**Restricted Rights Notice**

The IDL®, ION Script™, ION Java™, IDL Analyst™, ENVI®, and ENVI Zoom™ 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. ITT Visual Information Solutions reserves the right to make changes to this document at any time and without notice.

**Limitation of Warranty**

ITT Visual Information Solutions 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.

ITT Visual Information Solutions shall not be liable for any direct, consequential, or other damages suffered by the Licensee or any others resulting from use of the software packages or their documentation.

**Permission to Reproduce this Manual**

If you are a licensed user of these products, ITT Visual Information Solutions 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.

**Export Control Information**

This software and its associated documentation are subject to the controls of the Export Administration Regulations (EAR). It has been determined that this software is classified as EAR99 under U.S. Export Control laws and regulations, and may not be re-transferred to any destination expressly prohibited by U.S. laws and regulations. The recipient is responsible for ensuring compliance to all applicable U.S. Export Control laws and regulations.

**Acknowledgments**

ENVI® and IDL® are registered trademarks of ITT Corporation, registered in the United States Patent and Trademark Office. ION™, ION Script™, ION Java™, and ENVI Zoom™ are trademarks of ITT Visual Information Solutions.

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.


SMACC. Copyright © 2000–2004, Spectral Sciences, Inc. and ITT Visual Information Solutions. All rights reserved.

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.

BandMax®, Copyright © 2003, The Galileo Group Inc.

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

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

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


FLAASH is licensed from Spectral Sciences, Inc. under a U.S. Patent Pending.

Portions of this software are copyrighted by Merge Technologies Incorporated.

Support Vector Machine (SVM) is based on the LIBSVM library written by Chih-Chung Chang and Chih-Jen Lin (http://www.csie.ntu.edu.tw/~cjlin/libsvm), adapted by ITT Visual Information Solutions for remote sensing image supervised classification purposes.

IDL Wavelet Toolkit Copyright © 2002, Christopher Torrence.

IMSL is a trademark of Visual Numerics, Inc. Copyright © 1970–2006 by Visual Numerics, Inc. All Rights Reserved.

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

Chapter 1:
Functional List of IDL Routines ................................................................. 5

Chapter 2:
Alphabetical List of IDL Routines ............................................................... 27

Chapter 3:
Scientific Data Formats ............................................................................. 135
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>6</td>
</tr>
<tr>
<td>Animation</td>
<td>7</td>
</tr>
<tr>
<td>Array Creation</td>
<td>7</td>
</tr>
<tr>
<td>Array Manipulation</td>
<td>8</td>
</tr>
<tr>
<td>Color Table Manipulation</td>
<td>8</td>
</tr>
<tr>
<td>Date and Time</td>
<td>8</td>
</tr>
<tr>
<td>Debugging</td>
<td>8</td>
</tr>
<tr>
<td>Dialog Routines</td>
<td>8</td>
</tr>
<tr>
<td>Direct Graphics</td>
<td>9</td>
</tr>
<tr>
<td>Error Handling</td>
<td>9</td>
</tr>
<tr>
<td>Executive Commands</td>
<td>10</td>
</tr>
<tr>
<td>External Linking</td>
<td>10</td>
</tr>
<tr>
<td>Font Manipulation</td>
<td>10</td>
</tr>
<tr>
<td>Help Routines</td>
<td>10</td>
</tr>
<tr>
<td>Image Processing</td>
<td>10</td>
</tr>
<tr>
<td>Input/Output</td>
<td>12</td>
</tr>
<tr>
<td>Language Catalogs</td>
<td>13</td>
</tr>
<tr>
<td>Mapping</td>
<td>13</td>
</tr>
<tr>
<td>Mathematics</td>
<td>14</td>
</tr>
<tr>
<td>Object Class Library</td>
<td>17</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>22</td>
</tr>
<tr>
<td>Scientific Data Formats</td>
<td>22</td>
</tr>
<tr>
<td>Scope Functions</td>
<td>22</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>22</td>
</tr>
<tr>
<td>Statements</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>24</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>25</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.
IDLgrLight - Represents a source of illumination for three-dimensional graphic objects.
IDLgrModel - Represents a graphical item or group of items that can be transformed (rotated, scaled, and/or translated).
IDLgrScene - Represents the entire scene to be drawn and serves as a container of IDLgrView or IDLgrViewgroup objects.
IDLgrView - Represents a rectangular area in which graphics objects are drawn.
IDLgrViewgroup - Represents a simple container object, very similar to the IDLgrScene object, but can contain objects without a Draw method in addition to IDLgrView objects.
IDLgrWindow - Represents an on-screen area on a display device that serves as a graphics destination.
SCALE3 - 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.
IDLgrPolygon - Represents one or more polygons that share a given set of vertices and rendering attributes.
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.
IDLgrContour - Draws a contour plot from data stored in a rectangular array or from a set of unstructured points.
IDLgrSurface - Represents a shaded or vector representation of a mesh grid.
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

IDLgrTessellator - Decomposes a polygon description into a set of triangles to convert complex and/or concave polygons into a convex form suitable for drawing with the IDLgrPolygon object.
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.
IVECTOR - Creates an iTool and associated user interface (UI) configured to display and manipulate vector data.
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.
- `IDLgrVRML` - Saves the contents of an Object Graphics hierarchy into a VRML 2.0 format file.
- `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.
- `RECON3D` - Reconstructs a 3D representation of an object from 2D images.
- `SEARCH3D` - Finds “objects” or regions of similar data values within a volume.
- `SHADE_VOLUME` - Contours a volume to create a list of vertices and polygons that can be displayed using POLYSHADE.
- `SLICER3` - Interactive volume visualization tool.
- `VOXEL_PROJ` - Generates volume visualizations using voxel technique.
- `XOBJVIEW` - 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.
- `XVOLUME` - Utility for viewing and interactively manipulating volumes and isosurfaces.

**Animation**

- `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.
- `IDLimiMJPEG2000` - Provides the ability to read and write Motion JPEG2000 (MJ2) files.
- `IDLgrModel` - Animates the display of objects by displaying a single object in the collection of objects contained in the model when the RENDER_METHOD property is set.
- `IDLgrMPEG` - Creates an MPEG movie file from an array of image frames.
- `FLICK` - Causes the display to flicker between two images.
- `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.
- `XINTERANIMATE` - Displays animated sequence of images.

**Array Creation**

- `BINDGEN` - Returns byte array with each element set to its subscript.
- `BYTEARR` - Creates a byte vector or array.
- `CINDGEN` - Returns a complex array with each element set to its subscript.
- `COMPLEXARR` - Creates a complex, single-precision, floating-point vector or array.
- `DBLARR` - Creates a double-precision array.
- `DCINDGEN` - Returns a double-precision, complex array with each element set to its subscript.
- `DCOMPLEXARR` - Creates a complex, double-precision vector or array.
- `DINDGEN` - Returns a double-precision array with each element set to its subscript.
- `FINDGEN` - Returns a floating-point array with each element set to its subscript.
- `FLOATARR` - Returns single-precision, floating-point vector or array.
- `IDENTITY` - Returns an identity array (an array with ones along the main diagonal and zeros elsewhere) of the specified dimensions.
- `INDGEN` - Returns an integer array with each element set to its subscript.
- `INTARR` - Creates an integer vector or array.
- `L64INDGEN` - Returns a 64-bit integer array with each element set to its subscript.
- `LINDGEN` - Returns a longword integer array with each element set to its subscript.
- `LONG64ARR` - Returns a 64-bit integer vector or array.
- `LONARR` - Returns a longword integer vector or array.
- `MAKE_ARRAY` - Returns an array of the specified type, dimensions, and initialization.
- `OBJARR` - Creates an array of object references.
- `PTRARR` - Creates an array of pointers.
- `REPLICATE` - Creates an array of given dimensions, filled with specified value.
- `SINDGEN` - Returns a string array with each element set to its subscript.
- `STRARR` - Returns string array containing zero-length strings.
- `TIMEGEN` - Returns an array of double-precision floating-point values that represent times in Julian dates.
- `UINTGEN` - Returns unsigned integer array with each element set to its subscript.
- `UNITARR` - Returns an unsigned integer vector or array.
- `UL64INDGEN` - Returns an unsigned 64-bit integer array with each element set to its subscript.
- `ULINDGEN` - Returns an unsigned longword array with each element set to its subscript.
- `ULONG64ARR` - Returns an unsigned 64-bit integer vector or array.
ULONARR - Returns an unsigned longword integer vector or array.

Array Manipulation

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

CMYK_CONVERT - Converts color triples to and from RGB and CMYK.
COLOR_QUAN - Converts color triples to and from RGB, HLS, and HSV.
COLOR_QUAN - Converts true-color (24-bit) image to pseudo-color (8-bit) image.
COLORMAP_APPLICABLE - Determines whether the current visual class supports the use of a colormap.
CT_LUMINANCE - Calculates the luminance of colors.
CW_PALETTE_EDITOR - Creates compound 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.
IDLgrPalette - Represents a color lookup table that maps indices to red, green, and blue values.
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 Lightness, Hue, 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.
TVLCT - Loads display color tables.
XLOADCT - Provides GUI to interactively select and load color tables.
XPALLETTE - Displays widget used to create and modify color tables.

Date and Time

BIN_DATE - Converts ASCII date/time string to binary string.
CALDAT - Converts Julian date to month, day, year.
CALENDAR - Displays a calendar for a given month or year.
JULDAY - Returns Julian Day Number for given month, day, and year.
SYSTIME - Returns the current time as either a date/time string, as the number of seconds elapsed since 1 January 1970, or as a Julian date/time value.
TIMEGEN - Returns an array of double-precision floating-point values that represent date/times in terms of Julian values.

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

**PLOTS** - Plots vectors and points.

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

**POLYPLOT** - Generates a polygon plot.

**PLOT** - Draws a graph of vector arguments.

**PLOTERR** - Plots error bars over a previously drawn plot.

**PLOTFIELD** - Plots a 2-D field. N random points are picked, and from each point a path is traced along the field.

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

**POLYSHADE** - Creates a shaded-surface representation of one or more solids described by a set of polygons.

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

**PROFILES** - Interactively examines image profiles.

**PROJECT_VOL** - Returns a two-dimensional image that is the projection of a 3-D volume of data onto a plane (similar to an X-ray).

**RDPIX** - Interactively displays the X position, Y position, and pixel value at the cursor.

**SCALE3** - Sets up transformation and scaling parameters for basic 3-D viewing.

**SCALE3D** - Scales the 3-D unit cube (a cube with the length of each side equal to 1) into the viewing area.

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

**SET_SHADING** - Modifies the light source shading parameters that affect the output of SHADE_SURF and POLYSHADE.

**SHADE_SURFACE** - Creates a shaded-surface representation of a regular or nearly-regular gridded surface.

**SHADE_SURFACE_IRR** - Creates a shaded-surface representation of an irregularly gridded elevation dataset. Given a 3-D volume and a contour value, SHADE_VOLUME produces a list of vertices and polygons describing the contour surface.

**SHADE_VOLUME** - Given a 3-D volume and a contour value, produces a list of vertices and polygons describing the contour surface.

**SHOW3** - Combines an image, a surface plot of the image data, and a contour plot of the images data in a single tri-level display.

**SURFACE** - Draws a wire-mesh representation of a two-dimensional array projected into two dimensions, with hidden lines removed.

**THREED** - Plots a 2D array as a pseudo 3D plot.

**TV** - Displays an image.

**TVCRS** - Manipulates the image display cursor.

**TVLCT** - Loads a predefined color table or a color table from specified variables.

**TVRD** - Returns the contents of the specified rectangular portion of the current graphics window or device.

**TVSCL** - Scales and displays an image.

**WINDOW** - Creates a window for the display of graphics or text.

**WSET** - Selects the current window.

**WSHOW** - Exposes or hides the designated window.

**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** - Designates the error recovery method.

---

**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.

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

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

**AXIS** - Draws an axis of the specified type and scale at a given position.

**BAR_PLOT** - Creates a bar graph.

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

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

**CONTOUR** - Draws a contour plot.

**CREATE_VIEW** - Sets the various system variables required to define a coordinate system and a 3-D view.

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

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

**DEFROI** - Defines an irregular region of interest of an image using the image display system and the cursor and mouse.

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

**DRAW_ROI** - Draws a region or group of regions to the current Direct Graphics device.

**EMPTY** - Empties the graphics output buffer.

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

**ERRPLOT** - Plots error bars over a previously drawn plot.

**FLICK** - Causes the display to flicker between two output images at a given rate.

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

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

**LABEL_REGION** - Consecutively labels all of the regions, or blobs, of a bi-level image with a unique region index.

**LOADCT** - Loads a pre-defined color table.

**OPLOT** - Plots vector data over a previously drawn plot.

**OPLOTERR** - Plots error bars over a previously drawn plot.

**PLOT** - Draws a graph of vector arguments.

**PLOT_3DBOX** - Plots a function of two variables (e.g., Z=f(X, Y)) inside a 3-D box.

**PLOTFIELD** - Plots a 2-D field. N random points are picked, and from each point a path is traced along the field.

**PLOTERR** - Plots individual data points with error bars.

**PLOTS** - Plots vectors and points.

---

**Direct Graphics**

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

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

**AXIS** - Draws an axis of the specified type and scale at a given position.

**BAR** - Creates a bar graph.

**BOX** - Emulates operation of a variable-sized box cursor.

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

**CONTOUR** - Draws a contour plot.

**CREATE** - Sets the various system variables required to define a coordinate system and a 3-D view.

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

**CVT** - Creates a bitmap byte array for a button label.

**DEFROI** - Defines an irregular region of interest of an image using the image display system and the cursor and mouse.

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

**DRAW** - Draws a region or group of regions to the current Direct Graphics device.

**EMPTY** - Empties the graphics output buffer.

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

**ERRPLOT** - Plots error bars over a previously drawn plot.

**FLICK** - Causes the display to flicker between two output images at a given rate.

**FORMAT** - Formats numbers as strings for use as axis values.

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

**LABEL** - Consecutively labels all of the regions, or blobs, of a bi-level image with a unique region index.

**LOADCT** - Loads a pre-defined color table.

**OPLOT** - Plots vector data over a previously drawn plot.

**OPLOTERR** - Plots error bars over a previously drawn plot.

**PLOT** - Draws a graph of vector arguments.

**PLOT_3DBOX** - Plots a function of two variables (e.g., Z=f(X, Y)) inside a 3-D box.

**PLOTFIELD** - Plots a 2-D field. N random points are picked, and from each point a path is traced along the field.

**PLOTERR** - Plots individual data points with error bars.

**PLOTS** - Plots vectors and points.

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

**POLAR_SURFACE** - Interpolates a surface from polar coordinates (R, Theta, Z) to rectangular coordinates (X, Y, Z).

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

**POLYSHADE** - Creates a shaded-surface representation of one or more solids described by a set of polygons.

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

**PROFILES** - Interactively examines image profiles.

**PROJECT_VOL** - Returns a two-dimensional image that is the projection of a 3-D volume of data onto a plane (similar to an X-ray).

**RDPIX** - Interactively displays the X position, Y position, and pixel value at the cursor.

**SCALE3** - Sets up transformation and scaling parameters for basic 3-D viewing.

**SCALE3D** - Scales the 3-D unit cube (a cube with the length of each side equal to 1) into the viewing area.

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

**SET_SHADING** - Modifies the light source shading parameters that affect the output of SHADE_SURF and POLYSHADE.

**SHADE_SURFACE** - Creates a shaded-surface representation of a regular or nearly-regular gridded surface.

**SHADE_SURFACE_IRR** - Creates a shaded-surface representation of an irregularly gridded elevation dataset. Given a 3-D volume and a contour value, SHADE_VOLUME produces a list of vertices and polygons describing the contour surface.

**SHADE_VOLUME** - Given a 3-D volume and a contour value, produces a list of vertices and polygons describing the contour surface.

**SHOW3** - Combines an image, a surface plot of the image data, and a contour plot of the images data in a single tri-level display.

**SURFACE** - Draws a wire-mesh representation of a two-dimensional array projected into two dimensions, with hidden lines removed.

**THREED** - Plots a 2D array as a pseudo 3D plot.

**TV** - Displays an image.

**TVCRS** - Manipulates the image display cursor.

**TVLCT** - Loads a predefined color table or a color table from specified variables.

**TVRD** - Returns the contents of the specified rectangular portion of the current graphics window or device.

**TVSCL** - Scales and displays an image.

**WINDOW** - Creates a window for the display of graphics or text.

**WSET** - Selects the current window.

**WSHOW** - Exposes or hides the designated window.

**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.
### Executive Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.COMPILE</td>
<td>Compiles programs without running.</td>
</tr>
<tr>
<td>.CONTINUE</td>
<td>Continues execution of a stopped program.</td>
</tr>
<tr>
<td>.EDIT</td>
<td>Opens files in editor windows of the IDLDE.</td>
</tr>
<tr>
<td>.FULL_RESET_SESSION</td>
<td>Does everything .RESET_SESSION does, plus additional reset tasks such as unloading sharable libraries.</td>
</tr>
<tr>
<td>.GO</td>
<td>Executes a previously compiled $MAIN$ program.</td>
</tr>
<tr>
<td>.OUT</td>
<td>Continues execution until the current routine returns.</td>
</tr>
<tr>
<td>.RESET_SESSION</td>
<td>Resets much of the state of an IDL session without requiring the user to exit and restart the IDL session.</td>
</tr>
<tr>
<td>.RETURN</td>
<td>Continues execution until RETURN statement.</td>
</tr>
<tr>
<td>.RNEW</td>
<td>Erases $MAIN$ program variables and then executes .RUN.</td>
</tr>
<tr>
<td>.RUN</td>
<td>Compiles and executes IDL commands from files or keyboard.</td>
</tr>
<tr>
<td>.SKIP</td>
<td>Skips over the next n statements and then single steps.</td>
</tr>
<tr>
<td>.STEP</td>
<td>Executes one or n statements from the current position.</td>
</tr>
<tr>
<td>.STEPPOWER</td>
<td>Executes a single statement if the statement does not call a routine.</td>
</tr>
<tr>
<td>.TRACE</td>
<td>Similar to .CONTINUE, but displays each line of code before execution.</td>
</tr>
</tbody>
</table>

### External Linking

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL_EXTERNAL</td>
<td>Calls a function in an external sharable object and returns a scalar value.</td>
</tr>
<tr>
<td>DLM_LOAD</td>
<td>Explicitly causes a DLM to be loaded.</td>
</tr>
<tr>
<td>IDLcomActiveX</td>
<td>Creates an IDL object that encapsulates an ActiveX control.</td>
</tr>
<tr>
<td>IDLcomDispatch</td>
<td>Creates an IDL object that encapsulates a COM object.</td>
</tr>
<tr>
<td>IDLEXBR_ASSISTANT</td>
<td>Launches the Export Bridge Assistant.</td>
</tr>
<tr>
<td>IDLJavaObject</td>
<td>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.</td>
</tr>
<tr>
<td>LINKIMAGE</td>
<td>Merges routines written in other languages with IDL at run-time.</td>
</tr>
<tr>
<td>MAKE_DLL</td>
<td>Compiles and links sharable libraries (DLLs).</td>
</tr>
</tbody>
</table>

### Font Manipulation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFONT</td>
<td>Interactive vector font editor and display tool.</td>
</tr>
<tr>
<td>IDLgrFont</td>
<td>- represents a typeface, style, weight, and point size that may be associated with text objects.</td>
</tr>
<tr>
<td>PS_SHOW_FONTS</td>
<td>Displays all the PostScript fonts that IDL knows about.</td>
</tr>
<tr>
<td>PSAFM</td>
<td>Converts Adobe Font Metrics file to IDL format.</td>
</tr>
</tbody>
</table>

### Image Processing

#### Contrast Enhancement and Filtering

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPT_HIST_EQUAL</td>
<td>Performs adaptive histogram equalization.</td>
</tr>
<tr>
<td>BUTTERWORTH</td>
<td>Returns the absolute value of the low-pass Butterworth kernel.</td>
</tr>
<tr>
<td>BYTSCL</td>
<td>Scales all values of an array into range of bytes.</td>
</tr>
<tr>
<td>CANNY</td>
<td>Implements the Canny edge-detection algorithm.</td>
</tr>
<tr>
<td>CONVOL</td>
<td>Convolves two vectors or arrays.</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>HILBERT</td>
<td>Constructs a Hilbert transform.</td>
</tr>
<tr>
<td>HIST_EQUAL</td>
<td>Histogram-equalizes an image.</td>
</tr>
<tr>
<td>IR_FILTER</td>
<td>Performs the infinite or finite impulse response filter on data.</td>
</tr>
<tr>
<td>LEEFILT</td>
<td>Performs the Lee filter algorithm on an image array.</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>Returns the median value of Array or applies a median filter.</td>
</tr>
<tr>
<td>ROBERTS</td>
<td>Returns an approximation of Roberts edge enhancement.</td>
</tr>
<tr>
<td>SMOOTH</td>
<td>Smooths with a boxcar average.</td>
</tr>
<tr>
<td>SOBEL</td>
<td>Returns an approximation of Sobel edge enhancement.</td>
</tr>
<tr>
<td>UNSHARP_MASK</td>
<td>Performs an unsharp-mask sharpening filter on a two-dimensional array or a truecolor image.</td>
</tr>
</tbody>
</table>

### Feature Extraction/Image Segmentation

<table>
<thead>
<tr>
<th>Command</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>
</tbody>
</table>

### Help Routines

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Invokes the IDL Online Help facility when entered at the IDL command line.</td>
</tr>
<tr>
<td>DOC_LIBRARY</td>
<td>Extracts documentation headers from IDL programs.</td>
</tr>
<tr>
<td>HELP</td>
<td>Provides information about the current IDL session.</td>
</tr>
<tr>
<td>MEMORY</td>
<td>Returns information about dynamic memory currently in use by the IDL session.</td>
</tr>
<tr>
<td>MK_HTML_HELP</td>
<td>Converts text documentation headers to HTML files.</td>
</tr>
<tr>
<td>ONLINE_HELP</td>
<td>Invokes online help viewer from programs.</td>
</tr>
<tr>
<td>STRUCT_HIDE</td>
<td>Prevents the IDL HELP procedure from displaying information about structures or objects.</td>
</tr>
</tbody>
</table>
IMAGE_STATISTICS - Computes sample statistics for a given array of values.
ISOCONTOUR - Interprets the contouring algorithm found in the IDLgrContour object.
ISOSURFACE - Returns topologically consistent triangles by using oriented tetrahedral decomposition.
LABEL_REGION - Labels regions (blobs) of a bi-level image.
MAX - Returns the value of the largest element of Array.
MEDIAN - Returns the median value of Array or applies a median filter.
MIN - Returns the value of the smallest element of an array.
PROFILES - Interactively examines image profiles.
RADON - Returns the Radon transform of a two-dimensional image.
REGION_GROW - Perform region growing.
SEARCH2D - Finds “objects” or regions of similar data within a 2D array.
THIN - Returns the “skeleton” of a bi-level image.
UNIQ - Returns subscripts of the unique elements in an array.
WATERSHED - Applies the morphological watershed operator to a grayscale image.
WHERE - Returns subscripts of nonzero array elements.

Image Display
DISSOLVE - Provides a digital “dissolve” effect for images.
IDLgImage - Creates an image object that represents a mapping from a 2D array of data values to a 2D array of pixel colors.
IDLgPalette - Represents a color lookup table that maps indices to red, green, and blue values.
IMAGE - Creates an iTool and associated user interface (UI) configured to display and manipulate image data.
RDPIX - Interactively displays image pixel values.
SLIDE_IMAGE - Creates a scrolling graphics window for examining large images.
TV - Displays an image. To scale and display the image, use TVSCL.
TVCRS - Manipulates the image display cursor.
TVLCT - Loads display color tables.
TVSCL - Scales and displays an image.
XOBJVIEW - Displays object viewer widget.
XOBJVIEW_WRITE_IMAGE - Write the object currently displayed in XOBJVIEW to an image file.
ZOOM - Zooms portions of the display.
ZOOM_24 - Zooms portions of true-color (24-bit) display.

