASS - Returns the datatype’s class.
Result = H5T_GET_CLASS(Datatype_id)

H5T_GET_CSET - Returns the character set type of a string datatype.
Result = H5T_GET_CSET(Datatype_id)

H5T_GET_EBIAS - Returns the exponent bias of a floating-point type.
Result = H5T_GET_EBIAS(Datatype_id)

H5T_GET_FIELDS - Retrieves information about the positions and sizes of bit fields within a floating-point datatype.
Result = H5T_GET_FIELDS(Datatype_id)

H5T_GET_INPAD - Returns the padding method for unused internal bits within a floating-point datatype.
Result = H5T_GET_INPAD(Datatype_id)

H5T_GET_MEMBER_CLASS - Returns the datatype class of a compound datatype member.
Result = H5T_GET_MEMBER_CLASS(Datatype_id, Member)

H5T_GET_MEMBER_NAME - Returns the datatype name of a compound datatype member.
Result = H5T_GET_MEMBER_NAME(Datatype_id, Member)

H5T_GET_MEMBER_OFFSET - Returns the byte offset of a field within a compound datatype.
Result = H5T_GET_MEMBER_OFFSET(Datatype_id, Member)

H5T_GET_MEMBER_TYPE - Returns the datatype identifier for a specified member within a compound datatype.
Result = H5T_GET_MEMBER_TYPE(Datatype_id, Member)

H5T_GET_NMEMBERS - Returns the number of fields in a compound datatype.
Result = H5T_GET_NMEMBERS(Datatype_id)

H5T_GET_ORDER - Returns the byte order of an atomic datatype.
Result = H5T_GET_ORDER(Datatype_id)

H5T_GET_PAD - Returns the padding method of the least significant bit (lsb) and most significant bit (msb) of an atomic datatype.
Result = H5T_GET_PAD(Datatype_id)

H5T_GET_PRECISION - Returns the precision in bits of an atomic datatype.
Result = H5T_GET_PRECISION(Datatype_id)

H5T_GET_SIGN - Returns the sign type for an integer datatype.
Result = H5T_GET_SIGN(Datatype_id)

H5T_GET_SIZE - Returns the size of a datatype in bytes.
Result = H5T_GET_SIZE(Datatype_id)

H5T_GET_STRPAD - Returns the padding method for a string datatype.
Result = H5T_GET_STRPAD(Datatype_id)

H5T_GET_SUPER - Returns the base datatype from which a datatype is derived.
Result = H5T_GET_SUPER(Datatype_id)

H5T_IDLTYPE - Returns the IDL type code corresponding to a datatype.
Result = H5T_IDLTYPE(Datatype_id [, ARRAY_DIMENSIONS=variable] [, STRUCTURE=variable] )

H5T_MEMTYPE - Returns the native memory datatype corresponding to a file datatype.
Result = H5T_MEMTYPE(Datatype_id)

H5T_OPEN - Opens a named datatype.
Result = H5T_OPEN(Loc_id, Name)
NetCDF Routines

NCDF_ATTCOPY - Copies attribute from one netCDF file to another.
Result = NCDF_ATTCOPY( Incdf [, Invar | , /IN_GLOBAL] , Name, Outcdf [, Outvar] [, /OUT_GLOBAL] )

NCDF_ATTDEL - Deletes an attribute from a netCDF file.
NCDF_ATTDEL, Cdfid [, Varid | , /GLOBAL] , Name

NCDF_ATTGET - Retrieves value of an attribute from a netCDF file.
NCDF_ATTGET, Cdfid [, Varid | , /GLOBAL] , Name, Value

NCDF_ATTGET1 - Retrieves one element from a netCDF variable.
NCDF_ATTGET1, Cdfid, Varid, Value [, OFFSET=vector] [, /BYTE | , /CHAR | , /DOUBLE | , /FLOAT | , /LONG | , /SHORT]

NCDF_ATTINQ - Returns information about a netCDF attribute.
Result = NCDF_ATTINQ( Cdfid [, Varid | , /GLOBAL] , Name )

NCDF_ATTNAME - Returns the name of an attribute given its ID.
Result = NCDF_ATTNAME( Cdfid [, Varid | , /GLOBAL] , Attnum )

NCDF_ATTPUT - Creates an attribute in a netCDF file.
NCDF_ATTPUT, Cdfid [, Varid | , /GLOBAL] , Name, Value [, LENGTH=value] [, /BYTE | , /CHAR | , /DOUBLE | , /FLOAT | , /LONG | , /SHORT]

NCDF_ATTRENAME - Renames an attribute in a netCDF file that has been opened for writing.
NCDF_ATTRENAME, Cdfid [, Varid | , /GLOBAL] , Oldname, Newname

NCDF_CLOSE - Closes an open netCDF file.
NCDF_CLOSE, Cdfid

NCDF_CONTROL - Performs miscellaneous netCDF operations.

NCDF_CREATE - Creates a new netCDF file.
Result = NCDF_CREATE( Filename [, /CLOBBER | , /NOCLOBBER] )

NCDF_DIMDEF - Defines a dimension given its name and size.
Result = NCDF_DIMDEF( Cdfid, DimName, Size [, /UNLIMITED] )

NCDF_DIMID - Returns the ID of a netCDF dimension, given the name of the dimension.
Result = NCDF_DIMID( Cdfid, DimName )

NCDF_DIMINO - Returns the name and size of a dimension in a netCDF file, given its ID.
NCDF_DIMINO, Cdfid, Dimid, Name, Size

NCDF_DIMRENAME - Renames an existing dimension in a netCDF file that has been opened for writing.
NCDF_DIMRENAME, Cdfid, Dimid, NewName

NCDF_EXISTS - Returns True if the netCDF format library is supported on the current IDL platform.
Result = NCDF_EXISTS( )

NCDF_INQUIRE - Returns information about an open netCDF file.
Result = NCDF_INQUIRE(Cdfid)

NCDF_OPEN - Opens an existing netCDF file.
Result = NCDF_OPEN( Filename [, /NOWRITE | , /WRITE] )

NCDF_VARDEF - Adds a new variable to an open netCDF file in define mode.
Result = NCDF_VARDEF( Cdfid, Name [, Dim] [, /BYTE | , /CHAR | , /DOUBLE | , /FLOAT | , /LONG | , /SHORT] )

NCDF_VARGET - Retrieves a hyperslab of values from a netCDF variable.
NCDF_VARGET, Cdfid, Varid, Value [, COUNT=vector] [, OFFSET=vector] [, STRIDE=vector]

NCDF_VARGET1 - Retrieves one element from a netCDF variable.
NCDF_VARGET1, Cdfid, Varid, Value [, OFFSET=vector]

NCDF_VARID - Returns the ID of a netCDF variable.
Result = NCDF_VARID(Cdfid, Name)

NCDF_VARINQ - Returns information about a netCDF variable, given its ID.
Result = NCDF_VARINQ(Cdfid, Varid)

NCDF_VARPUT - Writes a hyperslab of values to a netCDF variable.
NCDF_VARPUT, Cdfid, Varid, Value [, COUNT=vector] [, OFFSET=vector] [, STRIDE=vector]

NCDF_VARRENAME - Renames a netCDF variable.
NCDF_VARRENAME, Cdfid, Varid, Name