Image Geometry Transformations
CONGRID - Resamples an image to any dimensions.
EXPAND - Shrinks/expands image using bilinear interpolation.
EXTRAC - Returns sub-matrix of input array. Array operators (e.g., * and :) should usually be used instead.
INTERPOLATE - Returns an array of interpolates.
INVERT - Computes the inverse of a square array.
POLY_2D - Performs polynomial warping of images.
POLYWARP - Performs polynomial spatial warping.
REBIN - Resizes a vector or array by integer multiples.
REFORM - Changes array dimensions without changing the total number of elements.
REVERSE - Reverses the order of one dimension of an array.
ROT - Rotates an image by any amount.
ROTATE - Rotates/transposes an array in multiples of 90 degrees.
SHIFT - Shifts elements of vectors or arrays by a specified number of elements.
TRANSPOSE - Transposes an array.
WARP_TRI - Warps an image using control points.

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

Regions of Interest
CW_DEFROI - Creates compound widget used to define region of interest.
DEFROI - Defines an irregular region of interest of an image.
DRAW_ROI - Draws region or group of regions to current Direct Graphics device.
IDLanROI - Represents a region of interest used for analysis.
IDLanROIGroup - Analytical representation of a group of regions of interest.
IDLgrROI - Object graphics representation of a region of interest.
IDLgrROIGroup - Object Graphics representation of a group of regions of interest.
LABEL_REGION - Labels regions (blobs) of a bi-level image.
### Functional List of IDL Routines

**Input/Output**

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REGION_GROW</strong></td>
<td>Grows an initial region to include all areas that match specified constraints.</td>
</tr>
<tr>
<td><strong>XROI</strong></td>
<td>Utility for defining regions of interest, and obtaining geometry and statistical data about these ROIs.</td>
</tr>
</tbody>
</table>

**General File Access**

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APP_USER_DIR</strong></td>
<td>Provides access to the application user directory.</td>
</tr>
<tr>
<td><strong>APP_USER_DIR_QUERY</strong></td>
<td>Locates existing application user directories.</td>
</tr>
<tr>
<td><strong>DIALOG_PICKFILE</strong></td>
<td>Creates native file-selection dialog.</td>
</tr>
<tr>
<td><strong>FILE_BASENAME</strong></td>
<td>Returns the final segment of a file path.</td>
</tr>
<tr>
<td><strong>FILE_COPY</strong></td>
<td>Copies files or directories to a new location.</td>
</tr>
<tr>
<td><strong>FILE_DELETE</strong></td>
<td>Removes a file or empty directory.</td>
</tr>
<tr>
<td><strong>FILE_DIRNAME</strong></td>
<td>Returns the directory name of a file path consisting of everything except the final segment of the file path.</td>
</tr>
<tr>
<td><strong>FILE_EXPAND_PATH</strong></td>
<td>Returns a fully qualified path regardless of the current working directory.</td>
</tr>
<tr>
<td><strong>FILE_INFO</strong></td>
<td>Returns status information about a file.</td>
</tr>
<tr>
<td><strong>FILE_LINES</strong></td>
<td>Returns the number of lines of text in a file.</td>
</tr>
<tr>
<td><strong>FILE_LINK</strong></td>
<td>Creates Unix file links.</td>
</tr>
<tr>
<td><strong>FILE_MKDIR</strong></td>
<td>Creates a new directory or directories.</td>
</tr>
<tr>
<td><strong>FILE_POLL_INPUT</strong></td>
<td>Blocks processing until it detects that a read operation on a specified file will succeed.</td>
</tr>
<tr>
<td><strong>FILE_READLINK</strong></td>
<td>Returns the path pointed to by a Unix symbolic link.</td>
</tr>
<tr>
<td><strong>FILE_SAME</strong></td>
<td>Determines it two different file names refer to the same underlying file.</td>
</tr>
<tr>
<td><strong>FILE_SEARCH</strong></td>
<td>Returns a string array containing the names of all files matching the input path specification.</td>
</tr>
<tr>
<td><strong>FILE_TEST</strong></td>
<td>Test a file or directory for existence and other specific attributes.</td>
</tr>
<tr>
<td><strong>FILE_WHICH</strong></td>
<td>Returns the path for the first file for the given name found by searching the specified path.</td>
</tr>
<tr>
<td><strong>FILEPATH</strong></td>
<td>Returns full path to a file in the IDL distribution.</td>
</tr>
<tr>
<td><strong>FSTAT</strong></td>
<td>Returns information about a specified file unit.</td>
</tr>
<tr>
<td><strong>PATH_SEP</strong></td>
<td>Returns the proper file path segment separator character for the current operating system.</td>
</tr>
</tbody>
</table>

**Image Data Formats**

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IDLffDICOM</strong></td>
<td>Contains the data for one or more images embedded in a DICOM part 10 file.</td>
</tr>
<tr>
<td><strong>IDLffDicomEx</strong></td>
<td>Allows you to read and write data contained in a DICOM file. See the Medical Imaging in IDL manual.</td>
</tr>
<tr>
<td><strong>IDLffJPEG2000</strong></td>
<td>Contains the data for one or more JPEG2000 files.</td>
</tr>
<tr>
<td><strong>IDLffMJPEG2000</strong></td>
<td>Allows you to create and play Motion JPEG2000 animations contained in MJ2 files.</td>
</tr>
<tr>
<td><strong>IDLffMrSID</strong></td>
<td>Allows you to query and load image data from a MrSID image file.</td>
</tr>
<tr>
<td><strong>MPEG_CLOSE</strong></td>
<td>Closes an MPEG sequence.</td>
</tr>
<tr>
<td><strong>MPEG_OPEN</strong></td>
<td>Opens an MPEG sequence.</td>
</tr>
<tr>
<td><strong>MPEG_PUT</strong></td>
<td>Inserts an image array into an MPEG sequence.</td>
</tr>
<tr>
<td><strong>MPEG_SAVE</strong></td>
<td>Saves an MPEG sequence to a file.</td>
</tr>
<tr>
<td><strong>READ_BMP</strong></td>
<td>Reads Microsoft Windows bitmap file (.BMP).</td>
</tr>
<tr>
<td><strong>READ_DICOM</strong></td>
<td>Reads an image from a DICOM file.</td>
</tr>
<tr>
<td><strong>READ_GIF</strong></td>
<td>Reads GIF file (.GIF).</td>
</tr>
<tr>
<td><strong>READ_IMAGE</strong></td>
<td>Reads the image contents of a file and returns the image in an IDL variable.</td>
</tr>
<tr>
<td><strong>READ_INTERFILE</strong></td>
<td>Reads Interfile (v3.3) file.</td>
</tr>
<tr>
<td><strong>READ_JPEG</strong></td>
<td>Reads JPEG file.</td>
</tr>
<tr>
<td><strong>READ_MRSID</strong></td>
<td>Reads MrSID file.</td>
</tr>
<tr>
<td><strong>READ_PICT</strong></td>
<td>Reads Macintosh PICT (version 2) bitmap file.</td>
</tr>
<tr>
<td><strong>READ_PPM</strong></td>
<td>Reads PGM (gray scale) or PPM (portable pixmap for color) file.</td>
</tr>
<tr>
<td><strong>READ_SRF</strong></td>
<td>Reads Sun Raster Format file.</td>
</tr>
<tr>
<td><strong>READ_TIFF</strong></td>
<td>Reads TIFF format file.</td>
</tr>
<tr>
<td><strong>READ_X11_BITMAP</strong></td>
<td>Reads X11 bitmap file.</td>
</tr>
<tr>
<td><strong>READ_XWD</strong></td>
<td>Reads X Windows Dump file.</td>
</tr>
<tr>
<td><strong>TVRD</strong></td>
<td>Reads an image from a window into a variable.</td>
</tr>
<tr>
<td><strong>WRITE_BMP</strong></td>
<td>Writes Microsoft Windows Version 3 device independent bitmap file (.BMP).</td>
</tr>
<tr>
<td><strong>WRITE_GIF</strong></td>
<td>Writes GIF file (.GIF).</td>
</tr>
<tr>
<td><strong>WRITE_IMAGE</strong></td>
<td>Writes an image and its color table vectors, if any, to a file of a specified type.</td>
</tr>
<tr>
<td><strong>WRITE_JPEG</strong></td>
<td>Writes JPEG file.</td>
</tr>
<tr>
<td><strong>WRITE_NRIF</strong></td>
<td>Writes NCAR Raster Interchange Format rasterfile.</td>
</tr>
<tr>
<td><strong>WRITE_PICT</strong></td>
<td>Writes Macintosh PICT (version 2) bitmap file.</td>
</tr>
<tr>
<td><strong>WRITE_PNG</strong></td>
<td>Writes Portable Network Graphics (PNG) file.</td>
</tr>
<tr>
<td><strong>WRITE_PPM</strong></td>
<td>Writes PPM (true-color) or PGM (gray scale) file.</td>
</tr>
<tr>
<td><strong>WRITE_SRF</strong></td>
<td>Writes Sun Raster File (SRF).</td>
</tr>
<tr>
<td><strong>WRITE_TIFF</strong></td>
<td>Writes TIFF file with 1 to 3 channels.</td>
</tr>
<tr>
<td><strong>XOBJVIEW_WRITE_IMAGE</strong></td>
<td>Write the object currently displayed in XOBJVIEW to an image file.</td>
</tr>
</tbody>
</table>
Scientific Data Formats

CDF Routines - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

EOS Routines - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

H5_BROWSER - Opens a GUI to view contents of HDF5 files.

HDF Routines - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

HDF5 Routines - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

HDF_BROWSE - Frees previously-reserved file units.

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

NCDF Routines - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

Other Data Formats

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

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

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

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

IDLffXMLDOM - Represents classes that provide support for IDL's XML Document Object Model (DOM). See "IDLffXMLDOM Classes" on page 83.

IDLffXMLSAX - Represents an XML SAX level 2 parser.

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

READ_ASCII - Reads data from an ASCII file.

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

READ_SYLK - Reads Symbolic Link format spreadsheet file.

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

READ_WAVE - Reads Wavefront Advanced Visualizer file.

WRITE_SYLK - Writes SYLK (Symbolic Link) spreadsheet file.

WRITE_WAVE - Writes unformatted binary data to a file.

WRITE_WAV - Writes Wavefront Advanced Visualizer file.

READ_BINARY - Reads unformatted binary data from a file.

READU - Reads unformatted binary data from a file.

READS - Reads formatted input from keyboard or file.

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

READ - Reads formatted input from a string variable.

READU - Reads unformatted binary data from a file.

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

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

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

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

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

SHMUNMAP - 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.

WRITEU - Writes unformatted binary data to a 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

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

IMAP - Displays georeferenced data in an IDL process.

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

MAP_2POINTS - Returns distance, azimuth, and path relating to the great circle or rhumb line connecting two points on a sphere.

MAP_CONTINENTS - Draws continental boundaries, filled continents, political boundaries, coastlines, and/or rivers, over an existing map projection established by 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.
KRIGW2D - Interpolates set of points using kriging.
LADFIT - 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.
POLY_FIT - Performs a least-square polynomial fit.
REGRESS - Computes fit using multiple linear regression.
SFIT - Performs polynomial fit to a surface.
SVDFIT - Multivariate least squares fit using SVD method.
TRIGRID - Interpolates irregularly-gridded data to a regular grid from a triangulation.

Differentiation and Integration

CRVLENGTH - Computes the length of a curve.
DERIV - Performs differentiation using 3-point Langrangian interpolation.
DERIVSIG - Computes standard deviation of derivative found by DERIV.
INT_2D - Computes the double integral of a bivariate function.
INT_3D - Computes the triple integral of a trivariate function.
INT_TABULATED - Integrates a tabulated set of data.
LSODE - Advances a solution to a system of ordinary differential equations one time-step H.
QROMB - Evaluates integral over a closed interval.
QROMO - Evaluates integral over an open interval.
QSIMP - Evaluates integral using Simpson’s rule.
RK4 - Solves differential equations using fourth-order Runge-Kutta method.

Eigenvalues and Eigenvectors

EIGENQL - Computes eigenvalues and eigenvectors of a real, symmetric array.
EIGENVEC - Computes eigenvectors of a real, non-symmetric array.
ELMHES - Reduces nonsymmetric array to upper Hessenberg form.
HQR - Returns all eigenvalues of an upper Hessenberg array.
TRIQL - Determines eigenvalues and eigenvectors of tridiagonal array.
TRIRED - Reduces a real, symmetric array to tridiagonal form.

Gridding and Interpolation

BILINEAR - Computes array using bilinear interpolation.
CONGRID - Shrinks or expands the size of an array by an arbitrary amount.
GRID_INPUT - Preprocesses and sorts two-dimensional scattered data points, and removes duplicate values.
GRID_TPS - Uses thin plate splines to interpolate a set of values over a regular 2D grid, from irregularly sampled data values.
GRID3 - Creates a regularly-gridded 3D dataset from a set of scattered 3D nodes.
GRIDDATA - Interpolates scattered data values and locations sampled on a plane or a sphere to a regular grid.
INTERPOL - Performs linear interpolation on vectors.
INTERPOLATE - Returns an array of interpolates.
KRIGW2D - Interpolates set of points using kriging.
MIN_CURVE_SURF - Interpolates points with a minimum curvature surface or a thin-plate-spline surface. Useful with CONTOUR.
**LAPACK Routines**

- **LA_CHOLODC** - Computes the Cholesky factorization of an $n$-by-$n$ symmetric positive-definite array.
- **LA_CHOLOMPROVE** - Uses Cholesky factorization to improve the solution to a system of linear equations.
- **LA_CHOLSOL** - Used in conjunction with LA_CHOLODC to solve a set of linear equations.
- **LA_DETERM** - Uses LU decomposition to compute the determinant of a square array.
- **LA_EIGENPROBLEM** - Uses the QR algorithm to compute eigenvalues and eigenvectors of an array.
- **LA_EIGENOL** - Computes selected eigenvalues and eigenvectors.
- **LA_EIGENVEC** - Uses the QR algorithm to compute all of some eigenvectors of an array.
- **LA_ELMHES** - Reduces a real nonsymmetric or complex array to upper Hessenberg form.
- **LA_GM LINEAR MODEL** - Used to solve a general Gauss-Markov linear model problem.
- **LA_HOR** - Uses the multishift QR algorithm to compute all eigenvalues of an array.
- **LA_INVERT** - Uses LU decomposition to compute the inverse of a square array.
- **LA_LEAST SQUARE_EQUALITY** - Used to solve linear least-squares problems.
- **LA_LEAST_SQUARES** - Used to solve linear least-squares problems.
- **LA_LINEAR_EQUATION** - Uses LU decomposition to solve a system of linear equations.
- **LA_LUDC** - Computes the LU decomposition of an array.
- **LA_LUMPROVE** - Uses LU decomposition to improve the solution to a system of linear equations.
- **LA_LUSOL** - Used in conjunction with LA_LUDC to solve a set of linear equations.
- **LA_SVD** - Computes the singular value decomposition of an array.
- **LA_TRIDC** - Computes the LU decomposition of a tridiagonal array.
- **LA_TRIMPROVE** - Improves the solution to a system of linear equations with a tridiagonal array.
- **LA_TRIQL** - Uses the QL and QR variants of the implicitly-shifted QR algorithm to compute the eigenvalues and eigenvectors of an array.
- **LA_TRIRID** - Reduces a real symmetric or complex Hermitian array to real tridiagonal form.
- **LA_TRISOL** - Used in conjunction with LA_TRIDC to solve a set of linear equations.

**Hypothesis Testing**

- **CTI_TEST** - Performs chi-square goodness-of-fit test.
- **FV_TEST** - Performs the F-variance test.
- **KW_TEST** - Performs Kruskal-Wallis H-test.
- **LNP_TEST** - Computes the Lomb Normalized Periodogram.
- **MD_TEST** - Performs the Median Delta test.
- **R_TEST** - Runs test for randomness.
- **RS_TEST** - Performs the Wilcoxon Rank-Sum test.
- **S_TEST** - Performs the Sign test.
- **TM_TEST** - Performs t-means test.
- **XSO_TEST** - Computes Chi-square goodness-of-fit test.

**Linear Systems**

- **CHOLDC** - Constructs Cholesky decomposition of a matrix.
- **CHOLSOL** - Solves set of linear equations (use with CHOLDC).
- **COND** - Computes the condition number of a square matrix.
- **CRAMER** - Solves system of linear equations using Cramer’s rule.
- **CROSSP** - Computes vector cross product.
- **DETERM** - Computes the determinant of a square matrix.
- **GS_ITER** - Solves linear system using Gauss-Seidel iteration.
- **IDENTITY** - Returns an identity array.
- **INVERT** - Computes the inverse of a square array.
- **LINBCG** - Solves a set of sparse linear equations using the iterative biconjugate gradient method.
- **LU_COMPLEX** - Solves complex linear system using LU decomposition.
- **LUDC** - Replaces array with the LU decomposition.
- **LUMPROVE** - Uses LU decomposition to iteratively improve an approximate solution.
- **LUSOL** - Solves a set of linear equations. Use with LUDC.
- **NORM** - Computes Euclidean norm of vector or Infinity norm of array.
- **SVDC** - Computes Singular Value Decomposition of an array.
- **SVSOL** - Solves set of linear equations using back-substitution.
- **TRACE** - Computes the trace of an array.
- **TRISOL** - Solves tridiagonal systems of linear equations.

**Mathematical Error Assessment**

- **CHECK_MATH** - Returns and clears accumulated math error status.
### Mathematics

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FINITE</strong></td>
<td>Returns True if its argument is finite.</td>
</tr>
<tr>
<td><strong>MACHAR</strong></td>
<td>Determines and returns machine-specific parameters affecting floating point arithmetic.</td>
</tr>
<tr>
<td><strong>ABS</strong></td>
<td>Returns the closest integer greater than or equal to X.</td>
</tr>
<tr>
<td><strong>CEIL</strong></td>
<td>Returns radius and center of circle, given 3 points.</td>
</tr>
<tr>
<td><strong>CIR_3PNT</strong></td>
<td>Rounds a complex array.</td>
</tr>
<tr>
<td><strong>DIAG_MATRIX</strong></td>
<td>Constructs a diagonal matrix from an input vector, or if given a matrix, then extracts a diagonal vector.</td>
</tr>
<tr>
<td><strong>DIST</strong></td>
<td>Creates array with each element proportional to its frequency.</td>
</tr>
<tr>
<td><strong>EXP</strong></td>
<td>Returns the exponential function of given expression.</td>
</tr>
<tr>
<td><strong>FLOOR</strong></td>
<td>Returns closest integer less than or equal to argument.</td>
</tr>
<tr>
<td><strong>IMAGINARY</strong></td>
<td>Returns the imaginary part of a complex value.</td>
</tr>
<tr>
<td><strong>ISHFT</strong></td>
<td>Performs integer bit shift.</td>
</tr>
<tr>
<td><strong>LEEFILT</strong></td>
<td>Performs the Lee filter algorithm on an image array.</td>
</tr>
<tr>
<td><strong>MATRIX_MULTIPL Y</strong></td>
<td>Calculates the IDL matrix-multiply operator (#) of two (possibly transposed) arrays.</td>
</tr>
<tr>
<td><strong>MATRIX_POWER</strong></td>
<td>Computes the product of a matrix with itself.</td>
</tr>
<tr>
<td><strong>PNT_LINE</strong></td>
<td>Returns the perpendicular distance between a point and a line.</td>
</tr>
<tr>
<td><strong>POLY_AREA</strong></td>
<td>Returns the area of a polygon given the coordinates of its vertices.</td>
</tr>
<tr>
<td><strong>PRIMES</strong></td>
<td>Computes the first K prime numbers.</td>
</tr>
<tr>
<td><strong>PRODUCT</strong></td>
<td>Returns the product of elements within an array.</td>
</tr>
<tr>
<td><strong>ROUND</strong></td>
<td>Returns the integer closest to its argument.</td>
</tr>
<tr>
<td><strong>SPH_4PNT</strong></td>
<td>Returns center and radius of a sphere given 4 points.</td>
</tr>
<tr>
<td><strong>SQRT</strong></td>
<td>Returns the square root of X.</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>Sums of the elements of an array.</td>
</tr>
<tr>
<td><strong>VOIGT</strong></td>
<td>Calculates intensity of atomic absorption line (Voight) profile.</td>
</tr>
</tbody>
</table>

### Miscellaneous Math Routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FINITE</strong></td>
<td>Returns True if its argument is finite.</td>
</tr>
<tr>
<td><strong>MACHAR</strong></td>
<td>Determines and returns machine-specific parameters affecting floating point arithmetic.</td>
</tr>
<tr>
<td><strong>ABS</strong></td>
<td>Returns the closest integer greater than or equal to X.</td>
</tr>
<tr>
<td><strong>CEIL</strong></td>
<td>Returns radius and center of circle, given 3 points.</td>
</tr>
<tr>
<td><strong>CIR_3PNT</strong></td>
<td>Rounds a complex array.</td>
</tr>
<tr>
<td><strong>DIAG_MATRIX</strong></td>
<td>Constructs a diagonal matrix from an input vector, or if given a matrix, then extracts a diagonal vector.</td>
</tr>
<tr>
<td><strong>DIST</strong></td>
<td>Creates array with each element proportional to its frequency.</td>
</tr>
<tr>
<td><strong>EXP</strong></td>
<td>Returns the exponential function of given expression.</td>
</tr>
<tr>
<td><strong>FLOOR</strong></td>
<td>Returns closest integer less than or equal to argument.</td>
</tr>
<tr>
<td><strong>IMAGINARY</strong></td>
<td>Returns the imaginary part of a complex value.</td>
</tr>
<tr>
<td><strong>ISHFT</strong></td>
<td>Performs integer bit shift.</td>
</tr>
<tr>
<td><strong>LEEFILT</strong></td>
<td>Performs the Lee filter algorithm on an image array.</td>
</tr>
<tr>
<td><strong>MATRIX_MULTIPL Y</strong></td>
<td>Calculates the IDL matrix-multiply operator (#) of two (possibly transposed) arrays.</td>
</tr>
<tr>
<td><strong>MATRIX_POWER</strong></td>
<td>Computes the product of a matrix with itself.</td>
</tr>
<tr>
<td><strong>PNT_LINE</strong></td>
<td>Returns the perpendicular distance between a point and a line.</td>
</tr>
<tr>
<td><strong>POLY_AREA</strong></td>
<td>Returns the area of a polygon given the coordinates of its vertices.</td>
</tr>
<tr>
<td><strong>PRIMES</strong></td>
<td>Computes the first K prime numbers.</td>
</tr>
<tr>
<td><strong>PRODUCT</strong></td>
<td>Returns the product of elements within an array.</td>
</tr>
<tr>
<td><strong>ROUND</strong></td>
<td>Returns the integer closest to its argument.</td>
</tr>
<tr>
<td><strong>SPH_4PNT</strong></td>
<td>Returns center and radius of a sphere given 4 points.</td>
</tr>
<tr>
<td><strong>SQRT</strong></td>
<td>Returns the square root of X.</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>Sums of the elements of an array.</td>
</tr>
<tr>
<td><strong>VOIGT</strong></td>
<td>Calculates intensity of atomic absorption line (Voight) profile.</td>
</tr>
</tbody>
</table>

### Multivariate Analysis

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLUSTER_WTS</strong></td>
<td>Computes cluster weights of array for cluster analysis.</td>
</tr>
<tr>
<td><strong>CLUSTER</strong></td>
<td>Performs cluster analysis.</td>
</tr>
<tr>
<td><strong>CLUSTER_TREE</strong></td>
<td>Computes the hierarchical clustering for a set of m items in an n-dimensional space.</td>
</tr>
<tr>
<td><strong>CTI_TEST</strong></td>
<td>Performs chi-square goodness-of-fit test.</td>
</tr>
<tr>
<td><strong>DENDRO_PLOT</strong></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><strong>DENDROGRAM</strong></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><strong>DISTANCE_MEASURE</strong></td>
<td>Computes the pairwise distance between a set of items or observations.</td>
</tr>
<tr>
<td><strong>KW_TEST</strong></td>
<td>Performs Kruskal-Wallis H-test.</td>
</tr>
<tr>
<td><strong>M_CORRELATE</strong></td>
<td>Computes multiple correlation coefficient.</td>
</tr>
</tbody>
</table>

### Nonlinear Equations

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BROYDEN</strong></td>
<td>Solves nonlinear equations using Broyden’s method.</td>
</tr>
<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>
</tbody>
</table>

### Optimization

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

### Probability

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

### Sparse Arrays

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

**NOTE:** SPRSIN must be used to convert to sparse storage format before the other routines can be used.
### Special Math Functions

- **BESELI** - Returns the I Bessel function of order N for X.
- **BESELJ** - Returns the J Bessel function of order N for X.
- **BESELY** - Returns the Y Bessel function of order N for X.
- **BETA** - Returns the value of the beta function.
- **ERF** - Returns the value of an error function.
- **ERFC** - Returns the value of a complementary error function.
- **ERFCX** - Returns the value of a scaled complementary error function.
- **EXPINT** - Returns the value of the exponential integral.
- **GAMMA** - Returns the gamma function of X.
- **IBETA** - Computes the incomplete beta function.
- **IGAMMA** - Computes the incomplete gamma function.
- **LAGUERRE** - Returns value of the associated Laguerre polynomial.
- **LEGENDRE** - Returns value of the associated Legendre polynomial.
- **LNGAMMA** - Returns logarithm of the gamma function of X.
- **POLY** - Evaluates polynomial function of a variable.
- **SPHER_HARM** - Returns value of the spherical harmonic function.

### Statistical Fitting

- **COMFIT** - Fits paired data using one of six common filtering functions.
- **CURVEFIT** - Fits multivariate data with a user-supplied function.
- **FUNCT** - Evaluates the sum of a Gaussian and a 2nd-order polynomial and optionally returns the value of its partial derivatives.
- **LADFIT** - Fits paired data using least absolute deviation method.
- **LINFIT** - Fits by minimizing the Chi-square error statistic.
- **REGRESS** - Multiple linear regression.
- **SVDFIT** - Multivariate least squares fit using SVD method.

### Statistical Tools

- **FACTORIAL** - Computes the factorial N!.
- **HIST_2D** - Returns histogram of two variables.
- **HISTOGRAM** - Computes the density function of an array.
- **KURTOSIS** - Computes statistical kurtosis of n-element vector.
- **MAX** - Returns the value of the largest element of an array.
- **MEAN** - Computes the mean of a numeric vector.
- **MEANABSDEV** - Computes the mean absolute deviation of a vector.
- **MEDIAN** - Returns the median value of Array or applies a median filter.
- **MIN** - Returns the value of the smallest element of an array.
- **MOMENT** - Computes mean, variance, skewness, and kurtosis.
- **RANDOM** - Returns normally-distributed pseudo-random numbers.
- **RANDOMU** - Returns uniformly-distributed pseudo-random numbers.
- **RANKS** - Computes magnitude-based ranks.
- **SKEWNESS** - Computes statistical skewness of an n-element vector.

### Time-Series Analysis

- **AC_CORRELATE** - Computes autocorrelation.
- **C_CORRELATE** - Computes cross correlation.
- **SMOOTH** - Smooths with a boxcar average.
- **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 logarithm 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

#### Analysis Objects

- **IDLanROI** - Represents a region of interest.
File Format Objects

IDLffDICOM - Contains the data for one or more images embedded in a DICOM Part 10 file.

IDLffDcomExt - Allows you to read and write data contained in a DICOM file. See the Medical Imaging in IDL manual.

IDLffDXF - Contains geometry, connectivity and attributes for graphics primitives.

IDLffJPEG2000 - 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.

IDLffLangCat - Finds and loads an XML language catalog.

IDLffMrSID - Used to query information about and load image data from a MrSID (.sid) image file.

IDLffXMLDOM - Represents an XML SAX level 2 parser.

IDLffXMLDOMTreeWalker - Allows tree-walking navigation of the XML DOM tree.

IDLffXMLDOMText - References a text node in the XML document.

IDLffXMLDOMProcessingInstruction - Represents a processing instruction in an XML document.

IDLffXMLDOMNamedNodeMap - Container for IDLffXMLDOM nodes that uses node names to access the nodes.

IDLffXMLDOMNode - Abstract class used as a superclass for other IDLffXMLDOMNode node classes.

IDLffXMLDOMNodeList - Allows iterative navigation of the XML DOM tree.

IDLffXMLDOMNodeList - Container for IDLffXMLDOM nodes.

IDLffXMLDOMNode - Represents a notation in the XML DTD.

IDLffXMLDOMProcessingInstruction - Represents a processing instruction in an XML document.

IDLffXMLDOMText - References a text node in the XML document.

IDLffXMLDOMTreeWalker - Allows tree-walking navigation of the XML DOM tree.

IDLffXMLSAX - Represents an XML SAX level 2 parser.

IDLgrMPEG - Creates an MPEG movie file from an array of image frames.

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

Graphic Objects—Destination

IDL_Container - A container for other objects.

IDLgrBuffer - An in-memory, off-screen destination object.

IDLgrClipboard - A destination object representing the native clipboard.

IDLgrPrinter - Represents a hardcopy graphics destination.

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

Graphic Objects—Display

IDL_Container - Object that holds other objects.

IDLgrModel - Represents a graphical item or group of items that can be transformed (rotated, scaled, and/or translated).

IDLgrScene - Represents the entire scene to be drawn and serves as a container of IDLgrView or IDLgrViewgroup objects.

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

IDLgrViewgroup - A simple container object that contains one or more IDLgrView objects. An IDLgrScene can contain one or more of these objects.

Graphic Objects—Visualization

IDLgrAxis - Represents a single vector that may include a set of tick marks, tick labels, and a title.

IDLgrColorbar - Consists of a color-ramp with an optional framing box and annotation axis.

IDLgrContour - Draws a contour plot from data stored in a regular or from a set of unstructured points.

IDLgrFont - Represents a typeface, style, weight, and point size that may be associated with text objects.

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.

IDLgrLegend - Provides a simple interface for displaying a legend.

IDLgrLight - Represents a source of illumination for 3D graphic objects.

IDLgrPalette - Represents a color lookup table that maps indices to red, green, and blue values.

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

IDLgrPlot - Creates sets of polylines connecting data points in 2D space.

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

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

IDLgrROI - Object graphics representation of a region of interest.

IDLgrROIGroup - Object Graphics representation of a group of regions of interest.
### Functional List of IDL Routines

| IDLgSurface | A shaded or vector representation of a mesh grid. No superclasses. |
| IDLgSymbol | Represents a graphical element that is plotted relative to a particular position. |
| IDLgTessellator | Converts a simple concave polygon (or a simple polygon with "holes") into a number of simple convex polygons (general triangles). |
| IDLgText | Represents one or more text strings that share common rendering attributes. |
| IDLgView | Represents a rectangular area in which graphics objects are drawn. It is a container for objects of the IDLgModel class. |
| IDLgViewgroup | A simple container object that contains one or more IDLgView objects. An IDLgScene can contain one or more of these objects. |
| IDLgVolume | Represents mapping from a 3D array of data to a 3D array of voxel colors, which, when drawn, are projected to two dimensions. |

### iTools System Objects

| IDLCommand | The base functionality for the iTools command buffer system. |
| IDLCommandSet | A container for IDLCommand objects, which allows a group of commands to be managed as a single item. |
| IDLComponent | A core or base component, from which all other components subclass. |
| IDLContainer | 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. |
|IDLData | 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. |
| IDLitDataContainer | A container for IDLData and IDLitDataContainer objects. This container is used to form hierarchical data structures. Data and DataContainer objects can be added and removed from a DataContainer during program execution, allowing for dynamic creation of composite data types. |
| IDLitDataOperation | A subclass to IDLitDataOperation that automates data access and automatically records information for the undo-redo system. |
| IDLitMessaging | An interface providing common methods to send or trigger messaging and error actions, which may occur during execution. |
| IDLitManipulator | The base functionality of the iTools manipulator system. |
| IDLitManipulatorContainer | A container for IDLitManipulator objects, which allows for the construction of manipulator hierarchies. This container implements the concept of a current manipulator for the items it contains. |
| IDLitManipulatorManager | A specialization of the manipulator container (IDLManipulatorContainer), which acts as the root of the manipulator hierarchy. |
| IDLitManipulatorVisual | The means for iTool developers to create visual elements associated with an interactive manipulator. |
| IDLitOperation | 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. |
| IDLitParameter | An interface providing parameter management methods to associate parameter names with IDLitData objects. |

### IDLParameterSet

A specialized subclass of the IDLitDataContainer class. This class provides the ability to associate names with contained IDLitData objects.

### IDLitReader

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.

### IDLitTool

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.

### IDLitUI

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

### IDLitVisualization

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

### IDLitWindow

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

### 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.

### Miscellaneous Objects

| IDLContainer | A container for other objects. |
| IDL_IDLBridge | Creates and controls a synchronous or an asynchronous slave IDL process that can operate sequentially or in parallel with the main IDL process. |
| IDL_Savefile | Provides complete query and restore capabilities for IDL SAVE files. |
| IDLComActiveX | Creates an IDL object that encapsulates an ActiveX control. |
| IDLComIDispatch | 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. |
| IDLsysMonitorInfo | Contains information about the display monitor or monitors attached to your system. Use this object to obtain information about the monitor(s) when building applications that use multiple monitors. |
| TrackBall | Translates widget events from a draw widget into transformations that emulate a virtual trackball (for transforming object graphics in three dimensions). |

### Network Objects

| IDLnetOGCWCS | Provides the ability to query and return data from and Open Geospatial Consortium Web Coverage Service server. |
| IDLnetOGCWMS | Provides the ability to query and return data from and Open Geospatial Consortium Web Map Service server. |

### 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 shareable object and returns a scalar value. |
Performance Testing

**MEMORY** - Provides information about the amount of dynamic memory currently in use by the IDL session.

**PROFILER** - Accesses the IDL Code Profiler used to analyze performance of applications.

**TIME_TEST2** - Performs speed benchmarks for IDL.

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.

**IDLgrAxis** - Displays a plot axis object that may include tick marks, text, and a title.

**IDLgrContour** - Draws a contour plot from data stored in a rectangular array or from a set of unstructured points.

**IDLgrLegend** - Provides a simple interface for displaying a legend.

**IDLgrPlot** - Creates a set of polylines connecting data points in two-dimensional space.

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

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


**OPLOT** - 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.

**PLOT_3DBOX** - Plots function of two variables inside 3D box.

**PLOT_FIELD** - Plots a 2D field using arrows.

**PLOTERR** - Plots individual data points with error bars.

**PLOTS** - Plots vectors and points.

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

**POLYFILLV** - Returns subscripts of pixels inside a polygon.

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

**PROFILES** - Interactively examines image profiles.

**THREED** - Plots a 2D array as a pseudo 3D plot.

**TRIANGULATE** - Constructs Delaunay triangulation of a planar set of points.

**TRIGRID** - Interpolates irregularly-gridded data to a regular grid from a triangulation.

**USERSYM** - Defines a new plotting symbol.

**VEL** - Draws a velocity (flow) field plot.

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

**VEL`** - Defines a new plotting symbol.

**WF_DRAW** - Draws weather fronts with smoothing.

**XPLLOT3D** - Utility for creating and interactively manipulating 3D plots.

**XYOUTS** - Draws text on currently-selected graphics device.
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_MSBBLK - Defines and loads a new message block into the current IDL session.
DEFINE_MSBBLK_FROM_FILE - Reads the definition of a message block from a file, and loads it into the current IDL session.
DEFSYSV - Creates a new system variable.
EXECUTE - Compiles, executes IDL statements contained in a string.
EXIT - Quits IDL and exits back to the operating system.
EXPAND_PATH - Expands path-definition string into full path name for use with the 1PATH system variable.
HEAP_FREE - Recursively frees all heap variables referenced by its input argument.
HEAP_GC - Performs "garbage collection" on heap variables.
HEAP_NOSAVE - Used to clear the save attribute of pointer or object heap variables.
HEAP_SAVE - Used to query whether a pointer or object heap variable is savable. It can also be used to change the heap variable save attribute.
IDL_VALIDNAME - Determines whether a string may be used as a valid IDL variable name or structure tag name.
IDLTSYS_CREATETOOL - Creates an instance of the specified tool registered within the 1Tools system.
ITCURRENT - Set the current tool in the 1Tools system.
ITDELETE - Deletes a tool in the 1Tools system.
ITREGISTER - Gets the identifier of the current tool in the 1Tools system.
ITGETCURRENT - Gets the current IDL session.
ITRESOLVE - Resolves all IDL code within the 1Tools directory, as well as all other IDL code required for the 1Tools framework.
KEYWORD_SET - Determines whether its arguments are non-zero (or non-NULL).
LMGR - Determines the type of license used by the current IDL session.
LOGICAL_AND - Performs a logical AND operation on its arguments.
LOGICAL_OR - Performs a logical OR operation on its arguments.
LOGICAL_TRUE - Determines whether its arguments are non-zero (or non-NULL).
MESSAGE - Issues error and informational messages.
N_ELEMENTS - Returns the number of elements contained in an expression or variable.
N_PARAMS - Returns the number of non-keyword parameters used in calling an IDL procedure or function.
N_TAGS - Returns the number of tags in a structure.
OBJ_CLASS - Determines the class name of an object.
OBJ_DESTROY - Destroys an object reference.
OBJ_HASMETHOD - Determines whether an object class implements or inherits a specified method or methods.
OBJ_ISA - Determines inheritance relationship of an object.
OBJ_NEW - Creates an object reference.
OBJ_VALID - Verifies validity of object references.
ON_ERROR - Verifies validity of object references.
ON_IOERROR - Declares I/O error exception handler.
PATH_CACHE - Controls IDL's path caching mechanism.
PREF_COMMIT - Commits IDL preferences in the pending state.
PREF_GET - Returns information about IDL preferences.
PREF_MIGRATE - Imports IDL preferences from other versions of IDL for use by the currently running version.
PREF_SET - Sets new values for IDL preferences.
PTR_FREE - Destroys a pointer.
PTR_NEW - Creates a pointer.
PTR_VALIDATE - Verifies the validity of pointers.
PTRARR - Creates an array of pointers.
RECALL_COMMANDS - Returns entries in IDL's command recall buffer.
REGISTER_CURSOR - Associates a given name with cursor information, used in conjunction with IDLgrWindow::SetCurrentCursor.
RESOLVE_ALL - Compiles any uncompiled routines.
RESOLVE_ROUTINE - Compiles a routine.
RETAIL - Returns control to the main program level.
RETURN - Returns control to the next-higher program level.
ROUTINE_INFO - Provides information about compiled procedures and functions.
STOP - Stops the execution of a running program or batch file.
STRMESSAGE - Returns the text of a given error number.
STRUCT_ASSIGN - Uses “Relaxed Structure Assignment” to copy structures.
STRUCT_HIDE - Designates the error recovery method.
SWAP_ENDIAN - Reverses the byte ordering of scalars, arrays or structures.
SWAP_ENDIAN_INPLACE - 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.
TAG_NAMES - Returns the names of tags in a structure.
TEMPORARY - Returns a temporary copy of a variable, and sets the original variable to “undefined”.
WAIT - Suspends execution of an IDL program for a specified period.

Query Routines

QUERY_ASCII - Obtains information about an ASCII file.
QUERY_BMP - Obtains information about a BMP image file.
QUERY_DICOM - Tests file for compatibility with READ_DICOM.
QUERY_GIF - Obtains information about a GIF image file.
QUERY_IMAGE - Determines if a file is recognized as an image file.
QUERY_JPEG - Obtains information about a JPEG image file.
**Functional List of IDL Routines**

**Saving/Restoring a Session**


**QUERY_MRSID** - Obtains information about a MrSID image file.

**QUERY_PICT** - Obtains information about a PICT image file.

**QUERY_PNG** - Obtains information about a PNG image file.

**QUERY_PPM** - Obtains information about a PPM image file.

**QUERY_SRF** - Obtains information about an SRF image file.

**QUERY_TIFF** - Obtains information about a TIFF image file.

**QUERY_WAV** - Checks that the file is actually a .WAV file and that the READ_WAV function can read the data in the file.

**Saving/Restoring a Session**

**HEAP_NOSAVE** - Used to clear the save attribute of pointer or object heap variables.

**HEAP_SAVE** - Used to query whether a pointer or object heap variable is savable. It can also be used to change the heap variable save attribute.

**IDL_Savefile** - Object that provides complete query and restore capabilities for IDL SAVE files.

**JOURNAL** - Logs IDL commands to a file.

**RESTORE** - Restores IDL variables and routines in an IDL SAVE file.

**SAVE** - Saves variables, system variables, and IDL routines in a file for later use.

**Scientific Data Formats**

**CDF Routines** - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

**EOS Routines** - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

**H5_BROWSER** - Opens a GUI to view contents of HDF5 files.

**HDF Routines** - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

**HDF5 Routines** - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

**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.

**NCDF Routines** - see the Scientific Data Formats chapter of this manual in quickref.pdf, located on your IDL CD-ROM.

**Signal Processing**

**A_CORRELATE** - Computes autocorrelation.

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

**BUTTERWORTH** - Returns the absolute value of the low-pass Butterworth kernel.

**CANNY** - Implements the Canny edge-detection algorithm.

**C_CORRELATE** - Computes cross correlation.

**CONVOL** - Convolves two vectors or arrays.

**CORRELATE** - Computes the linear Pearson correlation.

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

**FFT** - Returns the Fast Fourier Transform of an array.

**HANNING** - Creates Hanning and Hamming windows.

**HILBERT** - Constructs a Hilbert transform.

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

**IR_FILTER** - Performs the infinite or finite impulse response filter on data.

**LEEFILT** - Performs the Lee filter algorithm on an image array.

**M_CORRELATE** - Computes multiple correlation coefficient.

**MEDIAN** - Returns median value of an array or applies a median filter.

**P_CORRELATE** - Computes partial correlation coefficient.

**R_CORRELATE** - Computes rank correlation.

**SAVGOL** - Returns coefficients of Savitzky-Golay smoothing filter.

**SMOOTH** - Smooths with a boxcar average.

**TS_COEF** - Computes the coefficients for autoregressive filter.

**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.

**WTN** - Returns wavelet transform of the input array.

See Also - Wavelet Toolkit

**Statements**

**Compound Statements**

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

**Conditional Statements**

**IF...THEN...ELSE** - Conditionally executes a statement or block of statements.

**CASE** - Selects one statement for execution, depending on the value of an expression.

**SWITCH** - Selects one statement for execution, depending upon the value of an expression.
Loop Statements

**FOR** - Executes one or more statements repeatedly, incrementing or decrementing a variable with each repetition, until a condition is met.

**REPEAT...UNTIL** - Repeats statement(s) until expression evaluates to true. Subject is always executed at least once.

**WHILE...DO** - Performs statement(s) as long as expression evaluates to true. Subject is never executed if condition is initially false.

Jump Statements

**BREAK** - Exits from a loop (FOR, WHILE, REPEAT), CASE, or SWITCH statement.

**CONTINUE** - Starts the next iteration of the enclosing FOR, WHILE, or REPEAT loop.

**GOTO** - Transfers program control to point specified by label.

Functions and Procedures

**COMPILE_OPT** - Gives IDL compiler information that changes the default rules for compiling functions or procedures.

**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).

**FUNCTION** - Defines a function.

**PRO** - Defines a procedure.

Variable Scope

**COMMON** - Creates a common block.

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.

**STRLEN** - 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.

**N_TAGS** - Returns the number of tags in a structure.

**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.

**TAG_NAMES** - Returns the names of tags in a structure.

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.

**IDLEXBR_ASSISTANT** - Launches the Export Bridge Assistant.

**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.

**XDXF** - 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.

**XLOADCT** - Displays a tool for selecting and setting the current color table.

**XMTOOL** - Displays a tool for viewing XMANAGER widgets.

**XOBJVIEW** - 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.
XPALLETTE - Displays a tool for creating and modifying color tables.
XPCOLOR - Adjusts the value of the current foreground plotting color, IP COLOR.
XPLOT3D - Utility for creating and interactively manipulating 3D plots.
XROI - Utility for interactively defining and obtaining information about regions of interest.
XSURFACE - Provides a graphical interface to the SURFACE and SHADE_SURF commands.
XVAREDIT - Utility for editing any IDL variable.
XVOLUME - Utility for viewing and interactively manipulating volumes and isosurfaces.
XVOLUME_ROTATE - Utility for rotating a volume displayed in XVOLUME.
XVOLUME_WRITE_IMAGE - Utility for writing a volume displayed in XVOLUME to an image file.

Wavelet Toolkit

Widget Commands and Visualization Tools

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_MULTITRIES - Runs 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.

Wavelet Toolkit

IDL Quick Reference
**Color Manipulation**

- **CW_ANIMATE_RUN** - Displays images loaded into CW_ANIMATE.
- **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_PALETTE_EDITOR_SET** - Sets the CW_LIGHT_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_RGBSLIDER** - Creates compound widget with sliders for adjusting RGB color values.

**Data Entry and Display**

- **CW_FIELD** - Creates a widget data entry field.
- **CW_FILESEL** - Creates compound widget for file selection.
- **CW_FORM** - Creates compound widget for creating forms.

**Image/Data Manipulation**

- **CW_DEFROI** - Creates compound widget used to define region of interest.
- **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_ZOOM** - Creates widget for displaying zoomed images.

**Orientation**

- **CW_ARCBALL** - Creates compound widget for intuitively specifying 3D orientations.
- **CW_ORIENT** - Creates compound widget used to interactively adjust the 3D drawing transformation.

**User Interface**

- **CW_BGROUP** - Creates button group for use as a menu.
- **CW_FSLIDER** - Creates slider that selects floating-point values.
- **CW_PDMENU** - Creates widget pulldown menus.

**Window Routines**

- **IDLgrWindow** - Represents an on-screen area on a display device that serves as a graphics destination.
- **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.
Alphabetical List of IDL Routines

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>{ } (Braces)</td>
<td>Indicates that you must choose one of the values they contain</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>
<tr>
<td>[ ] (Braces)</td>
<td>Encloses a list of possible values, separated by vertical lines (</td>
</tr>
<tr>
<td></td>
<td>(Vertical lines)</td>
</tr>
<tr>
<td>[ , Value1, ... , Value8]</td>
<td>Indicates that any number of values can be specified.</td>
</tr>
<tr>
<td>[ , Value1, ... , Value8]</td>
<td>Indicates the maximum number of values that can be specified.</td>
</tr>
</tbody>
</table>

#### 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.

---

**Alphabetical List of IDL Routines**

**IDL Quick Reference**
• Do not include square brackets in your statement unless the brackets are italicized. Consider the following syntax:

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

An example of a valid statement is:

```idl
R = KRIG2D( Z, X, Y, BOUNDS=[0,0,1,1] )
```

• Note that when `[ Value_1, ..., Value_n ]` is listed, you can specify any number of arguments. When an explicit number is listed, as in `[ Value_1, ..., 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:

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

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

```idl
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:

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

An example of a valid statement is:

```idl
Result = CVTTOBM( A, THRESHOLD=150 )
```

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

```idl
[, 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=pixels ]` 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.
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:

\[
\text{WIDGET\_CONTROL, mywidget, GET\_UVALUE = userval}
\]

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:

\[
\text{PLOT, [X], Y [, /DATA | , /DEVICE | , /NORMAL]}
\]

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

\[
\text{PLOT, SIN(A), /DEVICE}
\]

- 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 6.3.

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.
Result = ADAPT_HIST_EQUAL(Image [, CLIP=value] [, FCN=vector] [, NREGIONS=nregions] [, TOP=value])

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 [, /DIMENSIONS])

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

ASCII_TEMPLATE - Presents a GUI that generates a template defining an ASCII file format.
Result = ASCII_TEMPLATE([Filename] [, BROWSE_LINES=lines] [, CANCEL=variable] [, GROUP=widget_id])

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

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

ATAN - Returns the arc-tangent of X.
Result = ATAN(X [, /PHASE] [, Thread pool keywords]) or Result = ATAN(Y, X) [, Thread pool keywords]
AXIS - Draws an axis of the specified type and scale.

Graphics Keywords: [ ], CHARSIZE=value
[ TICKLEN=value] [, 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] TICKINTERVAL=[string|vector]
[ X | Y | Z] TICKINTERVAL=scalar
[ X | Y | Z] TICKNAME=string
[ X | Y | Z] TICKS=integer
[ X | Y | Z] TICK_GET=variable
[ X | Y | Z] TITLE=string
[ X | Y | Z] VALUE=value

BEGIN...END - Defines a block of statements.
BEGIN statements
END | ENDF | ENDELSE | ENDFOR | ENDFOR | ENDPAR | ENDWHILE

BESELY - Returns the Y Bessel function of order N for X.
Result = BESELY(X, N [, /DOUBLE] [, ITER=variable])

BETA - Returns the value of the beta function.
Result = BETA(X, 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.

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] )

BIT_FFS - Returns the index of the first bit (non-zero) in an integer.
Result = BIT_FFS(Value )

BIT_POPULATION - Returns the number of set (non-zero) bits in an integer.
Result = BIT_POPULATION(Value )

BLAS_AXPY - Updates existing array by adding a multiple of another array.
BLAS_AXPY, Y, 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 SWITCH statement.
BREAK

BREAKPOINT - Sets and clears breakpoints for debugging.

BROYDEN - Solves nonlinear equations using Broyden's method.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL_EXTERNAL</td>
<td>Calls a function in an external sharable object and returns a scalar value.</td>
</tr>
<tr>
<td>CALL_FUNCTION</td>
<td>Calls an IDL function.</td>
</tr>
<tr>
<td>CALL_METHOD</td>
<td>Calls an IDL object method.</td>
</tr>
<tr>
<td>CALDATE</td>
<td>Converts Julian date to month, day, year.</td>
</tr>
<tr>
<td>CALL_PROCEDURE</td>
<td>Calls an IDL procedure.</td>
</tr>
<tr>
<td>CANNY</td>
<td>Implements the Canny edge-detection algorithm.</td>
</tr>
<tr>
<td>CASE</td>
<td>Selects one statement for execution, depending on the value of an expression.</td>
</tr>
<tr>
<td>CATCH</td>
<td>Declares and clears exception handlers.</td>
</tr>
<tr>
<td>CD</td>
<td>Sets and/or changes the current working directory.</td>
</tr>
<tr>
<td>CDFF * Routines</td>
<td>See “CDF Routines” on page 132.</td>
</tr>
<tr>
<td>CEIL</td>
<td>Returns the closest integer greater than or equal to X.</td>
</tr>
<tr>
<td>CHEBYSHEV</td>
<td>Returns the forward or reverse Chebyshev polynomial expansion.</td>
</tr>
<tr>
<td>CHECK_MATH</td>
<td>Returns and clears accumulated math error status.</td>
</tr>
<tr>
<td>CHISQR_CVF</td>
<td>Computes cutoff value in a Chi-square distribution.</td>
</tr>
<tr>
<td>CHISQR_PDF</td>
<td>Computes Chi-square distribution function.</td>
</tr>
<tr>
<td>CHOLDC</td>
<td>Constructs Cholesky decomposition of a matrix.</td>
</tr>
<tr>
<td>CHOLSOL</td>
<td>Solves set of linear equations (use with CHOLDC).</td>
</tr>
<tr>
<td>CINDGEN</td>
<td>Returns a complex array with each element set to its subscript.</td>
</tr>
<tr>
<td>CIR_3PNT</td>
<td>Returns radius and center of circle, given 3 points.</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Closes the specified files.</td>
</tr>
<tr>
<td>BUTTERWORTH</td>
<td>Returns an array that contains the absolute value of the low-pass Butterworth kernel.</td>
</tr>
<tr>
<td>CALL_EXTERNAL</td>
<td>Calls an IDL object method.</td>
</tr>
<tr>
<td>CALL_FUNCTION</td>
<td>Calls an IDL function.</td>
</tr>
<tr>
<td>CALL_METHOD</td>
<td>Calls an IDL object method.</td>
</tr>
<tr>
<td>CALDATE</td>
<td>Converts Julian date to month, day, year.</td>
</tr>
<tr>
<td>CALL_PROCEDURE</td>
<td>Calls an IDL procedure.</td>
</tr>
<tr>
<td>CANNY</td>
<td>Implements the Canny edge-detection algorithm.</td>
</tr>
<tr>
<td>CASE</td>
<td>Selects one statement for execution, depending on the value of an expression.</td>
</tr>
<tr>
<td>CATCH</td>
<td>Declares and clears exception handlers.</td>
</tr>
<tr>
<td>CD</td>
<td>Sets and/or changes the current working directory.</td>
</tr>
<tr>
<td>CDFF * Routines</td>
<td>See “CDF Routines” on page 132.</td>
</tr>
<tr>
<td>CEIL</td>
<td>Returns the closest integer greater than or equal to X.</td>
</tr>
<tr>
<td>CHEBYSHEV</td>
<td>Returns the forward or reverse Chebyshev polynomial expansion.</td>
</tr>
<tr>
<td>CHECK_MATH</td>
<td>Returns and clears accumulated math error status.</td>
</tr>
<tr>
<td>CHISQR_CVF</td>
<td>Computes cutoff value in a Chi-square distribution.</td>
</tr>
<tr>
<td>CHISQR_PDF</td>
<td>Computes Chi-square distribution function.</td>
</tr>
<tr>
<td>CHOLDC</td>
<td>Constructs Cholesky decomposition of a matrix.</td>
</tr>
<tr>
<td>CHOLSOL</td>
<td>Solves set of linear equations (use with CHOLDC).</td>
</tr>
<tr>
<td>CINDGEN</td>
<td>Returns a complex array with each element set to its subscript.</td>
</tr>
<tr>
<td>CIR_3PNT</td>
<td>Returns radius and center of circle, given 3 points.</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Closes the specified files.</td>
</tr>
</tbody>
</table>
CLUST_WTS - Computes the cluster weights of an array for cluster analysis.
Result = CLUST_WTS(Array [, DOUBLE] [, N_CLUSTERS=value] [, N_ITERATIONS=value] [, 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 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])

CMYK_CONVERT - Converts color triples to and from RGB and CMYK.

COLOR_CONVERT - Converts colors to and from the RGB (Red Green Blue) and a number of other color spaces.

COLOR_EXCHANGE - Replaces image pixels of a given color with pixels of a new color.
Result = COLOR_EXCHANGE(Image, Color, ReplaceColor [, /THRESHOLD=value])

COLOR_QUAN - Converts true-color (24-bit) image to pseudo-color (8-bit) image.
Result = COLOR_QUAN(Image_R, Image_G, Image_B) or
Result = COLOR_QUAN(Image, FromColor, ToColor)

COLOR_RANGE_MAP - Maps all the pixels of an image to another set of pixels, using source and target ranges to control the mapping.
Result = COLOR_RANGE_MAP(Image, FromColor1, ToColor1, ToColor2)

COLORIZE_SAMPLE - Colorizes a greyscale image by matching luminance levels with an RGB sample table.
Result = COLORIZE_SAMPLE(Image, Sample)

COLOMAP_APPLICABLE - Determines whether the current visual class supports the use of a colormap.
Result = COLOMAP_APPLICABLE(redrawRequired)

COLOMAP_GRADIENT - Maps an image into a specified luminance-based gradient.
Result = COLOMAP_GRADIENT(Image, Gradient)

COLOMAP_ROTATION - Maps pixels within a given hue range to another hue range, using the HSV hue component.
Result = COLOMAP_ROTATION(Image, SrcAngleStart, SrcAngleStop, DstAngleStart, DstAngleStop [, /SOURCE_CW], [ /DEST_CW])

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

COMMAND_LINE_ARGS - returns string values supplied when the user starts IDL with the -arg or -args command line options.
Result = COMMAND_LINE_ARGS(COUNT=variable)

COMMON - Creates a common block.
COMMON Block_Name, Variable1, ..., Variable_n

COMPILE_OPT - Gives IDL compiler information that changes the default rules for compiling functions or procedures.
COMPILE_OPT opt1 [, opt2, ..., opt_n]
Note: opt_n can be IDL2, DEFINT32, HIDDEN, LOGICAL_PREDICATE, OBSOLETE, STRICTARR, or STRICTARR_SUBS

COMPLEX - Converts argument to complex type.
Result = COMPLEX(Real [, Imaginary] [, /DOUBLE] [, Thread pool keywords])
or
Result = COMPLEX(Expression, Offset, D1 [, ... , D8] [, /DOUBLE] [, Thread pool keywords])

COMPLEXARR - Creates a complex, single-precision, floating-point vector or array.
Result = COMPLEXARR(D1 [, ... , D8] [, /NOZERO])

COMPRESS - Computes the condition number of a square matrix.
Result = COND(A [, /DOUBLE] [, LNM=1 {0 | 1 | 2}])
CONGRID - Resamples an image to any dimensions.
Result = CONGRID( Array, X, Y, Z [, /CENTER]
[, CUBIC=value [-1 to 0]] [, /INTERP]
[, /MINUS_ONE])

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

CONSTRAINED_MIN - Minimizes a function using Generalized Reduced Gradient Method.
CONSTRAINED_MIN, X, Xbdn, Gbdn, Nobj, Gcomp,
Inform [, ESPTOP=value [, LIMSER=value
[, /MAXIMIZE] [, NSTOP=value] [, REPORT=filename
[, TITLE=string]]]

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

CONTOUR - Draws a contour plot.
CONTOUR, Z [, X, Y] [, C_CHARSIZE=value
[, C_CHARTHICK=integer [, C_COLORS=vector]
[, C_LABELS=vector [each element 0 or 1]]
[, C_LINESTYLE=vector [[/FILL], /CELL_FILL] |
[, C_ANNOTATION=vector_of_strings]
[, C_ORIENTATION=degrees [C_SPACING=value]]
[, C_THICK=vector [, /CLOSED] [, /DOWNHILL]]
[, /FOLLOW] [, /REGULAR] [, /ISOTROPIC]
[, LINES=vector [, NLEVELS=integer [1 to 60]]
[, MAX_VALUE=value [, MIN_VALUE=value
[, OVERPLOT]]]
[, /PATH_DATA_COORDS]
PATH_FILENAME=string, PATH_INFO=variable,
PATH_XY=variable ] [, TRIANGULATION=variable]
[, /PATH_DOUBLE] [, /XLOG] [, /YLOG]
[, ZAXIS=[{0 | 1 | 2 | 3 | 4}] [, /ZLOG]

Graphics Keywords: Accepts all PLOT graphics
keywords except for: LINESTYLE, PSYM, SYMSIZE.

CONVERT_COORD - Transforms coordinates to and from the coordinate systems supported by IDL.
Result = CONVERT_COORD(X [, Y [, Z]] [, /DATA
[, /DEVICE] [, /NORMAL] [, /DOUBLE] [, /T3D]
[, /T0_DATA] [, /T0_DEVICE] [, /T0_NORMAL])

CONVOL - Converts two vectors or arrays.
Result = CONVOL(Array, Kernel [, Scale_Factor]
[, BIAS=value [, /CENTER] [, /EDGE_WRAP]
[, /EDGE_TRUNCATE] [, /EDGE_ZERO
[, INVALID=value [, MISSING=value [, /NAN]]
[, /NORMALIZE] [, Thread pool keywords]])

COORD2TO3 - Returns 3D data coordinates given normalized screen coordinates.
Result = COORD2TO3(Mx, My, Dim, D0 [, PTH])

COPY_LUN - Copies data between two open files.
COPY_LUN, FromUnit, ToUnit [, Num] [, /EOF]
[, /LINES] [, TRANSFER_COUNT=value]

CORRELATE - Computes the linear Pearson correlation.
Result = CORRELATE(X [, Y] [, /COVARIANCE
[, /DOUBLE]])

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

COSH - Returns the hyperbolic cosine of X.
Result = COSH(X [, Thread pool keywords])

CPU - Changes the values stored in the read-only CPU system variable.
CPU [, /RESET] [, RESTORE=structure
[, TPOOL_MAX_ELTS = NumMaxElts]
[, TPOOL_MIN_ELTS = NumMinElts]
[, TPOOL_THREADS = NumThreads]
[, VECTOR_ENABLE]]

Cramer - Solves system of linear equations using Cramer’s rule.
Result = Cramer(A, B [, /DOUBLE] [, /ZERO=value])

CREATE_CURSOR - Returns a 16x16 image suitable for a cursor from the input string array.
Result = CREATE_CURSOR(StringArray
[, HOTSPOT=variable [, MASK=variable]])

CREATE_STRUCT - Creates and concatenates structures.
Result = CREATE_STRUCT(Tag1, Value1, ..., Tagn
Value[n], NAME=string)
or
Result = CREATE_STRUCT(Tag1, ..., Tagm, Value1, ..., Value[m]
NAME=string)

CREATE_VIEW - Sets up 3D transformations.
CREATE_VIEW [, AX=value [, AY=value [, AZ=value]
[, PERSP=value [, /RADIANS] [, WINX=pixels]
[, WNY=pixels [, XMAX=scalar [, XMIM=scalar]
[, YMAX=scalar [, YMIN=scalar [, ZFACTOR=value
[, ZMAX=scalar [, ZMIN=scalar [, ZOOM=scalar or
3-element vector]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]}
CURVEFIT - Fits multivariate data with a user-supplied function.
Result = CURVEFIT( X, Y, Weights, A [, Sigma]
[. CHISO=variable] [. /DOUBLE] [. FITA=vector]
[. FUNCTION_NAME=string] [. ITER=variable]
[. ITMAX=value] [. /NODERIVATIVE]
[. STATUS=[0 1 2]] [. TOL=value]
[. YERROR=variable] )

CW_ARCBALL - Creates compound widget for intuitively specifying 3D orientations.
Result = CW_ARCBALL( Parent, Names
[. BUTTON_UVALUE=variable] [. COLUMN=vector]
[. EVENT_FUNC=string] [. /EXCLUSIVE | , /NONEXCLUSIVE] [. /SPACE=pixels]
[. XOFFSET=pixels] [. YOFFSET=pixels] [. /FRAME=font]
[. FRAME=width] [. IDS=variable]
[. LABEL_LEFT=string] [. LABEL_TOP=string]
[. MAP] [. /NO_RELEASE] [. /RETURN_ID] [. /RETURN_INDEX] [. /RETURN_NAME]
[. ROW=value] [. /SCROLL] [. SET_VALUE=value]
[. SPACE=value] [. TAB_MODE=value]
[. X_SCROLL_SIZE=width]
[. Y_SCROLL_SIZE=height] [. SET_VALUE=value]
[. UNAME=string] [. UVALUE=value]
[. XOFFSET=value] [. XSIZE=width]
[. YOFFSET=value] [. YSIZE=value] )

CW_ANIMATE - Creates compound widget for animation.
Result = CW_ANIMATE( Parent, Sizex, Sizex, Nframes
[. MPEG_BITRATE=value] [. MPEG_FORMAT=value]
[. MPEQ_QUALITY=value] [. /NO_KILL]
[. OPEN_FUNC=string] [. PIXMAPS=vector]
[. TAB_MODE=value] [. /TRACK] [. UNAME=string]
[. UVALUE=value] )

CW_ANIMATE_GETP - Gets pixmap window IDs used by CW_ANIMATE.
CW_ANIMATE_GETP, Widget, Pixmaps
[. /KILL_ANYWAY]

CW_ANIMATE_LOAD - Loads images into CW_ANIMATE.
CW_ANIMATE_LOAD, Widget [, /CYCLE]
[. FRAME=value=0 to NFRAMES)] [. IMAGE=value]
[. /ORDER] [. WINDOW=window_num] [. X0, Y0, Sx, Sy] [. /NO_KILL]
[. XOFFSET=pixels] [. XSIZE=pixels] [. YOFFSET=pixels] [. YSIZE=pixels]

CW_ANIMATE_RUN - Displays images loaded into CW_ANIMATE.
CW_ANIMATE_RUN, Widget [. Rate=0 to 100]]
[. NFRAMES=variable] [. /STOP]

CW_ARCBALL - Creates compound widget for intuitively specifying 3D orientations.
Result = CW_ARCBALL( Parent, COLORS=vector
[. FRAME=pixels] [. NOEDIT] [. /RETURN_EVENTS]
[. /ROW] [. STRING=value] [. TAB_MODE=value]
[. /TEXT_FRAME] [. TITLE=string] [. UNAME=string]
[. UVALUE=value] [. VALUE=value]
[. XSIZE=characters] [. YSIZE=lines] )

CW_CLR_INDEX - Creates compound widget to select color index.
Result = CW_CLR_INDEX( Parent
[. COLOR_VALUES=vector] [. NCOLORS=variable]
[. START_COLOR=variable]
[. EVENT_FUNC='function_name'] [. /FRAME]

CW_CLR_INDEX - continued
[. LABEL=variable] [. NCOLORS=variable]
[. START_COLOR=variable] [. TAB_MODE=value]
[. UNAME=string] [. UVALUE=value]
[. VALUE=value] [. XSIZE=pixels] [. YSIZE=pixels]

CW_COORD - Converts 2D and 3D coordinates between coordinate systems.
Result = CW_COORD( [, /DEGREES] [, /DOUBLE]
[. FROM_CYLIN=vector] [. FROM_POLAR=vector]
[. FROM_RECT=vector] [. /TO_CYLIN]
[. /TO_POLAR] [. /TO_RECT] [. /TO_SPHERE] )

CUTTOBM - Creates a bitmap byte array for a button label.
Result = CUTTOBM( Array [, THRESHOLD=value=0 to 255])

CW_FILESEL - Creates compound widget for file selection.
Result = CW_FILESEL( Draw [, IMAGE_SIZE=vector]
[. OFFSET=vector] [. /ORDER] [. /RESTORE]
[. TAB_MODE=variable] [. ZOOM=vector])

CW_FIELD - Creates a widget data entry field.
Result = CW_FIELD( Parent [, /ALL_EVENTS]
[. FRAME=pixels] [. NOEDIT] [. /RETURN_EVENTS]
[. /ROW] [. STRING=value] [. TAB_MODE=value]
[. /TEXT_FRAME] [. TITLE=string] [. UNAME=string]
[. UVALUE=value] [. VALUE=value]
[. XSIZE=characters] [. YSIZE=lines] )

CW_FILESEL - Creates compound widget for file selection.
Result = CW_FILESEL( Parent [, /FILENAME]
[. FILTER=string array] [. /FIX_FILTER] [. /FRAME]
[. IMAGE_FILTER] [. MULTIPLE | , /SAVE]
[. PATH=string] [. TAB_MODE=value]
[. UNAME=string] [. UVALUE=value]
[. /WARN_EXIST] )

CURVEFIT
CW_LIGHT_EDITOR - Creates compound widget to edit properties of existing IDL Light objects in a view.

CW_LIGHT_EDITOR_GET - Gets the CW_LIGHT_EDITOR properties.
Result = CW_LIGHT_EDITOR_GET( WidgetID [, /COLUMN] )

CW_LIGHT_EDITOR_SET - Sets the CW_LIGHT_EDITOR properties.
Result = CW_LIGHT_EDITOR_SET( WidgetID [, /COLUMN] )

CW_PALETTE_EDITOR - Creates compound widget to display and edit color palettes.

CW_PALETTE_EDITOR_GET - Gets the CW_PALETTE_EDITOR properties.
Result = CW_PALETTE_EDITOR_GET( WidgetID [, /COLUMN] )

CW_PALETTE_EDITOR_SET - Sets the CW_PALETTE_EDITOR properties.
Result = CW_PALETTE_EDITOR_SET( WidgetID [, /COLUMN] )

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

CW_TMPL - Template for compound widgets that use XMANAGER.
Result = CW_TMPL( Parent [, TAB_MODE=value] [, UNAME=string] [, VALUE=value] )

CW_ZOOM - Creates widget for displaying zoomed images.

D

DBLARR - Creates a double-precision array.
Result = DBLARR( D1 [, ... , D8] [, /NOZERO] )

Alphabetical List of IDL Routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CW_FORM</td>
<td>Creates compound widget for creating forms.</td>
</tr>
<tr>
<td>CW_FSLIDER</td>
<td>Creates slider that selects floating-point values.</td>
</tr>
<tr>
<td>CW_LIGHT_EDITOR</td>
<td>Creates compound widget to edit properties of existing IDL Light objects in a view.</td>
</tr>
<tr>
<td>CW_LIGHT_EDITOR_GET</td>
<td>Gets the CW_LIGHT_EDITOR properties.</td>
</tr>
<tr>
<td>CW_LIGHT_EDITOR_SET</td>
<td>Sets the CW_LIGHT_EDITOR properties.</td>
</tr>
<tr>
<td>CW_PALETTE_EDITOR</td>
<td>Creates compound widget to display and edit color palettes.</td>
</tr>
<tr>
<td>CW_PALETTE_EDITOR_GET</td>
<td>Gets the CW_PALETTE_EDITOR properties.</td>
</tr>
<tr>
<td>CW_PALETTE_EDITOR_SET</td>
<td>Sets the CW_PALETTE_EDITOR properties.</td>
</tr>
<tr>
<td>CW_RGBSLIDER</td>
<td>Creates compound widget with sliders for adjusting RGB color values.</td>
</tr>
<tr>
<td>CW_TMPL</td>
<td>Template for compound widgets that use XMANAGER.</td>
</tr>
<tr>
<td>CW_ZOOM</td>
<td>Creates widget for displaying zoomed images.</td>
</tr>
<tr>
<td>DLBLARR</td>
<td>Creates a double-precision array.</td>
</tr>
</tbody>
</table>

IDL Quick Reference
DCINDGEN - Returns a double-precision, complex array with each element set to its subscript.
Result = DCINDGEN( D1 [, ..., Dn] [ , Thread pool keywords] )

DCOMPLEX - Converts argument to double-precision complex type.
Result = DCOMPLEX( Real [, Imaginary] [ , Thread pool keywords] )
or
Result = DCOMPLEX( Expression, Offset, D1 [, ..., Dn] [ , Thread pool keywords] )

DCOMPLEXARR - Creates a complex, double-precision vector or array.
Result = DCOMPLEXARR( D1 [, ..., Dn] [ , NOZERO] )

DEFINE_KEY - Programs keyboard function keys.
DEFINE_KEY. Key [, Value] [, MATCH_PREVIOUS] [, NOECHO] [, TERMINATE]
UNIX Keywords: [/BACK_CHARACTER] [/BACK_WORD] [/CONTROL] [/ESCAPE]
[/DELETE_WORD] [/END_OF_FILE] [/ENTER_LINE]
[/FORWARD_CHARACTER] [/FORWARD_WORD]
[/INSERT_OVERFLOW] [/INSERTOVERSTRIKE_TOGGLE]
[/NEXT_LINE] [/PREVIOUS_LINE] [/RECALL]
[/REDRAW] [/START_OF_LINE]

DEFINE_MSGBLK - Defines and loads a new message block into the current IDL session.
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.
DEFINE_MSGBLK_FROM_FILE. Filename [ , BLOCK=BlockName] [ , IGNORE_DUPLICATE]
[ , PREFIX=PrefixStr] [ , VERBOSIZE]

DEFROI - Defines an irregular region of interest of an image.
Result = DEFROI( X1, Y1, X2, Y2) [ , /NOREGION]

DEFSYSV - Creates a new system variable.
DEFSYSV.Name, Value [ , Read_Only] [ , EXISTS=variable]

DELVAR - Deletes variables from the main IDL program level.
DELVAR. V1 [, V2]

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. Clusters, Linkdistance
[ , LABEL_CHARSIZE=value] [ , LABEL_CHARTHICK=value]
[ , LABEL_COLOR=value] [ , LABEL_NAMES=vector]
[ , LABEL_ORIENTATION=value]
[ , LINESCOLOR=value] [ , LINESSTYLE=value]
[ , ORIENTATION=1 | 2 | 3 | 4] [ , 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. Clusters, Linkdistance, Outverts,
Outcomm [ , LEAFNODES=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, Sig_x, Sig_y, )

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.
DEVICE [ , /AVANTGARDE] [ , /BMAN] [ , /COURIER] [ , /HELVETICA] [ , /ISOLATINO] [ , /LATINO] [ , /SCHOOLBOOK] [ , /SYMBOL] [ , TIMES]
ZAPPCARTHCURVE] [ , ZAPFDINGBATS [ , PS]] [ , /AVERAGE_LINES[REGIS]] [ , /BINARY] [ , /CAR]
, /TEXT[CGM]] [ , /BITS_PER_PIXEL=1 | 2 | 4 | 8][PS]] [ , /BOLD[PS]] [ , /BOOK[PS]]
, /BYPASS_TRANSLATION[WIN, X]] [ , /CLOSE[Z]]
, /CLOSE_DOCUMENT[PRINTER]]
, /CLOSE_FILE[CGM, HP, METAFI, PCL, PS, REGIS, TEK]] [ , /CMYK[PS]] [ , /COLOR[CPL, CPS]]
, /COLORS=value[CGM, TEK]] [ , COPY=] [ , Xsource, Ysource, cols, rows, Xdest, Ydest]
[ , Window_index][WIN, X]]
, /CURSOR_FROM_SHELL[WIN, X]]
, /CURSOR_IMAGE=value[16-element short int vector][WIN, X]]
, /CURSOR_Mask=value[WIN, X]]
, /CURSOR_FROM_ORIGINAL[WIN, X]]
, /CURSOR_STANDARD=value[WIN, arrow=32512, I-beam=32513, hourglass=32514, black cross=32515, up arrow=32516, size(N)=32640, icon(N)=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=] [
, /DECOMPOSED[WIN, X]]
, /DIRECT_COLOR[X]]]
, /EJECT=[0 | 1 | 2][HP]]
, /ENCAPSULATED=[0 | 1][PS]]
, /ENCODING=1 (binary) [ , 2 (text) [ , 3 (NCR binary)] [CGM]]
, /FILENAME=filename[CGM, HP, METAFI, PCL, PS, REGIS, TEK]]
, /FLOYD[PCL, X]]
, /FONT_INDEX=integer[PS]]
Alphabetical List of IDL Routines

DIALOG_READ_IMAGE - Presents GUI for reading image files.

DIALOG_READ_IMAGE[Filename],[ DIALOG_PARENT=widget_id], [FILE=variable]
Alphabetical List of IDL Routines

DIALOG_WRITE_IMAGE - Draws region or group of regions to current Direct Graphics device.

EDGE_DOG - Applies the Difference of Gaussians filter to a 2D image array.

EIGENQL - Computes eigenvalues and eigenvectors of a real, symmetric array.

EIGENVEC - Computes eigenvectors of a real, non-symmetric array.

ELMHES - Reduces nonsymmetric array to upper Hessenberg form.

EMBOSS - Applies a convolution operator to a 2D image array to generate an array.

EMPTY - Empties the graphics output buffer.

ENABLE_SYSRTN - Enables/disables IDL system routines.

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

EOS_Routines - See “EOS Routines” on page 134 of the quick-ref.pdf located on your IDL CD-ROM.
FILE_INFO - Returns status information about a file.
   Result = FILE_INFO(Path [, /NOEXPAND_PATH] )

FILE_LINES - Returns the number of lines of text in a file.
   FILE_LINES(Path [, /COMPRESS] [, /NOEXPAND_PATH] )

FILE_LINK - Creates UNIX file links.

FILE_MKDIR - Creates a new directory, or directories, with default access permissions for the current process.
   FILE_MKDIR, File [ ... FileN] [, /NOEXPAND_PATH] ...

FILE_MOVE - Renames files and directories.

FILE_POLL_INPUT - Blocks processing until it detects that a read operation on a specified file will succeed.
   Result = FILE_POLL_INPUT(Units [, COUNT=value] [, TIMEOUT=value] )

FILE_READLINK - Returns the path pointed to by a UNIX symbolic link.
   FILE_READLINK(Path [, /ALLOW_NEXISTENT] [, /NOEXPAND_PATH] )

FILESAME - Determines whether two different file names refer to the same underlying file.
   Result = FILESAME(Path1, Path2 [, /NOEXPAND_PATH] )

FILE_SEARCH - Returns a string array containing the names of all files matching the input path specification.
   Result = FILE_SEARCH(Path, Specification) or

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


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.

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

FINITE - Returns True if its argument is finite.

FIX - Converts argument to integer type, or type specified by TYPE keyword.
   Result = FIX( Expression [, Offset [, D1 [, ..., D8]]] [, /PRINT] [, TYPE=type code[0 to 15]] [, Thread pool keywords] )

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]]] [, /PRINT] [, /NOZERO] [, Thread pool keywords] )

FLOOR - Returns closest integer less than or equal to argument.
   Result = FLOOR(X [, /L64] [, Thread pool keywords] )

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

FLTARR - Returns a single-precision, floating-point vector or array.
   Result = FLTARR( D1 [, ..., D8] [, /NOZERO] )

FLUSH - Flushes file unit buffers.
   FLUSH, Unit1, ..., UnitN

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
Alphabetical List of IDL Routines

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 [ , Units1, ..., UnitsN ]
[ , 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 - Evaluates sum of a Gaussian and a 2nd-order polynomial and returns value of its partial derivatives.
FUNCTION X, A, F [, PDer]

FUNCTION - Defines a function.
FUNCTION Function_Name, parameter1, ..., parameterN

FX_ROOT - Computes real and complex roots of a univariate nonlinear function using an optimal Muller’s method.
Result = FX_ROOT(X, Func [, , /DOUBLE] [ , TMAX=value ] [ , /STOP ] [ , TOL=value ] )

FZ_ROOTS - Finds roots of a complex polynomial using Laguerre’s method.
Result = FZ_ROOTS( C [, , /DOUBLE] [ , EPS=value ] [ , /NO_POLISH ] )

G

GAUSSFIT - Fits the sum of a Gaussian and a quadratic.
Result = GAUSSFIT( X, Y, A[, , /CHISQ=variable] [ , ESTIMATES=vector ] [ , MEASURE_ERRORS=vector ] [ , NTERMS=integer [3 to 6]] [ , SIGMA=variable ] [ , YERROR=variable ] )

GAMMA - Returns the gamma function of Gamma.
Result = GAMMA( Z )

GAMMA_CVF - Computes cutoff value in Gaussian distribution.
Result = GAMMA_CVF(P)

Gauss_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( X )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [, , /NEGATIVE] [ , /TILT ] )

GaussFIT - Fits the sum of a Gaussian and a quadratic.
Result = GAUSSFIT( X, Y, A[, , /CHISQ=variable] [ , ESTIMATES=vector ] [ , MEASURE_ERRORS=vector ] [ , NTERMS=integer [3 to 6]] [ , SIGMA=variable ] [ , YERROR=variable ] )

GAMMA - Returns the gamma function of Gamma.
Result = GAMMA( Z )

GAMMA_CVF - Computes cutoff value in Gaussian distribution.
Result = GAMMA_CVF(P)

Gauss_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( X )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [, , /NEGATIVE] [ , /TILT ] )

GaussFIT - Fits the sum of a Gaussian and a quadratic.
Result = GAUSSFIT( X, Y, A[, , /CHISQ=variable] [ , ESTIMATES=vector ] [ , MEASURE_ERRORS=vector ] [ , NTERMS=integer [3 to 6]] [ , SIGMA=variable ] [ , YERROR=variable ] )

GAMMA - Returns the gamma function of Gamma.
Result = GAMMA( Z )

GAMMA_CVF - Computes cutoff value in Gaussian distribution.
Result = GAMMA_CVF(P)

Gauss_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( X )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [, , /NEGATIVE] [ , /TILT ] )

GaussFIT - Fits the sum of a Gaussian and a quadratic.
Result = GAUSSFIT( X, Y, A[, , /CHISQ=variable] [ , ESTIMATES=vector ] [ , MEASURE_ERRORS=vector ] [ , NTERMS=integer [3 to 6]] [ , SIGMA=variable ] [ , YERROR=variable ] )

GAMMA - Returns the gamma function of Gamma.
Result = GAMMA( Z )

GAMMA_CVF - Computes cutoff value in Gaussian distribution.
Result = GAMMA_CVF(P)

Gauss_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( X )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [, , /NEGATIVE] [ , /TILT ] )

GaussFIT - Fits the sum of a Gaussian and a quadratic.
Result = GAUSSFIT( X, Y, A[, , /CHISQ=variable] [ , ESTIMATES=vector ] [ , MEASURE_ERRORS=vector ] [ , NTERMS=integer [3 to 6]] [ , SIGMA=variable ] [ , YERROR=variable ] )

GAMMA - Returns the gamma function of Gamma.
Result = GAMMA( Z )

GAMMA_CVF - Computes cutoff value in Gaussian distribution.
Result = GAMMA_CVF(P)

Gauss_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( X )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [, , /NEGATIVE] [ , /TILT ] )

GaussFIT - Fits the sum of a Gaussian and a quadratic.
Result = GAUSSFIT( X, Y, A[, , /CHISQ=variable] [ , ESTIMATES=vector ] [ , MEASURE_ERRORS=vector ] [ , NTERMS=integer [3 to 6]] [ , SIGMA=variable ] [ , YERROR=variable ] )

GAMMA - Returns the gamma function of Gamma.
Result = GAMMA( Z )

GAMMA_CVF - Computes cutoff value in Gaussian distribution.
Result = GAMMA_CVF(P)

Gauss_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( X )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [, , /NEGATIVE] [ , /TILT ] )

GaussFIT - Fits the sum of a Gaussian and a quadratic.
Result = GAUSSFIT( X, Y, A[, , /CHISQ=variable] [ , ESTIMATES=vector ] [ , MEASURE_ERRORS=vector ] [ , NTERMS=integer [3 to 6]] [ , SIGMA=variable ] [ , YERROR=variable ] )

GAMMA - Returns the gamma function of Gamma.
Result = GAMMA( Z )

GAMMA_CVF - Computes cutoff value in Gaussian distribution.
Result = GAMMA_CVF(P)

Gauss_PDF - Computes Gaussian distribution function.
Result = GAUSS_PDF( X )

GAUSS2DFIT - Fits a 2D elliptical Gaussian equation to rectilinearly gridded data.
Result = GAUSS2DFIT( Z, A [, , X, Y ] [, , /NEGATIVE] [ , /TILT ] )
Alphabetical List of IDL Routines

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]
  [, LAMDA=value] [, 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[ , Image]

H_EQ_INT - Interactively histogram-equalizes the color tables of an image or a region of the display.
  Result = H_EQ_INT[ , Image]

HANNING - Creates Hanning and Hamming windows.
  Result = HANNING( N [ , N_2] [, ALPHA=value] [0.5 to 1.0]) [, /DOUBLE]

HDF_* Routines - See "HDF Routines" on page 138 the quick-ref.pdf located on your IDL CD-ROM.

HDF5_* Routines - See "HDF5 Routines" on page 144 the quick-ref.pdf located on your IDL CD-ROM for the new H5_* routines.

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 meta-data into an output structure.
  Result = HDF_READ( [Filename] [, DFR8=variable ]
  [, DF24=variable ] [, TEMPLATE=variable ]

HEAP_FREE - Recursively frees all heap variables referenced by its input argument.
  HEAP_FREE, Var [, /OBJ] [, /PTR] [, /VERBOSE]

HEAP_GC - Performs garbage collection on heap variables.
  HEAP_GC [, /OBJ] [, /PTR] [, /VERBOSE]

HEAP_NOSAVE - Used to clear the save attribute of pointer or object heap variables.
  HEAP_NOSAVE, HeapVar

HEAP_SAVE - Used to query whether a pointer or object heap variable is savable. It can also be used to change the heap variable save attribute.
  Result = HEAP_SAVE(HeapVar [, Set])

HELP - Provides information about the current IDL session.
  HELP, Expression[, ... , Expression_n] [, /BREAKPOINTS]
  [, /FUNCTIONS] [, /HEAP_VARIABLES] [, /KEYS]
  [, /LAST_MESSAGE] [, /LEVEL=value] [, /MEMORY]
  [, /MESSAGES] [, NAMES=string_of_variable_names]
  [, /OBJECTS] [, OUTPUT=variable] [, /PATH_CACHE]
  [, /PREFERENCES] [, /PROCEDURES]
  [, /RECALL_COMMANDS] [, /Routines]
  [, /SHARED_MEMORY] [, /SOURCE_FILES]

HELP - continued
  [, /STRUCTURES] [, /SYSTEM_VARIABLES]
  [, /TRACEBACK]

HILBERT - Constructs a Hilbert transform.
  Result = HILBERT( X [, D] )

HIST_2D - Returns histogram of two variables.
  Result = HIST_2D( V_1, V_2, Bin1=width] [, Bin2=height]
  [, MAX1=value] [, MAX2=value] [, MIN1=value]
  [, MIN2=value] )

HIST_EQUAL - Histogram-equalizes an image.
  Result = HIST_EQUAL( A [, BINSIZE=value]
  [, FCN=vector] [, /HISTOGRAM_ONLY]
  [, MAX=value] [, MIN=value] [, OMN=value]
  [, OMIN=value] [, PERCENT=value] [, TOP=value] )

HISTOGRAM - Computes the density function of an array.
  Result = HISTOGRAM( Array [, BINSIZE=value]
  [, INPUT=variable] [, LOCATIONS=variable]
  [, MAX=value] [, MIN=value] [, NaN]
  [, Nbins=value] [, OMX=value] [, OMN=value]
  [, OMIN=value] [, /L64] [, REVERSE_INDICES=variable] )

HLS - Creates color table in Hue, Lightness, Saturation color system.
  HLS, Litlo, X, Satlo, Satli, Hue, Loops [, Colr]

HOUGH - Returns the Hough transform of a two-dimensional image.
  Hough Transform: Result = HOUGH( Array
  [, /DOUBLE] [, DRHO=scalar] [, D=scalar]
  [, DY=scalar] [, /GRAY] [, NRHO=scalar]
  [, NTHETA=scalar] [, RHO=variable] [, RMIN=scalar]

Hough Backprojection: Result = HOUGH( Array,
  /BACKPROJECT, RHO=variable, THETA=variable
  [, /DOUBLE] [, RN=scalar] [, DY=scalar]
  [, DX=scalar] [, NY=scalar] [, XMIN=scalar]
  [, YMIN=scalar] )

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.

| Result = IBETA(A, B, Z[, /DOUBLE] [, EPS=value] [, ITER=value] [, ITMAX=value]) |

IBETA - Computes the incomplete beta function.

| Result = IBETA(A, B, Z[, /DOUBLE] [, EPS=value] [, ITER=value] [, ITMAX=value]) |

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

| Axis Object Keywords: [ , {X | Y | Z}GRIDSTYLE={0 | 1 | 2 | 3 | 4 | 5 | 6} , {X | Y | Z}MAJOR=integer , {X | Y | Z}MINOR=integer |

ICONTOUR Axis Object Keywords - continued

| [ , {X | Y | Z}GRIDSTYLE={0 | 1 | 2 | 3 | 4} , {X | Y | Z}MAJOR=integer , {X | Y | Z}MINOR=integer |

IDENTITY - Returns an identity array.

| IDENTITY = IDENTITY(N[, /DOUBLE] ) |

IDL_Container Object - See “IDL_Container” on page 78.

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

| IDL_ValidationObject = IDL_ValidationObject(String[, /CONVERT_ALL] [, CONVERT_SPACES]) |

IDLanROI Object - See “IDLanROI” on page 79.

IDLanROIGroup Object - See “IDLanROIGroup” on page 80.

IDLEXBR_ASSISTANT - Launches the Export Bridge Assistant.


IDLIDICOM Object - See “IDLIDICOM” on page 80.

IDLIDicomEx Object - See Appendix B, “IDL DICOM Quick Reference” in the Medical Imaging in IDL manual.

IDLIDicomExCmg Object - See Appendix B, “IDL DICOM Quick Reference” in the Medical Imaging in IDL manual.

IDLIDicomExQuery Object - See Appendix B, “IDL DICOM Quick Reference” in the Medical Imaging in IDL manual.

IDLIDicomExStorScu Object - See Appendix B, “IDL DICOM Quick Reference” in the Medical Imaging in IDL manual.

IDLIDXF Object - See “IDLIDXF” on page 81.


IDLLangCat Object - See “IDLLangCat” on page 82.

IDLIMJPEG2000 Object - See “IDLIMJPEG2000” on page 82.
IDLffMrSID Object - See "IDLffMrSID" on page 83.
IDLffShape Object - See "IDLffShape" on page 83.
IDLffXMLDOM - See "IDLffXMLDOM Classes" on page 84.
IDLffXMLSAX Object - See "IDLffXMLSAX" on page 89.
IDLit* Objects - IDLit* objects and their methods are described starting with "IDLitCommand" on page 107.
IDLITSYS_CREATETOOL - Creates an instance of the specified tool registered within the iTools system.

Result = IDLITSYS_CREATETOOL( StrTool )
[ . ANISTROPIC_SCALE_2D=value ]
[ . ANISTROPIC_SCALE_3D=value ]
[ . DIMENSIONS=[width, height] ]
[ . /DISABLE_SPLASH SCREEN ]
[ . IDENTIFIER=variable ]
[ . INITIAL_DATA=data ]
[ . LOCATION=[x, y] ]
[ . MACRO_NAME=string or string array ]
[ . NO_SAVEPROMPT ]
[ . OVERPLOT=[ToolID] ]
[ . RENDERER={0 | 1} ]
[ . USER_INTERFACE=string ]
[ . VIEW_GRID=vector ]
[ . VIEW_NEXT ]
[ . VIEW_NUMBER=number ]
[ . VISUALIZATION_TYPE=avisistype ]

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 ]

IGAMMA - Computes the incomplete gamma function.


IIMAGE - Creates an iTool and associated user interface (UI) configured to display and manipulate image data.

IMAGE[Image, [X, Y]]
iTool Common Keywords:
[ . ANISTROPIC_SCALE_2D=value ]
[ . ANISTROPIC_SCALE_3D=value ]
[ . BACKGROUND_COLOR=value ]
[ . DIMENSIONS=[x, y] ]
[ . /DISABLE_SPLASH SCREEN ]
[ . /FIT_TO_VIEW ]
[ . IDENTIFIER=variable ]
[ . LOCATION=[x, y] ]
[ . MACRO_NAME=string or string array ]
[ . NAME=string ]
[ . /NO_SAVEPROMPT ]
[ . OVERPLOT=[ToolID] ]
[ . RENDERER={0 | 1} ]
[ . SCALE_ISOTROPIC=value ]
[ . STYLE_NAME=string ]
[ . TITLE=string ]
[ . VIEW_GRID=[columns, rows] ]
[ . VIEW_NUMBER=integer ]
[ . VIEW_TITLE=string ]
[ . VIEW_ZOOM=value ]

Image Keywords:
[ . ALPHA_CHANNEL=2-D array ]
[ . BLUE_CHANNEL=2-D array ]
[ . GREEN_CHANNEL=2-D array ]
[ . IMAGE_DIMENSIONS=[width, height] ]
[ . IMAGE_LOCATION=[x, y] ]
[ . INSERT_COLORBAR=value ]
[ . RED_CHANNEL=2-D array ]
[ . RGB_TABLE=byte array of 256 by 3 or 3 by 256 elements ]
[ . TRANSPARENCY=value ]
[ . ZVALUE=value ]

Image Object Keywords:
[ . BLEND_FUNCTION=vector ]
[ . CHANNEL=hexadecimal bitmask ]
[ . CLIP_PLANES=arrays ]
[ . /HIDE ]
[ . /INTERPOLATE ]
[ . /ORDER ]
[ . SUB_RECT=[x, y, xdim, ydim] ]
[ . WINDOW_SCALE ]

IMAGE_STATISTICS - Computes sample statistics for a given array of values.

IMAGE_STATISTICS, Data [, /Labeled ]
[ . /WEIGHTED ]
[ . WEIGHT_SUM=variable ]
[ . /VECTOR ]
[ . LUT=variable ]
[ . MASK=variable ]
[ . STDDEV=variable ]
[ . DATA_SUM=variable ]
[ . SUM_OF_SQUARES=variable ]
[ . MINIMUM=variable ]
[ . MAXIMUM=variable ]
[ . VARIANCE=variable ]

IMAGINARY - Returns the imaginary part of a complex value.

Result = IMAGINARY(Complex_Expression, Thread pool keywords)

IMAP - Creates an iTool and associated user interface (UI) configured to display and manipulate map data.

IMAP[ , MAP_PROJECTION=string ]
or
IMAP{ Image, [ X, Y ] }, GEOFF={variable} [ , GRID_UNITS=value ]
[ , MAP_PROJECTION=string ]
or
IMAP{ Z, [ X, Y ] }, /CONTOUR [ , GRID_UNITS=value ]
[ , MAP_PROJECTION=string ]
iTool Common Keywords:
[ . ANISTROPIC_SCALE_2D=value ]
[ . ANISTROPIC_SCALE_3D=value ]
[ . BACKGROUND_COLOR=value ]
[ . DIMENSIONS=[x, y] ]
[ . /DISABLE_SPLASH SCREEN ]
[ . /FIT_TO_VIEW ]
[ . IDENTIFIER=variable ]
[ . LOCATION=[x, y] ]
[ . MACRO_NAME=string or string array ]
[ . NAME=string ]
[ . /NO_SAVEPROMPT ]
[ . OVERPLOT=[ToolID] ]
[ . RENDERER={0 | 1} ]
[ . SCALE_ISOTROPIC=value ]
**Alphabetical List of IDL Routines**

- **INDGEN** - Return an integer array with each element set to its subscript.
  
  ```idl```
  ```
  Result = INDGEN(D1, ... , Dn, [ , BYT E = /BYTE ] , / COMPLEX ] ,
  [ , TYPE=value ] [ , Thread pool keywords ]
  ```

- **INT_2D** - Computes the double integral of a bivariate function.
  
  ```idl```
  ```
  Result = INT_2D( Fxy, AB_Limits, PQ_Limits, Pts
  [ , / DOUBLE ] [ , / ORDER ]
  ```

- **INT_3D** - Computes the triple integral of a trivariate function.
  
  ```idl```
  ```
  Result = INT_3D( Fxyz, AB_Limits, PQ_Limits, UV_Limits, Pts
  [ , / DOUBLE ]
  ```

- **INT_TABULATED** - Integrates a tabulated set of data.
  
  ```idl```
  ```
  Result = INT_TABULATED( X, F [ , / DOUBLE ]
  ```

- **INTERP** - Performs linear interpolation on vectors.
  
  ```idl```
  ```
  For regular grids: Result = INTERPOL( V, N
  [ , / LSQUADRIC ] [ , / QUADRIC ] [ , / SPLINE ] )
  ```

- **INTERPOLATE** - Returns an array of interpolates.
  
  ```idl```
  ```
  Result = INTERPOLATE( P, X, Y, Z
  [ , / CUBIC=value [-1 to 0] ] [ , / GRID ] [ , / M I S S I N G=value]
  [ , / DOUBLE ]
  ```

- **INTERVAL_VOLUME** - Generates a tetrahedral mesh from volumetric data.
  
  ```idl```
  ```
  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=variable
  [ , / PROGRESS_PERCENT=percent [0 to 100]
  [ , / PROGRESS_USERDATA=value
  ```

- **INVERT** - Computes the inverse of a square array.
  
  ```idl```
  ```
  Result = INVERT( Array [ , Status ] [ , / DOUBLE ]
  ```

- **IOCTL** - Performs special functions on UNIX files.
  
  ```idl```
  ```
  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 ]
  ```

- **IPL O T** - Creates an iTool and associated user interface (UI) configured to display and manipulate plot data.
  
  ```idl```
  ```
  I PLOT [ , X, Y, Z
  or
  I PLOT, [ X, Y, Z
  ```
IR_FILTER - Filters data with an infinite impulse response (IIR) or finite impulse response (FIR) filter.

Result = IR_FILTER( C [, D], Data [, /DOUBLE] )

ISHFT - Performs integer bit shift.

Result = ISHFT( P1, P2 [, /Thread pool keywords] )

ISOCONTOUR - Interprets the contouring algorithm found in the IDLgrContour object.

ISOCONTOUR, Values, Outverts, Outconn

[ , AUXDATA_IN=variable, AUXDATA_OUT=variable ]
[ , C_LABEL_INTERVAL=vector of values ]
[ , C_LABEL_SHOW=vector of integers ]
[ , C_LABEL_SHOW=vector of integers ]
[ , N_LEVELS=levels ]
[ , OUT_LABEL_OFFSETS=variable ]
[ , OUT_LABEL_POLYLINES=variable ]
[ , OUT_LABEL_STRINGS=variable ]
[ , OUTCONN_INDICES=variable ]
[ , POLYGONS=array of polygon descriptions ]

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

ISOSURFACE, Data, Value, Outverts, Outconn

[ , AUXDATA_IN=variable, AUXDATA_OUT=variable ]

ISOSURFACE - continued

[ , GEOM_XYZ=array, TETRAHEDRA=array ]
[ , PROGRESS_CALLBACK=string ]
[ , PROGRESS_METHOD=string ]
[ , PROGRESS_OBJECT=objectref ]
[ , PROGRESS_PERCENT=percent(0 to 100) ]
[ , PROGRESS_USERDATA=variable ]

ISOSURFACE - continued

[ , ANISTROPIC_SCALE_2D=value ]
[ , ANISTROPIC_SCALE_3D=value ]
[ , BACKGROUND_COLOR=value ]
[ , DIMENSIONS=[x, y] ]
[ , /DISABLE_SPLASH_SCREEN] [/FIT_TO_VIEW]
[ , IDENTIFIER=variable] [ , LOCATION=[x, y] ]
[ , MACRO NAMES=string or string array ]
[ , NAME=string] [ , /NO_SAVEPROMPT]
[ , OVERPLOT=ToolID] [ , SCALE ISOTROPIC=value ]
[ , STYLE_NAME=string] [ , TITLE=string ]
[ , VIEW_GRID=[columns, rows]] [ , /VIEW_NEXT]

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

Result = IR_FILTER( C [, D], Data [, /DOUBLE] )

ISHFT - Performs integer bit shift.

Result = ISHFT( P1, P2 [, /Thread pool keywords] )

ISOCONTOUR - Interprets the contouring algorithm found in the IDLgrContour object.

ISOCONTOUR, Values, Outverts, Outconn

[ , AUXDATA_IN=variable, AUXDATA_OUT=variable ]
[ , C_LABEL_INTERVAL=vector of values ]
[ , C_LABEL_SHOW=vector of integers ]
[ , C_LABEL_SHOW=vector of integers ]
[ , N_LEVELS=levels ]
[ , OUT_LABEL_OFFSETS=variable ]
[ , OUT_LABEL_POLYLINES=variable ]
[ , OUT_LABEL_STRINGS=variable ]
[ , OUTCONN_INDICES=variable ]
[ , POLYGONS=array of polygon descriptions ]

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

ISOSURFACE, Data, Value, Outverts, Outconn

[ , AUXDATA_IN=variable, AUXDATA_OUT=variable ]

ISOSURFACE - continued

[ , GEOM_XYZ=array, TETRAHEDRA=array ]
[ , PROGRESS_CALLBACK=string ]
[ , PROGRESS_METHOD=string ]
[ , PROGRESS_OBJECT=objectref ]
[ , PROGRESS_PERCENT=percent(0 to 100) ]
[ , PROGRESS_USERDATA=variable ]

ISOSURFACE - continued

[ , ANISTROPIC_SCALE_2D=value ]
[ , ANISTROPIC_SCALE_3D=value ]
[ , BACKGROUND_COLOR=value ]
[ , DIMENSIONS=[x, y] ]
[ , /DISABLE_SPLASH_SCREEN] [/FIT_TO_VIEW]
[ , IDENTIFIER=variable] [ , LOCATION=[x, y] ]
[ , MACRO NAMES=string or string array ]
[ , NAME=string] [ , /NO_SAVEPROMPT]
[ , OVERPLOT=ToolID] [ , SCALE ISOTROPIC=value ]
[ , STYLE_NAME=string] [ , TITLE=string ]
[ , VIEW_GRID=[columns, rows]] [ , /VIEW_NEXT]
ALPHABETICAL LIST OF IDL Routines

[VIEW_NUMBER=integer] [VIEW_TITLE=string] [VIEW_ZOOM=value]

Surface Keywords:
- [RGB_TABLE=array of 256 by 3 or 3 by 256 elements]
- [TEXTURE_ALPHA=2-D array]
- [TEXTURE_BLUE=2-D array]
- [TEXTURE_GREEN=2-D array]
- [TEXTURE_IMAGE=array]
- [TEXTURE_RED=2-D array]
- [/X | Y | Z]LOG

Surface Object Keywords:
- [BOTTOM=index or RGB vector]
- [CLIP_PLANE=array]
- [COLOR=RGB vector]
- [DEPTH_OFFSET=value]
- [/DOUBLE]
- [/EXTENDED_LEGO]
- [HIDDEN_LINES]=/HIDE
- [INSERT_COLORBAR=value]
- [INSERT_LEGEND=value]
- [LINESTYLE=value]
- [MAX_VALUE=value]
- [MIN_VALUE=value]
- [SHADING=(0 | 1)]
- [SHOW_SKIRT]=/SKIRT=
- [SKIRT=Z value]
- [STYLE=(0 | 1 | 2 | 3 | 4 | 5 | 6)]
- [TEXTURE_HIGHERS=(0 | 1)]
- [TEXTURE_INTERP]=/INTERP=
- [TRANS_PARENCY=value]
- [USE_TRIANGLES]
- [VERT_COLOR=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]SUBTICKLENS=integer
- [X | Y | Z]TEXT_COLOR=RGB vector
- [X | Y | Z]TICK_FONT_INDEX=(0 | 1 | 2 | 3 | 4)
- [X | Y | Z]TICK_FONT_SIZE=integer
- [X | Y | Z]TICKFORMAT=string or string array
- [X | Y | Z]TICKINTERVAL=value
- [X | Y | Z]TICKLABEL=(0 | 1 | 2 | 3 | 4 | 5 | 6)]
- [X | Y | Z]TICKLENS=value
- [X | Y | Z]TICKNAME=string
- [X | Y | Z]TICKVALUES=vector
- [X | Y | Z]TITLE=string

ITCURRENT - Set 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 iTool.
Result = ITGETCURRENT([, iToolID=variable])

ITREGISTER - Registers tool object classes with the iTools system.
ITREGISTER, Name, ItemName [, /ANNOTATION]
[, TYPES=string] [, /UI_PANEL] [, /UI_SERVICE]
[, /USER_INTERFACE] [, /VISUALIZATION]

ITRESET - Resets the iTools session.
ITRESET[, /NO_PROMPT]

ITDELETE - Deletes a tool in the iTools system.
ITDELETE, [iToolID]

ITGETCURRENT - Gets the identifier of the current iTool.
Result = ITGETCURRENT([, iToolID=variable])

ITREGISTER, Name, ItemName [, /ANNOTATION]
[, TYPES=string] [, /UI_PANEL] [, /UI_SERVICE]
[, /USER_INTERFACE] [, /VISUALIZATION]

ITRESET - Resets the iTools session.
ITRESET[, /NO_PROMPT]

ITDELETE - Deletes a tool in the iTools system.
ITDELETE, [iToolID]

ITGETCURRENT - Gets the identifier of the current iTool.
Result = ITGETCURRENT([, iToolID=variable])

ITREGISTER, Name, ItemName [, /ANNOTATION]
[, TYPES=string] [, /UI_PANEL] [, /UI_SERVICE]
[, /USER_INTERFACE] [, /VISUALIZATION]

ITRESET - Resets the iTools session.
ITRESET[, /NO_PROMPT]
IVOLUME - Creates an iTool and associated user interface (UI) configured to display and manipulate volume data.

IVOLUME(, Vol0[, Vol1[, Vol2[, Vol3]])

iTool Common Keywords:
- [ANISTROPICSCALE_2D=value]
- [ANISTROPICSCALE_3D=value]
- [BACKGROUND_COLOR=value]
- [DIMENSIONS=[x, y, z]]
- [DISABLESplashScreen][/FITTOVIEW]
- [IDENTIFIER=variable][, LOCATION=[x, y, z]]
- [MACRONAMES=string or string array]
- [NAME=string][, /NO_SAVEPROMPT]
- [OVERPLOT=iToolID][, RENDERER=0|1]
- [SCALE_ANISTROPICSCALE=value]
- [STYLE_NAME=string][, TITLE=string]
- [VIEW_GRID=[columns, rows]][, /VIEW_NEXT]
- [VIEW_NUMBER=integer][, VIEW_TITLE=string]
- [VIEW_ZOOM=value]

Volume Keywords:
- [AUTO_RENDER]
- [RENDER EXTENTS=[0|1|2]]
- [RENDER QUALITY=[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]
- [BOUNDS=[xmin, ymin, zmin, xmax, ymax, zmax]]
- [CLIPPLANES=vector]
- [COMPOSITEFUNCTION=[0|1|2|3]]
- [DEPTHDUE=[bright, zdim]][, /4D]
- [HINTS=[0|1|2|3]][, INSERTCOLORBAR=value]
- [INTERPOLATE][, /LIGHTINGMODEL]
- [OPACITYTABLE=byte array of 256 elements]
- [OPACITYTABLE=byte array of 256 elements]
- [RENDER_STEP=[x, y, z]]
- [RGB_TABLE=byte array of 256 by 3 or 256 elements][, RGB_TABLE=byte array of 256 by 3 or 256 elements]
- [TWO_SIDED]
- [ZBUFFER]
- [ZERO_opacity_SKIP=[0|1]]

IVOLUME - continued

Axis Object Keywords:
- [GRIDSTYLE=[0|1|2|3|4][5][6]][, [X Y Z]MACH=[integer]
- [X Y Z]MINOR=[integer]
- [X Y Z]RANGE=[min, max]

IVOLUME

J

JOURNAL - Logs IDL commands to a file.IDL.

JOURNAL([], AX)

JULDAY - Returns Julian Day Number for given month, day, and year.

Result = JULDAY(Month,Day,Year,Hour,Minute,Second)

K

KEYWORD_SET - Returns True if Expression is defined and non-zero or an array.

Result = KEYWORD_SET(Expression)

KRIG2D - Interpolates set of points using kriging.

Result = KRIG2D(X[, Y][, EXPONENTIAL=vector][, SPHERICAL=vector][, /REGULAR][, XGRID=[xstart, xspacing]][, XVALUES=vector][, YGRID=[ystart, yspacing]][, YVALUES=vector][, GS=xspacing, yspacing][, BOUNDS=[xmin, ymin, xmax, ymax][, NX=value][, NY=value])

KURTOSIS - Computes statistical kurtosis of n-element vector.

Result = KURTOSIS(X[, /DOUBLE][, /NAN])

KW_TEST - Performs Kruskal-Wallis H-test.

Result = KW_TEST(X[, DF=variable][, /MISSING=nonzero_value])

L

L64INDGEN - Returns a 64-bit integer array with each element set to its subscript.

Result = L64INDGEN(D[1], ..., D[d][, Thread pool keywords])


Result = LABEL_DATE(, [DATE_FORMAT=string/string array][, AM_PM=2-element vector of strings][, DAYS_OF_WEEK=7-element vector of strings])
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.

LA_ELMHES - Reduces a real nonsymmetric or complex non-Hermitian array to upper Hessenberg form H.

LA_GM_LINEAR_MODEL - Used to solve a general Gauss-Markov linear model problem.
Result = LA_GM_LINEAR_MODEL( A, B, D, Y [, /DOUBLE] )

LA_HQR - Uses the multishift QR algorithm to compute all eigenvalues of an n-by-n upper Hessenberg array.
Result = LA_HQR(H [, Q] [, /DOUBLE] [, NORM_BALANCE=variable] [, PERMUTE_RESULT=variable] [, STATUS=variable])

LA_INVERT - Uses LU decomposition to compute the inverse of a square array.
Result = LA_INVERT(A [, /DOUBLE] [, STATUS=variable])

LA_LEAST_SQUARE_EQUALITY - Used to solve the linear least-squares problem.

LA_LEAST_SQUARES - Used to solve the linear least-squares problem.

LA_LINEAR_EQUATION - Uses LU decomposition to solve a system of linear equations.
Result = LA_LINEAR_EQUATION( Array, B [, BACKWARD_ERROR=variable] [, /DOUBLE] [, FORWARD_ERROR=variable] [, STATUS=variable])
LAFIT - Fits by minimizing the Chi-square error statistic.

\[ \text{Result} = \text{LAFIT}(X, Y, \text{CHISQ} \Rightarrow \text{variable}) \]
\[ [\text{MEASURE_ERRORS} \Rightarrow \text{vector}], \text{PROB} \Rightarrow \text{variable}] \]
\[ [\text{SIGMA} \Rightarrow \text{variable}], \text{YFIT} \Rightarrow \text{variable}] \]

LINKIMAGE - Merges routines written in other languages with IDL at run-time.

\[ \text{LINKIMAGE}, \text{Name}, \text{Image}, [\text{Type}, \text{Entry}] \]
\[ [\text{/DEVICE}], [\text{/FUNCTION}], [\text{/KEYWORDS}] \]
\[ [\text{MAX_ARGS} \Rightarrow \text{value}], [\text{MIN_ARGS} \Rightarrow \text{value}] \]

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

\[ \text{Result} = \text{LL_ARC_DISTANCE}(\text{Lon}_i, \text{Lon}, \text{Dist}, \text{Az}, [\text{/DEGREES}]) \]

LMFIT - Does a non-linear least squares fit.

\[ \text{Result} = \text{LMFIT}(X, Y, A, [\text{ALPHA} \Rightarrow \text{variable}]) \]
\[ [\text{CHISQ} \Rightarrow \text{variable}], [\text{CONVERGENCE} \Rightarrow \text{variable}] \]
\[ [\text{COVAR} \Rightarrow \text{variable}], [\text{FITA} \Rightarrow \text{vector}] \]
\[ [\text{FUNCTION_NAME} \Rightarrow \text{string}], [\text{ITER} \Rightarrow \text{variable}] \]
\[ [\text{ITMAX} \Rightarrow \text{variable}], [\text{ITMIN} \Rightarrow \text{variable}] \]
\[ [\text{MEASURE_ERRORS} \Rightarrow \text{vector}], [\text{SIGMA} \Rightarrow \text{variable}] \]
\[ [\text{TOL} \Rightarrow \text{variable}] \]

LNP_TEST - Computes the Lomb Normalized Periodogram.

\[ \text{Result} = \text{LNP_TEST}(X, Y, [\text{/DOUBLE}]) \]
\[ [\text{HIFAC} \Rightarrow \text{scale_factor}], [\text{JMA} \Rightarrow \text{variable}] \]
\[ [\text{OFAC} \Rightarrow \text{variable}], [\text{WK1} \Rightarrow \text{variable}], [\text{WK2} \Rightarrow \text{variable}] \]

LOADCT - Loads one of the predefined IDL color tables.

\[ \text{LOADCT}[, \text{Table} = \text{value}], [\text{FILE} = \text{string}] \]
\[ [\text{GET_NAMES} \Rightarrow \text{variable}], [\text{NCOLORS} \Rightarrow \text{value}] \]
\[ [\text{RGB_TABLE} \Rightarrow \text{value}], [\text{SILENT}] \]

LOCALSET - Returns the current locale of the operating platform.

\[ \text{Result} = \text{LOCALSET}() \]

LOGICAL_AND - Performs a logical AND operation on its arguments.

\[ \text{Result} = \text{LOGICAL_AND}(\text{Arg1}, \text{Arg2}) \]

LOGICAL_OR - Performs a logical OR operation on its arguments.

\[ \text{Result} = \text{LOGICAL_OR}(\text{Arg1}, \text{Arg2}) \]

LOGICAL_TRUE - Determines whether its arguments are non-zero (or non-NULL).

\[ \text{Result} = \text{LOGICAL_TRUE}(\text{Arg}) \]
LON64ARR - Returns a 64-bit integer vector or array.
Result = LON64ARR( D1[,...],/NOZERO)

LONARR - Returns a longword integer vector or array.
Result = LONARR( D1[,...],/NOZERO)

LONG - Converts argument to longword integer type.
Result = LONG( Expression [, Offset[, D1[,...]]]
[, Thread pool keywords] )

LONG64 - Converts argument to 64-bit integer type.
Result = LONG64( Expression [, Offset[, D1[,...]]]
[, 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[, RTOL=value]]
[, /QUIET_])

LU_COMPLEX - Solves complex linear system using LU decomposition.
Result = LU_COMPLEX( A, B [, /DOUBLE]
[, /INVERSE] [, /SPARSE] )

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])

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 [, /RADIANS | /RHUMB] )

MAP_CONTINENTS - Draws continental boundaries, filled continents, political boundaries, coastlines, and/or rivers, over an existing map projection established by MAP_SET.
MAP_CONTINENTS [, /COASTS] [, COLOR=index]
[, /CONTINENTS] [, /COUNTRIES]
[, /FILL_CONTINENTS=1 | 2]
[, /ORIENTATION=value] [, /HIRES] [, LIMIT=vector]
[, MAP_STRUCTURE=structure] [, MLINESTYLE=0 | 1 | 2 | 3 | 4 | 5]
[, MLINEWIDTH=vector] [, /RIVERS]
[, SPACING=centimeters [, /USA]

Graphics Keywords: [, /TH3D], [ZVALUE=value [0 to 1]]

MAP_GRID - Draws parallels and meridians over a map projection.
MAP_GRID [, /BOX_AXES] [, /CLIP_TEXT=0]
[, LA TALIGN=value[0.0 to 1.0]]
[, LONALIGN=value[0.0 to 1.0]]
[, LATLAB=longitudes] [, LONLAB=longitudes]
[, ORIENTATION=clockwise_degrees_from_horizon]
[, CHARSIZE=value] [, COLOR=index]
[, /FILL_HORIZON] [, GLINESTYLE=0 | 1 | 2 | 3 | 4 | 5]
[, GLINETHICK=vector] [, /HORIZON]
[, INCREMENT=value]
[, LABEL=n[label_every_nth_gridline]]
[, LATDEL=degrees] [, LATLINES=vector]
[, LONDEL=degrees] [, LONLINES=vector]
[, LONNAMES=vector] [, LONS=vector]
[, MAP_STRUCTURE=structure] [, /NO_GRID]
MAP_IMAGE - Returns an image warped to fit the current map projection. (Use when map data is larger than the display).

Result = MAP_IMAGE(Image [, Startx, Starty [, Xsize, Ysize]] [, LATMIN=degrees[-90 to 90]] [, LATMAX=degrees[-90 to 90]] [, LONMIN=degrees[-180 to 180]] [, LONMAX=degrees[-180 to 180]] [, /BILINEAR] [, COMPRESS=value] [, SCALE=value] [, IMAGE_STRUCTURE=structure] [, MASK=variable] [, MAX_VALUE=value] [, MIN_VALUE=value] [, MISSING=value] )

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.

Result = MAP_PROJ_INIT(Projection [, DATUM=value] [, /GCTP] [, LIMIT=variable] [, /RADIANS] [, /RELAXED])

Keywords—Projection Parameters:

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

Result = MAP_PROJ_IMAGE(Image [, Startx, Starty [, Xsize, Ysize]] [, LAT_MIN=degrees[-90 to 90]] [, LAT_MAX=degrees[-90 to 90]] [, LON_MIN=degrees[-180 to 180]] [, LON_MAX=degrees[-180 to 180]] [, /BILINEAR] [, COMPRESS=value] [, SCALE=value] [, IMAGE_STRUCTURE=structure] [, MASK=value] [, MAX_VALUE=value] [, MIN_VALUE=value] [, MISSING=value] )

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.

Result = MAP_PROJ_INIT(Projection [, DATUM=value] [, /GCTP] [, LIMIT=vector] [, /RADIANS] [, /RELAXED])

Keywords—Projection Parameters:
Alphabetical List of IDL Routines

Keywords—Projection Parameters:
- CENTRAL_AZIMUTH=degrees_east_of_north
- ELLIPSOID=array
- /ISOTROPIC
- LIMIT=vector
- SAT=vector
- /SCALE=value
- STANDARD_PARALLELS=array

Graphics Keywords:
- POSITION=[X0, Y0, X1, Y1]
- /T3D
- ZVALUE=value[0 to 1]

MATRIX_MULTIPLY - Calculates the IDL matrix-multiply operator (#) of two (possibly transposed) arrays.
Result = MATRIX_MULTIPLY(A, B [, /ATRANSPOSE] [, /BTRANSPOSE] [, Thread pool keywords])

MATRIX_POWER - Computes the product of a matrix with itself.
Result = MATRIX_POWER(A, N [, /DOUBLE] [, STATUS=value])

MAX - Returns the value of the largest element of Array.

MEDIAN - Computes the median value of Array or applies a median filter.
Result = MEDIAN(Array [, Width] [, /DOUBLE] [, /MEDIAN] [, /NAN])

MEAN - Computes the mean of a numeric vector.
Result = MEAN(X [, /DOUBLE] [, /NAN])

MEANABSDEV - Computes the mean absolute deviation of a vector.
Result = MEANABSDEV(X [, /DOUBLE] [, /MEDIAN] [, /NAN])

MEMORY - Returns a vector containing information on the amount of dynamic memory currently in use by the IDL session.

MESH_CLIP - Clips a polygonal mesh to an arbitrary plane in space and returns a polygonal mesh of the remaining portion.
Result = MESH_CLIP(Plane, Verts_in, Conn_in, Verts_out, Conn_out [, AUXDATA_IN=Array, AUXDATA_OUT=variable[, PUT_VERTICES=variable]])

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_VERTICES=percent] [, PERCENT_POLYGONS=percent] [, PROGRESS_CALLBACK=string] [, PROGRESS_METHOD=string] [, PROGRESS_OBJECT=variable] [, /PROGRESS_PERCENT=percent[0 to 100]] [, /PROGRESS_USERDATA=value])

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, Verts1, Conn1 [, /COMBINE_VERTICES] [, TOLERANCE=value])

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=Array] [, FIXED_EDGE_VERTICES] [, LAMBDAD=value])

MESH_SURFACEAREA - Computes various mesh properties and enables IDL to determine if a mesh encloses space (is a solid).
Result = MESH_SURFACEAREA(Verts, Conn, AUXDATA_in=Array, MOMENT=value)

MESH_VALIDATE - Checks for NaN values in vertices, removes unused vertices, and combines close vertices.
Result = MESH_VALIDATE(Verts, Conn [, /REMOVE_NAN] [, /PACK_VERTICES] [, /COMBINE_VERTICES] [, TOLERANCE=value])

MESH_VOLUME - Computes the volume that the mesh encloses.
Result = MESH_VOLUME(Verts, Conn [, DIMENSION=value] [, MAX=variable] [, /NAN] [, SUBSTITUTE_MAX] [, /Thread pool keywords])

MESSAGE - Issues error and informational messages.

MIN - Returns the value of the smallest element of an array.

MIN_CURVE_SURF - Interpolates over either a plane or a sphere with a minimum curvature surface or a thin-plate-spline surface.
MK HTML_HELP - Converts text documentation headers to HTML files.
MK HTML_HELP, Sources, Filename [, /NAN] [, /VERBOSE]

MODIFYCT - Saves modified color tables in the IDL color table file.
MODIFYCT, Itab, Name, R, G, B [, FILE=filename]

MOMENT - Computes mean, variance, skewness, and kurtosis.

MORPH_CLOSE - Applies closing operator to a binary or grayscale image.

MORPH_DISTANCE - Estimates N-dimensional distance maps, which contain for each foreground pixel the distance to the nearest background pixel, using a given norm.
Result = MORPH_DISTANCE(Data [, /BACKGROUND] [, NEIGHBOR_SAMPLING={1 | 2 | 3}] [, /NO_COPY] )

MORPH_GRADIENT - Applies the morphological gradient operator to a grayscale image.

MORPH_HITORMISS - Applies the hit-or-miss operator to a binary image.
Result = MORPH_HITORMISS( Image, HitStructure, MissStructure )

MORPH_OPEN - Applies the opening operator to a binary or grayscale image.

MORPH_THIN - Performs a thinning operation on binary images.
Result = MORPH_THIN( Image, HitStructure, MissStructure )

MORPH_TOPHAT - Applies top-hat operator to a grayscale image.

MPEG_CLOSE - Closes an MPEG sequence.
MPEG_CLOSE, mpegID

MPEG_OPEN - Opens an MPEG sequence.
mpegID = MPEG_OPEN( Dimensions[], BITRATE=value] [, FILENAME=string] [, IFRAME_GAP=integer value] [, MOTION_VEC_LENGTH={1 | 2 | 3}] [, QUALITY=value(0 to 100)] )

MPEG_PUT - Inserts an image array into an MPEG sequence

MPEG_SAVE - Encodes and saves an open MPEG sequence.
MPEG_SAVE, mpegID [, FILENAME=string]
Alphabetical List of IDL Routines

O

OBJ_CLASS - Determines the class name of an object.
Result = OBJ_CLASS( [Arg] [, COUNT=variable ] [ , /SUPERCALSS=must specify Arg] )

OBJ_DESTROY - Destroys an object reference.
OBJ_DESTROY( ObjRef[, Arg1, …, Argn] )

OBJ_HASMETHOD - Determines whether an object class implements a specified method or methods.
Result = OBJ_HASMETHOD( ObjRef, Method )

OBJ_ISA - Determines inheritance relationship of an object.
Result = OBJ_ISA( ObjRef, Instance[, ClassName] )

OBJ_NEW - Creates an object reference.
Result = OBJ_NEW( [ObjectClassName[, Arg1, …, Argn ]])

OBJ_VALIDATE - Verifies validity of object references.

OBJJARR - Creates an array of object references.
Result = OBJJARR( D1[, …, Dn] [, /NOZERO] )

ON_ERROR - Designates the error recovery method.
ON_ERROR, N

ON_IOERROR - Declares I/O error exception handler.
ON_IOERROR, Label
   - Label: Statement to perform upon I/O error

ONLINE_HELP - Invokes online help viewer from programs.

Windows-Only Keywords: [, /CONTEXT]

OPEN - Opens files for reading, updating, or writing.
OPEN, Unit, File
OPENW, Unit, File
OPENU, Unit, File

Keywords (all platforms): [, /APPEND ] [, /COMPRESS ] [, BUFsize=0 | 1 | value >512 ] [, /DELETE ]
 [, ERROR=variable ] [, /F77_UNFORMATTED ]
 [, GET_LUN ] [, /MORE ] [, /NOEXPAND_PATH ]
 [, /STDIO ] [, /SWAP_ENDIAN ]
 [, SWAP_IF_BIG_ENDIAN ]
 [, SWAP_IF_LITTLE_ENDIAN ] [, /VAX_FLOAT ]
 [, WIDTH=value ] [, /XDR]

UNIX-Only Keywords: [, /RAW10]

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=( X0, Y0, X1, Y1 )]
 [, COLOR=value ] [, LINESSTYLE=[0 | 1 | 2 | 3 | 4 | 5 ]]
 [, /NOCLIP ] [, PSYM=integer {0 to 10] ]
 [, SYMSIZE=value ] [, /T3D ] [, ZVALUE=value {0 to 1} ]

OPLOTRERR - Draws error bars over a previously-drawn plot.
OPLOTRERR, [ X] Y, Err [, Psys ]

P

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 | 1 ]
 [, SEED.NORMAL=value ] [, 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 ] [, /VARIABLES=variable ]
 [, /STANDARDIZE ] [, VARIANCES=variable ]

PLOT - Plots vector arguments as X versus Y graphs.
PLOT, [ X], Y [, /ISOTROPIC ] [, MAX_VALUE=value ]
 [, MIN_VALUE=value ] [, NSUM=value ] [, /POLAR ]
 [, THICK=value ] [, /XLOG ] [, /YLOG ] [, /YNOZERO]

Graphics Keywords: [, BACKGROUND=color_index ]
 [, CHAR_SIZE=value ] [, CHAR_THICK=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 ] CHAR_SIZE=value
 [, /X | Y ] GRID_STYLE=integer{0 to 5]}
 [, /X | Y ] MARGIN=[left, right] ]
 [, /X | Y ] MINOR=integer ]
 [, /X | Y ] RANGE=[min, max] ]
 [, /X | Y ] STYLES=variable ] [, /X | Y ] THICK=value ]
 [, /X | Y ] TICK_GET=variable ]
 [, /X | Y ] TICK_FORMAT=string ]
 [, /X | Y ] TICK_INTERVAL=value ]
 [, /X | Y ] TICKLAYOUT=scalar ]
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_LINESTYLE=vector]
[ , /CLOSED] [ , /IRREGULAR]
[ , LEVELS=vector]
[ , NLEVELS=integer]
[ , MIN_VALUE=value]
[ , /OVERPLOT]
[ , /PATH_DATA_COORDS]
[ , TRIANGULATION=variable] [ , XLOG] [ , YLOG]
[ , ZAXIS]
[ , SHOW_TRIANGULATION=color_index]

POLAR_SURFACE - Interpolates a surface from polar coordinates to rectangular coordinates.

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 = Dim_p, Dim_q] [ , 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] [, , /SIGN])

POLY_FIT - Performs a least-square polynomial fit.
Result = POLY_FIT(X, Y, Degree [, CHISQ=value] [ , COVAR=value] [ , , /DOUBLE]
[ , MEASURE_ERRORS=vector] [ , SIGMA=value]
[ , STATUS=value] [ , YBAND=value]
[ , YERROR=value] [ , YFIT=value])

POLYFILL - Fills the interior of a polygon.

POLYFILL, X [, Y, Z] [, IMAGE_COORD=coordinate]
[ , /IMAGE_INTERP] [ , /LINE_FILL]
[ , PATTERN=vector] [ , SPACING=centimeters]
[ , TRANSPARENT=value]

Graphics Keywords: 
[ , CLIP=vector]
[ , COLOR=value]
[ , DATA] [ , /DEVICE] [ , /NORMAL]
[ , LINES_STYLE=vector]
[ , PSYM=value]
[ , THICK=value]

POLYFILLV - Returns subscripts of pixels inside a polygon.
Result = POLYFILLV( X, Y, Z, Ru, Sp, Run, L, Length )

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

Result = POLYSHADE( Vertices, Polygons) or
Result = POLYSHADE(X, Y, Z, Polygons)

Keywords: 
[ , /DATA] [ , /NORMAL]
[ , POLY_SHADES=vector] [ , SHADES=vector] [ , /T3D]
[ , TOP=value] [ , XSIZE=columns] [ , YSIZE=rows]

POLYWARP - Performs polynomial spatial warping.

POLYWARP, Xi, Yi, Xo, Yo, Degree, Kx, Ky [, , /DOUBLE]
[ , STATUS=value]

POPD - Removes the top directory on the working directory stack maintained by PUSH/POPD.

POWELL - Minimizes a function using the Powell method.

POWELL, P, Xi, Tiol, Fmin, Func [, , /DOUBLE]
[ , ITER=value] [ , ITMAX=value]
Alphabetical List of IDL Routines

PREF_COMMIT - Commits IDL preferences already in the pending state.
Result = PREF_COMMIT(PreferenceName [, /RESET] [, /RESIGNAL])

PREF_GET - Returns information about IDL preferences.
Result = PREF_GET(PreferenceName [, /NAMES_ALL | /NUM_PENDING | /STRUCTURE])

PREF_MIGRATE - Imports IDL user preferences from other versions of IDL for use by the currently running version.
Result = PREF_MIGRATE([], /MACRO | /PREFERENCE [, /STARTUP])

PRINT - Prints contents of the directory stack maintained by PRINTD/POPD.
PRINTD

PRO - Defines a procedure.
PROCEDURE Name, argument1, ..., argumentn

PRODUCT - Returns the product of elements within an array.

PROFILE - Extracts a profile from an image.

PROFILER - Accesses the IDL Code Profiler used to analyze performance of applications.
PROFILER [ Module | /CLEAR | DATA=variable] [ OUTPUT=variable | /REPORT | /RESET] [/SYSTEM]

PROFILES - Interactively examines image profiles.
PROFILES, Image [, /ORDER] [, SX=value] [, SY=value] [, WSIZE=value]

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

PS_SHOW_FONTS - Displays all the PostScript fonts that IDL knows about.
PS_SHOW_FONTS [, /NOLATIN]

PSAFM - Converts Adobe Font Metrics file to IDL format.
PSAFM, Input_Filename, Output_Filename

PSEUDO - Creates pseudo-color table based on Lightness, Hue, and Brightness system.
PSEUDO, Lilo, Lithi, Satio, Sushi, Hue, Loops [, Colr]

PTR_FREE - Destroys a pointer.
PTR_FREE, P1, ..., Pn

PTR_NEW - Creates a pointer.
Result = PTR_NEW([ InitExpr ] [, /ALLOCATE_HEAP] [, /NO_COPY] )

PTR_VALID - Verifies the validity of pointers.

PTRARR - Creates an array of pointers.
Result = PTRARR(D1 [, ... , Dn] [, /ALLOCATE_HEAP] [, /NOZERO] )

PUSHD - Pushes a directory to top of directory stack maintained by PRINTD/POPD.
PUSHD, Dir

Q

QGRID3 - Interpolates the dependent variable values to points in a regularly sampled volume.
Result = QGRID3(X, Y, Z, Tetrahedra) or
Result = QGRID3(X, Y, Z, F, Tetrahedra)
Keywords: [, DELTA=Array] [, DIMENSION=Array] [, MISSING=variable] [, START=Array]
QHULL - Constructs convex hulls, Delaunay triangulations, and Voronoi diagrams.

$query_{ASCII}$ - Tests a file for compatibility with READ_ASCII.

QHULL, V, Tr or QHULL, V0, V1, ..., Vn, Tr
[. BOUNDS=variable] [. CONNECTIVITY=variable]
[. DELAUNAY] [. SPHERE=variable]
[. VDIAGRAM=variable] [. VNORMALS=variable]
[. VVERTICES=variable]

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] [. K=value] )

QUERY_ASCII - Obtains information about a text file.

Result = QUERY_ASCII( Filename [, Info] )

QUERY_MRSID - Obtains information about a MRSID image.

Result = QUERY_MRSID( Filename [, Info] [. LEVEL=bv] )

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_SRIF - Obtains information about an SRF image file.

Result = QUERY_SRIF( 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_CORRELATE - Computes rank correlation.


RANKS - Computes magnitude-based ranks.

Result = RANKS(X)

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

READ, [Prompt,] Var1, ..., Varn
READF, [Prompt,] Unit, Var1, ..., Varn

Keywords: [. AM_PM=string, string] [. DAYS_OF_WEEK=string_array{7 names}] [. FORMAT=value] [. MONTHS=string_array{12 names}] [. PROMPT=string]
Alphabetical List of IDL Routines

READ_ASCII - Reads data from an ASCII file.

Result = READ_ASCII(Filename)
[,...]
[COMMENT_SYMBOL=string] [,...]
[DATA_START=lines_to_skip] [,...]
[DELETEIMITER=string] [,...]
[HEADER=variable] [,...]
[MISSING_VALUE=value] [,...]
[NUM_RECORDS=value] [,...]
[RECORD_START=index] [,...]
[TEMPLATE=variable] [,...] [/VERBOSE]

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

Result = READ_BINARY(Filename [FileUnit] [,...]
[TEMPLATE=template] [,...]
[DATA_TYPE=typecodes] [,...]
[DATA_DIMS=array] [,...]
[ENDIAN=string] [,...])

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

Result = READ_BMP(Filename [, R, G, B] [, Ihdr] [,...]
[, /RGB] [,...])

READ_DICOM - Reads an image from a DICOM file.

Result = READ_DICOM(Filename [, Red, Green, Blue] [,...]
[IMAGE_INDEX=index] [, /DICOMEX])

READ_GIF - Reads a GIF image.

READ_GIF(Filename, Image [, R, G, B] [,...]
[BACKGROUND_COLOR=variable] [,...]
[CLOSE] [,...]
[DELAY_TIME=variable] [,...]
[DISPOSAL_METHOD=variable] [,...] [/MULTIPLE]
[REPEAT_COUNT=variable] [,...]
[TRANSPARENT=variable] [,...]
[USER_INPUT=variable]

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

Result = READ_IMAGE(Filename [, Red, Green, Blue] [,...]
[IMAGE_INDEX=index])

READ_INTERFILE - Reads Interfile (v3.3) file.

READ_INTERFILE, File, Data

READ_JPEG - Reads JPEG file.

READ_JPEG(Filename [, UNIT=string] , Image [,...]
[Colorable] [,...]
[BUFFER=variable] [,...]
[COLORS=value] [,...]
[DITHER={0 | 1 | 2}] [,...]
[GRAYSCALE] [,...] [/ORDER] [,...]
[TRUE={1 | 2 | 3}] [,...]
[TWO_PASS_QUANTIZE] [,...])


Result = READ_JPEG2000(Filename [, Red, Green, Blue] [,...]
[DISCARD_LEVELS=value] [,...]
[MAX_LAYERS=value] [,...] [/ORDER]
[REGION={StartX, StartY, Width, Height}])

READ_MRSID - Reads MRSID file.

Result = READ_MRSID(Filename [, LEVEL=lv] [,...]
[SUB_RECT=rect])

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

READ_PICT, Filename, Image [, R, G, B]


Result = READ_PNG(Filename [, R, G, B] [,...]
[ORDER] [,...] [/VERBOSE] [,...]
[TRANSPARENT=variable])
or

RESULT_PNG, Filename, Image [, R, G, B] [,...]
[ORDER] [,...] [/VERBOSE] [,...]
[TRANSPARENT=variable]

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

RESULT_PPM, Filename, Image [, MAXVAL=variable]

READ_SRF - Reads Sun Raster Format file.

READ_SRF, Filename, Image [, R, G, B]

READ_SYLK - Reads Symbolic Link format spreadsheet file.

RESULT_SYLK(Filename [, ARRAY] [,...]
[COLMAJOR] [,...]
[NROWS=column] [,...]
[NROWS=rows] [,...]
[STARTCOL=column] [,...]
[STARTROW=row] [,...]
[USEDOUBLES] [,...]
[USELONGS])

READ_TIFF - Reads TIFF format file.

RESULT_TIFF(Filename [, R, G, B] [,...]
[CHANNELS=scalar or vector] [,...]
[DOT_RANGE=variable] [,...]
[GEOTIFF=variable] [,...]
[IMAGE_INDEX=value] [,...]
[ICC_PROFILE=variable] [,...]
[INTERLEAVE={0 | 1 | 2}] [,...]
[ORIENTATION=variable] [,...]
[PHOTOSHOP=variable] [,...]
[PLANARCONFIG=variable] [,...]
[SUB_RECT={x, y, width, height}] [,...]
[UNSIGNED] [,...] [/VERBOSE]

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

RESULT_WAV(Filename [, Rate] [,...])

READ_WAVE - Reads Wavefront Advanced Visualizer file.

READ_WAVE, File, Variables, Names, Dimensions [,...]
[MESHNAMES=variable]

READ_X11_BITMAP - Reads X11 bitmap file.

READ_X11_BITMAP, File, Bitmap [, X, Y] [,...]
[EXPAND_TO_BYTES]

READ_XWD - Reads X Windows Dump file.

RESULT_XWD(Filename [, R, G, B])

READS - Reads formatted input from a string variable.

READS, Input, Var_, ..., Var_n [, AM_PM=string, string] [,...]
[DAYS_OF_WEEK=string_array{7 names}] [,...]
[FORMAT=value] [,...]
[MONTHS=string_array{12 names}]

READU - Reads unformatted binary data from a file.

READU, Unit, Var_, ..., Var_n [,...]
[TRANSFER_COUNT=variable]

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

Result = REAL_PART(Z)
<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REBIN</td>
<td>Resizes a vector or array by integer multiples.</td>
</tr>
<tr>
<td>RECALL_COMMANDS</td>
<td>Returns entries in IDL's command recall buffer.</td>
</tr>
<tr>
<td>RECON3</td>
<td>Reconstructs a 3D representation of an object from 2D images.</td>
</tr>
<tr>
<td>REDUCE_COLORS</td>
<td>Reduces the number of colors used in an image by eliminating unused pixel values.</td>
</tr>
<tr>
<td>REFORM</td>
<td>Changes array dimensions without changing the total number of elements.</td>
</tr>
<tr>
<td>REGION_GROW</td>
<td>Perform region growing.</td>
</tr>
<tr>
<td>REGISTER_CURSOR</td>
<td>Associates the given name with the given cursor information.</td>
</tr>
<tr>
<td>REGRESS</td>
<td>Computes fit using multiple linear regression.</td>
</tr>
<tr>
<td>REVERSE</td>
<td>Reverses the order of one dimension of an array.</td>
</tr>
<tr>
<td>ROTATE</td>
<td>Rotates an image by any amount.</td>
</tr>
<tr>
<td>ROT</td>
<td>Rotates/transposes an array in multiples of 90 degrees.</td>
</tr>
<tr>
<td>ROUND</td>
<td>Rounds the argument to its closest integer.</td>
</tr>
<tr>
<td>ROUTINE_INFO</td>
<td>Provides information about compiled procedures and functions.</td>
</tr>
<tr>
<td>RS_TEST</td>
<td>Performs the Wilcoxon Rank-Sum test.</td>
</tr>
<tr>
<td>SAVGOL</td>
<td>Returns coefficients of Savitzky-Golay smoothing filter.</td>
</tr>
</tbody>
</table>

**RECALL_COMMANDS**

```
Result = RECALL_COMMANDS( )
```

**RECON3**

```
```

**REDUCE_COLORS**

```
Result = REDUCE_COLORS( )
```

**REFORM**

```
Result = REFORM( Array, D_1 [, ... D_8] [, /OVERWRITE] )
```

**REGION_GROW**

```
Result = REGION_GROW( Array, ROIPixels [, /ALL_NEIGHBORS] [, /NAN] [,STDEV_MULTIPLIER=value] [,THRESHOLD=[min,max]] )
```

**REGISTER_CURSOR**

```
REGISTER_CURSOR, Name, Image[,MASK=value] [,HOTSPOT=value] [, /OVERWRITE]
```

**REGRESS**

```
```

**REPEAT...UNTIL**

```
REPEAT ... UNTIL - Repeats statement(s) until expression evaluates to true. Subject is always executed at least once.
```

**REPLICATE**

```
Result = REPLICATE( Value, D_1 [, ... D_8] [, /THREAD_POOL_KEYWORDS] )
```

**REPLICATE_INPLACE**

```
Result = REPLICATE_INPLACE( X, Value [,D_1] [,D_2,... D_8] [, /THREAD_POOL_KEYWORDS] )
```

**RESOLVE_ALL**

```
RESOLVE_ALL [,CLASS=string] [, /CONTINUE_ON_ERROR] [, /QUIET]
```

**RESOLVE_ROUTINE**

```
RESOLVE_ROUTINE, Name [, /COMPILE_FULL_FILE] [, /NEITHER] [, /IS_FUNCTION] [, /NO_RECOMPILE]
```

**RESTORE**

```
```

**REBIN**

```
Result = REBIN( Array, D_1 [, ... D_8] [, /SAMPLE] )
```

**RESOLVE_COMMANDS**

```
Result = RESOLVE_COMMANDS( )
```

**RESOLVE_ROUTINE**

```
RESOLVE_ROUTINE, Name [, /COMPILE_FULL_FILE] [, /NEITHER] [, /IS_FUNCTION] [, /NO_RECOMPILE]
```

**RESTORE**

```
```

**RECALL_COMMANDS**

```
Result = RECALL_COMMANDS( )
```

**RECON3**

```
```

**REDUCE_COLORS**

```
Result = REDUCE_COLORS( )
```

**REFORM**

```
Result = REFORM( Array, D_1 [, ... D_8] [, /OVERWRITE] )
```

**REGION_GROW**

```
Result = REGION_GROW( Array, ROIPixels [, /ALL_NEIGHBORS] [, /NAN] [,STDEV_MULTIPLIER=value] [,THRESHOLD=[min,max]] )
```

**REGISTER_CURSOR**

```
REGISTER_CURSOR, Name, Image[,MASK=value] [,HOTSPOT=value] [, /OVERWRITE]
```

**REGRESS**

```
```
SAVE - Saves variables, system variables, and IDL routines in a file for later use.
SAVE [, Var1, ... ,VARn] [, /ALL] [, /COMM, /VARIABLES] [, /COMPRESS]
[, DESCRIPTION=string] [, /EMBEDDED]
[, FILENAME=string] [, /Routines]
[, /SYSTEM_VARIABLES] [, /VERBOSE]

SCALE3 - Sets up axis ranges and viewing angles for 3D plots.
SCALE3 [, Xrange=vector] [, Yrange=vector] [, AX=degrees] [, AZ=degrees]

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

SCOPE_LEVEL - Returns the scope level of the currently running procedure or function.
Result = SCOPE_LEVEL()

SCOPE_TRACEBACK - Returns the current interpreter call stack (the sequence of routine calls to the present point).
Result = SCOPE_TRACEBACK( [ /STRUCTURE] [, /SYSTEM] )

SCOPE_VARFETCH - Returns variables outside the local scope of the currently running procedure or function.
Variable = SCOPE_VARFETCH(VarName [, COMMON=string] [, /ENTER] [, LEVEL=value]
[, /REF_EXTRA])

SCOPE_VARNAME - Returns the names of variables outside the local scope of the currently running procedure or function.
Result = SCOPE_VARNAMES(Var1, ... ,VARn] [, COMMON=string] [, COUNT=variable] [, LEVEL=value]
[, /REFER_EXTRA])

SEARCH2D - Finds “objects” or regions of similar data within a 2D array.
Result = SEARCH2D(Array, Xpos, Ypos, Min_Val, Max_Val [, /DECREASE, /INCREASE]
[, LPF_BAND=integer(≥31)] [, /DIAGONAL])

SEARCH3D - Finds “objects” or regions of similar data values within a volume.
Result = SEARCH3D(Array, Xpos, Ypos, Zpos, Min_Val, Max_Val [, /DECREASE, /INCREASE]
[, LPF_BAND=integer(≥31)] [, /DIAGONAL])

SEM_CREATE - Creates a reference to a semaphore in the current IDL process. Creates the semaphore itself if necessary.
Result = SEM_CREATE(strName [, /DESTROY_SEMAPHORE])

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

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

SEM_RELEASE - Releases the lock on the specified semaphore held by the current IDL process.
SEM_RELEASE, strName

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] [, VALUE=[darkest, brightest]]

SETENV - Adds or changes an environment variable.
SETENV, Environment_Expression

SFIT - Performs polynomial fit to a surface.
Result = SFIT(Data, Degree [, /IRREGULAR, KX=variable, /MAX_DEGREE])

SHADE_SURF - Creates a shaded-surface representation of gridded data.
SHADE_SURF, Z [, X, Y] [, AX=degrees] [, AZ=degrees]
[, IMAGE=variable] [, MAX_VALUE=value]
[, MIN_VALUE=value] [, PIXELS=pixels] [, /SAVE]
[, SHADES=array] [, /XLOG] [, /YLOG]

Graphics Keywords: [ , CHARHEIGHT=value]
[ , CHARWIDTH=value]
[, POSITION=[X0, Y0, X1, Y1]] [, SUBTITLE=string]
[, /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]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=variable]
[, [X | Y | Z]TICK_GET=variable]
[, [X | Y | Z]TITLE=string]
[, ZVALUE=value(0 to 1)]

SHADE_SURF_IRR - Creates a shaded-surface representation of an irregularly gridded dataset.
[, /T3D]

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]
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHMDEBUG</td>
<td>Print debugging information when a variable loses reference to an underlying shared memory segment.</td>
</tr>
<tr>
<td>SHMMAP</td>
<td>Maps anonymous shared memory, or local disk files, into the memory address space of the currently executing IDL process.</td>
</tr>
<tr>
<td>SHMVAR</td>
<td>Creates an IDL array variable that uses the memory from a current mapped memory segment created by the SHMMAP procedure.</td>
</tr>
<tr>
<td>SHOW3</td>
<td>Displays array as image, surface plot, and contour plot simultaneously.</td>
</tr>
<tr>
<td>SHOWFONT</td>
<td>Displays a TrueType or vector font.</td>
</tr>
<tr>
<td>SIMPLEX</td>
<td>Use the simplex method to solve linear programming problems.</td>
</tr>
<tr>
<td>SIN</td>
<td>Returns the trigonometric sine of X.</td>
</tr>
<tr>
<td>SINDGEN</td>
<td>Returns a string array with each element set to its subscript.</td>
</tr>
<tr>
<td>SIZE</td>
<td>Returns array size and type information.</td>
</tr>
<tr>
<td>SKEWNESS</td>
<td>Computes statistical skewness of an n-element vector.</td>
</tr>
<tr>
<td>SLICER3</td>
<td>Interactive volume visualization tool.</td>
</tr>
<tr>
<td>SLIDE_IMAGE</td>
<td>Creates a scrolling graphics window for examining large images.</td>
</tr>
<tr>
<td>SMOOTH</td>
<td>Smooths with a boxcar average.</td>
</tr>
<tr>
<td>SOBEL</td>
<td>Returns an approximation of Sobel edge enhancement.</td>
</tr>
<tr>
<td>SOCKET</td>
<td>Opens client-side TCP/IP Internet socket as IDL file unit.</td>
</tr>
<tr>
<td>SORT</td>
<td>Returns indices of an array sorted in ascending order.</td>
</tr>
<tr>
<td>SPAWN</td>
<td>Spawns child process for access to operating system.</td>
</tr>
</tbody>
</table>

**IDL Quick Reference**
### Alphabetical List of IDL Routines

**UNIX-Only Keywords:** 
- /NOSHELL
- /NOTTYRESET
- /NULL_STDIN
- /SH
- /STDERR
  
- UNIT=variable [Command required, Result not allowed]

**Windows-Only Keywords:** 
- /HIDE
- /LOG_OUTPUT
- /NOSHELL
- /NOWAIT
- /NULL_STDIN
- /STDERR

#### 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]] [, BOUT=variable] [, GOUT=variable] [, GS=[lonspacing, latspacing]] [, NLON=value] [, NLAT=value])

#### 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] )

#### SPL_INTERP
- Performs cubic spline interpolation.
  - Result = SPL_INTERP( X, Y, Y2 [, /DOUBLE] )

#### SPLINE
- Performs cubic spline interpolation.
  - Result = SPLINE( X, Y, T [, Sigma] [, /DOUBLE] )

#### SPLINE_P
- Performs parametric cubic spline interpolation.
  - Result = SPLINE_P( X, Y, Xr, Yr [, /DOUBLE] )

#### SPRSAB
- Performs matrix multiplication on sparse matrices.
  - Result = SPRSAB( A, B [, /DOUBLE] )

#### SPRSAX
- Multiplies sparse matrix by a vector.
  - Result = SPRSAX( A, X [, /DOUBLE] )

#### SPRSIN
- Converts matrix to row-index sparse matrix.

#### SPRSTP
- Constructs the transpose of a sparse matrix.
  - Result = SPRSTP(A)

#### SQRT
- Returns the square root of X.
  - Result = SQRT( X [, /POOL] )

#### STANDARDIZE
- Converts standardized variables.
  - Result = STANDARDIZE( A [, , /DOUBLE] )

#### STDERR
- Computes the standard deviation of an n-element vector.
  - Result = STDERR( X [, , /DOUBLE] [, /NAN] )

#### STOP
- Stops the execution of a running program or batch file.
  - Result = STOP [[ Exp1, ..., Expn] ...]

#### 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] )

#### STRSTREAMLINE
- Generates the visualization graphics from a path.
  - STREAMLINE, Verts, Conn, Normals, Outverts, Outconn [, ANISOTROPY=vector] [, SIZE=vector] [, PROFILE=vector]

#### STRETCH
- Stretches color table for contrast enhancement.
  - Result = STRETCH( Low, High [, Gamma] )

#### STRING
- Converts its arguments to string type.
  - Result = STRING( Expression [, ... , Expression] [, AM_PM=string, string] [, DAYS_OF_WEEK=string_array[7 names]] [, FORMAT=value] [, MONTHS=string_array[12 names]] [, /PRINT] )

#### STRJOIN
- Collapses a string scalar or array into merged strings.
  - Result = STRJOIN( String [, Delimiter] [, /SINGLE] )

#### STRLEN
- Returns the length of a string.
  - Result = STRLEN( Expression )

#### STRLOWCASE
- Converts a string to lower case.
  - Result = STRLOWCASE( String )

#### STRMATCH
- Compares search string against input string expression.
  - Result = STRMATCH( String, SearchString [, /FOLD_CASE] )

#### STRMESSAGE
- Returns the text of an error number.

#### 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, Search String [, Pos] [, /REVERSE_OFFSET] [, /REVERSE_SEARCH] )

#### STRPRINTF
- Inserts the contents of one string into another.
  - Result = STRPRINTF( Destination, Source [, Position] )

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

#### STRSJOIN
- Collapses a string scalar or array into merged strings.
  - Result = STRSJOIN( String [, Delimiter] [, /SINGLE] )

#### STRSLEN
- Returns the length of a string.
  - Result = STRSLEN( Expression )

#### STRSLOWCASE
- Converts a string to lower case.
  - Result = STRSLOWCASE( String )

#### STRSTREAMLINE
- Generates the visualization graphics from a path.
  - STREAMLINE, Verts, Conn, Normals, Outverts, Outconn [, ANISOTROPY=vector] [, SIZE=vector] [, PROFILE=vector]

#### STREXTRACT
- Extracts a substring from a string.
  - Result = STREXTRACT( String [, Delimiter] [, /SINGLE] )

#### STREAMLINE
- Generates the visualization graphics from a path.
  - STREAMLINE, Verts, Conn, Normals, Outverts, Outconn [, ANISOTROPY=vector] [, SIZE=vector] [, PROFILE=vector]
SURFR - SVSOL - SVDFIT - SWAP_ENDIAN - STRUCT_ASSIGN - STRTRIM - IDL Quick Reference

66

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, Arg 1 [, Arg 2, ... , Arg n]

STRUPCASE - Converts a string to upper case.
Result = STRUPCASE(String)

SURFACE - Plots an array as a wireframe mesh surface.

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.

SVDFIT - Performs multivariate least squares fit using SVD method.

SVSOL - Solves set of linear equations using back-substitution.

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 scallars, arrays or structures.
SWAP_ENDIAN_INPLACE, Variable [, /SWAP_IF_BIG_ENDIAN] [, /SWAP_IF_LITTLE_ENDIAN]

SWAP_ENDIAN_INPLACE - Reverses the byte ordering of scallars, 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 d value.
String = SYSTIME([0 [, ElapsedSeconds]] [, /UTC] ) or
Seconds = SYSTIME(1 [, /SECONDS] ) or
Julian = SYSTIME(1 [, /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=exp(eye at (0,0,p))] [, ROTATE=[x, y, z]] [, SCALE=[x, y, z]] [, TRANSLATE=[x, y, z]] [, /XYEXCH [, /XZEXCH] [, /YZEXCH]

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=array, 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_IN=array] [, MOMENT=variable] )

STRTRIM
**THIN** - Returns the “skeleton” of a bi-level image.

Result = THIN(Image[, , NEIGHBOR_COUNT]
         [, /PRUNE])

**THREED** - Plots a 2D array as a pseudo 3D plot.

         [ , 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]
                    [, DAY=value] [, HOUR=value] [, MINUTE=value] [, SECOND=value]
                    [, STEP_SIZE=value] [, UNITS=string ]
                    [, /YEAR=value ])

**TOTAL** - Sums of the elements of an array.

Result = TOTAL(Array [ , Dimension ] [, CUMULATIVE]
               [, /DOUBLE] [, /INTEGER] [, /NAN]
               [, /PRESELECT_TYPE] [, Thread pool keywords ])

**TRACE** - Computes the trace of an array.

Result = TRACE(A [, /DOUBLE])

**TrackBall Object** - See “TrackBall” on page 122.

**TRANSPOSE** - Transposes an array.

Result = TRANSPOSE(Array [, P])

**TRI_SURF** - Interpolates gridded set of points with a smooth quintic surface.

Result = TRI_SURF(Z [, X, Y] [, /EXTRAPOLATE]
                 [, /MISSING=value] [, /REGULAR] [, /GRID=xstart, xspacing]
                 [, XVALUES=variable ] [, YGRID=xstart, yspacing]
                 [, YVALUES=variable ] [, 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]]
       [, FVALUE=variable [, /REPEATS=variable]]
       [, TOLERANCE=value ]

**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)

**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]

Graphics Keywords: [, /DATA [, /DEVICE [, /NORMAL [, /T3D [, Z=value]]]]]]

**TVLCT** - Loads display color tables.

TVLCT, V, V1, V2 [, 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 [, XN [, YN [, CHANNEL]]]]]
               [, CHANNEL=value] [, /ORDER [, TRUE=[1 | 2 | 3] [, /WORDS]])
UNIQ - Performs an unsharp-mask sharpening filter on a two-dimensional array or a TrueColor image.

UNIQ = UNSHARP_MASK(Image [, AMOUNT=value] [, THRESHOLD=value] [, TRUE=[1 2 3]])

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=variable] [, SCALE=value] [, VERTICES=variable]

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


VERS - Performs a unsharp-mask sharpening filter on a two-dimensional array or a TrueColor image.

VERS = UNIQ( Image [, X [, Y [, Channel]]] ) or TVSCL, Image [ X, Y, Channel] ]

Keywords: [/CENTIMETERS | /INCHES] [, /NAN] [, /ORDER] [, TOP=value] [, TRUE=[1 2 3]] [, /WORDS] [, XSIZE=value] [, YSIZE=value]

Graphics Keywords: [, CHANNEL=value] [, /DATA] [, /DEVICE] [, /NORMAL] [, /T3D] Z=value [Thread pool keywords]

TVSCL - Scales and displays an image.

TVSCL, Image [, Position]

or TVSCL, Image [ X, Y, Channel]]

Keywords: [/CENTIMETERS | /INCHES] [, /NAN] [, /ORDER] [, TOP=value] [, TRUE=[1 2 3]] [, /WORDS] [, XSIZE=value] [, YSIZE=value]

Graphics Keywords: [, CHANNEL=value] [, /DATA] [, /DEVICE] [, /NORMAL] [, /T3D] Z=value [Thread pool keywords]

ULINDGEN - Returns unsigned integer array with each element set to its subscript.

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

UINT - Converts argument to unsigned integer type.

Result = UINT( Expression [, Offset [, D1 [, ... , D8]]] [, Thread pool keywords] )

UINTARR - Returns an unsigned integer vector or array.

Result = UINTARR( D1 [, ... , D8] [, /NOZERO] )

ULINDGEN - Returns an unsigned longword array with each element set to its subscript.

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

ULBIG64ARR - Returns an unsigned 64-bit integer vector or array.

Result = ULBIG64ARR( D1 [, ... , D8] [, /NOZERO] )

ULONARR - Returns an unsigned longword integer vector or array.

Result = ULONARR( D1 [, ... , D8] [, /NOZERO] )

ULONG - Converts argument to unsigned longword integer type.

Result = ULONG( Expression [, Offset [, D1 [, ... , D8]]] [, Thread pool keywords] )

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

Result = ULONG64( Expression [, Offset [, D1 [, ... , D8]]] [, Thread pool keywords] )

UNIQ - Returns subscripts of the unique elements in an array.

Result = UNIQ( Array [, Index] )

UNISHARP_MASK - Performs an unsharp-mask sharpening filter on a two-dimensional array or a TrueColor image.

Result = UNISHARP_MASK( Image [, AMOUNT=value] [, RADIUS=value] [, THRESHOLD=value] [, TRUE=[1 2 3]])

USERSYM - Defines a new plotting symbol.

USERSYM, X [, Y] [, COLOR=value] [, /FILL] [, THICK=value]
Alphabetical List of IDL Routines

WARP_TRI - Warps an image using control points.

   Result = WARP_TRI Xc, Yc, 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])

DELETE - Deletes IDL graphics windows.

   WDELETE (Window_Index, ...)

WF_DRAW - Draws weather fronts with smoothing.

   WF_DRAW X, Y [, /COLD] [, /AWARM] [, /COCLUDED]
   [, /FRONT_TYPE=3] [, /STATIONARY] ,
   [:CONVERGENCE] [, /FRONT_TYPE=5] [, /COLOR=value]
   [, /DATA] [, /DEVICE] [, /NORMAL] [, /INTERVAL=value]
   [, /FSYM=value] [, SYM_HTB=value]
   [, SYM_LEN=value] [, THICK=value]

WHERE - Returns subscripts of nonzero array elements.

   Result = WHERE ( Array_Expression [, Count]
   [. COMPLEMENT=variable] [. /L64]
   [, NCOMPLEMENT=variable] [, Thread pool
   keywords])

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

WIDGET_ACTIVEX - Create an ActiveX control and place it into an
IDL widget hierarchy.

   Result = WIDGET_ACTIVEX ( Parent, COM_ID,
   [, /ALIGN_BOTTOM] [, /ALIGN_CENTER]
   [, /ALIGN_LEFT] [, /ALIGN_RIGHT] [, /ALIGN_TOP]
   [, EVENT_FUNC=string] [, EVENT_PRO=string]
   [, FUNC_GET_VALUE=string] [, ID_TYPE=value]
   [, KILL_NOTIFY=string] [, /NO_COPY]
   [, NOTIFY_REALIZE=string]
   [, PRO_SET_VALUE=string] [, SCR_XSIZE=width]
   [, SCR_YSIZE=height] [, /SENSITIVE]
   [, UNAME=string] [, UNITS={0 | 1 | 2}]
   [, UVALUE=value] [, XOFFSET=value]
   [, XSIZE=value] [, YOFFSET=value] [, YSIZE=value]
   [, /BASE_ALIGN_RIGHT] [, /BASE_ALIGN_TOP]
   [, /BITMAP=value] [, COLUMNN=cols] [, ROW=nrows]
   [, /CONTEXT_EVENTS] [, /CONTEXT_MENU]
   [, EVENT_FUNC=string] [, EVENT_PRO=string]
   [, /EXCLUSIVE] [, /NONEXCLUSIVE] [, /FLOATING]
   [, FRAME=width] [, FUNC_GET_VALUE=string]
   [, /GRID_LAYOUT]
   [, GROUP_LEADER=widget_id{must specify for modal
dialogs}] [, /KBRD_FOCUS_EVENTS]
   [, KILL_NOTIFY=string] [, /MAP{not for modal bases}]
   [, /MASK] [, /NO_COPY]
   [, NOTIFY_REALIZE=string]

WIDGET_BASE - continued

   [, /PRO_SET_VALUE=string] [, SCR_XSIZE=width]
   [, SCR_YSIZE=height] [, /SCROLL{not for modal
   bases}] [, /SENSITIVE] [, SPACE=value{ignored if
   exclusive or nonexclusive}] [, TAB_MODE=value]
   [, TITLE=string] [, TLB_FRAME_ATTRIB=value{top-
   level bases only}] [, TLB_ICONIFY_EVENTS{top-
   level bases only}]
   [, /TBK_REQUEST_EVENTS{top-level bases only}]
   [, /TBK_MOVE_EVENTS{top-level bases only}]
   [, /TBK_SIZE_EVENTS{top-level bases only}]
   [, /TOOLBAR] [, /TRACKING_EVENTS]
   [, UNAME=string] [, UNITS={0 | 1 | 2}]
   [, UVALUE=value] [, YOFFSET=value]
   [, XOFFSET=value] [, XSIZE=value] [, YSIZE=value]

   [, /BASE_ALIGN_RIGHT] [, /BASE_ALIGN_TOP]
   [, /BITMAP=value] [, COLUMNN=cols] [, ROW=nrows]
   [, /CONTEXT_EVENTS] [, /CONTEXT_MENU]
   [, EVENT_FUNC=string] [, EVENT_PRO=string]
   [, /EXCLUSIVE] [, /NONEXCLUSIVE] [, /FLOATING]
   [, FRAME=width] [, FUNC_GET_VALUE=string]
   [, /GRID_LAYOUT]
   [, GROUP_LEADER=widget_id{must specify for modal
dialogs}] [, /KBRD_FOCUS_EVENTS]
   [, KILL_NOTIFY=string] [, /MAP{not for modal bases}]
   [, /MASK] [, /NO_COPY]
   [, NOTIFY_REALIZE=string]

WIDGET_BUTTON - Creates button widgets.

   Result = WIDGET_BUTTON( Parent,
   [, /ACCELERATOR=string] [, /ALIGN_CENTER]
   [, /ALIGN_LEFT] [, /ALIGN_RIGHT] [, /BITMAP]
   [, /CHECKED_MENU] [, /DYNAMIC_RESIZE]
   [, EVENT_FUNC=string] [, EVENT_PRO=string]
   [, FONT=string] [, FRAME=width]
   [, FUNC_GET_VALUE=string]
   [, GROUP_LEADER=widget_id{must specify for modal
   bases}] [, /HELP]
   [, KILL_NOTIFY=string] [, /MENU] [, /NO_COPY]
   [, /NO_RELEASE] [, NOTIFY_REALIZE=string]
   [, PRO_SET_VALUE=string]
   [, /PUSHBUTTON_EVENTS] [, SCR_XSIZE=width]
   [, SCR_YSIZE=height] [, /SENSITIVE]
   [, /SEPPARATOR] [, TAB_MODE=value]
   [, TOOLTIP=string] [, /TRACKING_EVENTS]
   [, UNAME=string] [, UNITS={0 | 1 | 2}]
   [, UVALUE=value] [, VALUE=value]
   [, X_BITMAP_EXTRA=bits] [, XOFFSET=value]
   [, XSIZE=value] [, YOFFSET=value] [, YSIZE=value] )

IDL Quick Reference

WIDGET_BUTTON
WIDGET_COMBOBOX - Creates editable droplist widgets.

Result = WIDGET_COMBOBOX( Parent
[. /DYNAMIC_RESIZE] [. /EDITABLE]
[. EVENT_FUNC=str(string) [. EVENT_PRO=str(string)]
[. FONT=str(string) [. FRAME=value]
[. FUNC_GET_VALUE=str(string)
[. GROUP_LEADER=widget_id]
[. IGNORE_ACCELERATORS=value]
[. KILL_NOTIFY=str(string) [. /NO_COPY]
[. NOTIFY_REALIZE=str(string]
[. PRO_SET_VALUE=str(string)

WIDGET_COMBOBOX - continued
[. RESOURCE_NAME=str(string) [. SCR_XSIZE=value]
[. SCR_YSIZE=value [. /SENSITIVE]
[. TAB_MODE=value [. /TRACKING_EVENTS]
[. UNAME=str(string) [. UNITS={0 | 1 | 2}]
[. UVALUE=value [. VALUE=value]
[. XOFFSET=value [. YOFFSET=value]
[. YOFFSET=value [. XSIZE=value]

WIDGET_CONTROL - Realizes, manages, and destroys widgets.

WIDGET_CONTROL[, Widget_ID]
All widgets: [. BAD_ID=variable [. /CLEAR_EVENTS]
[. DEFAULT_FONT=str(string) [. /NO_AXES]
[. DELAY_DESTROY=do not specify Widget_ID]]
[. DESTROY= [. EVENT_FUNC=str(string)
[. EVENT_PRO=str(string)
[. GET_UVALUE=str(string)
[. GROUP_LEADER=widget_id]
[. KILL_NOTIFY=str(string)
[. MAP= [. NO_COPY]
[. NOTIFY_REALIZE=str(string]
[. PRO_SET_VALUE=str(string)
[. /PUSHBUTTON_EVENTS] [. /REALIZE]
[. /RESET=do not specify Widget_ID]]
[. SCR_XSIZE=value [. SCR_YSIZE=value [. /NO_COPY]
[. SEND_EVENT=structure [. /NO_TIMEOUT]
[. SHOW [. TIMER=value]
[. TLB_GET_OFFSET=variable]
[. TLB_GET_SIZE=variable]
[. TLB_REQUEST_EVENTS]
[. TLB_SET_TITLE=str(string)
[. TLB_SET_YOFFSET=value]
[. TRACKING_EVENTS [. UNITS={0 | 1 | 2}]
[. UPDATE= [. XOFFSET=value [. XSIZE=value]
[. YOFFSET=value [. YSIZE=value]

WIDGET_BASE: [. /CONTEXT_EVENTS]
[. /ICONIFY [. /KBD_FOCUS_EVENTS]
[. TAB_MODE=value [. /TLB_ICONIFY_EVENTS]
[. TLB_KILL_REQUEST_EVENTS]
[. TLB_MOVE_EVENTS [. /TLB_SIZE_EVENTS]

WIDGET_BUTTON: [. /BITMAP]
[. /DYNAMIC_RESIZE] [. GET_VALUE=value]
[. /INPUT_FOCUS [. /PUSHBUTTON_EVENTS]
[. /SET_BUTTON [. SET_VALUE=value]
[. TAB_MODE=value] [. TOOLTIP=str(string)
[. X_BITMAP_EXTRA=bits]

WIDGET_COMBOBOX:
[. COMBOBOX_ADDITEM=str(string)
[. COMBOBOX_DELETEITEM=integer]
[. COMBOBOX_INDEX=integer]
[. /DYNAMIC_RESIZE] [. GET_VALUE=value]
[. IGNOR.ACCELERATORS=[string_array][0 | 1]]
[. SET_COMBOBOX_SELECT=integer]
[. SET_VALUE=value [. TAB_MODE=value]

WIDGET_DRAW: [. /DRAW_BUTTON_EVENTS]
[. /DRAW_EXPOSE_EVENTS]
[. /DRAW_KEYBOARD_EVENTS={0 | 1 | 2}]
[. /DRAW_MOTION_EVENTS]
[. /DRAWExpose_EVENTS]
[. DRAW_KEYBOARD_EVENTS={0 | 1 | 2}]
[. DRAW_VIEWPORT_EVENTS]
[. DRAW_WHEEL_EVENTS]
[. DRAW_XSIZE=integer [. DRAW_YSIZE=integer]
[. GET_DRAW_VIEW=variable]
[. GET_UVALUE=str(int=value)
[. IGNOR.ACCELERATORS=[string_array][0 | 1]]
[. INPUT_FOCUS [. SET_DRAW_NOTIFY=str(string)
[. SET_DRAW_VIEW=x, y [. /SET_DROP_EVENTS]
[. TOOLTIP=str(string)

WIDGET_DROPLIST: [. /DYNAMIC_RESIZE]
[. GET_VALUE=value]
[. GET_DROP_LIST_SELECT=integer]
[. SET_DROP_LIST_Select=integer]
[. SET_VALUE=value [. TAB_MODE=value]

WIDGET_LABEL: [. /DYNAMIC_RESIZE]
[. GET_VALUE=value]
[. SET_LABEL_SELECT=integer]
[. SET_VALUE=value [. TAB_MODE=value]

WIDGET_LIST: [. /CONTEXT_EVENTS]
[. /EDITABLE]
[. GET_VALUE=value]
[. IGNORE.ACCELERATORS=[string_array][0 | 1]]
[. MULTIPLE_PROPERTIES]
[. PROPERTY_SET_SELECTED=empty string, string, or array of strings]
[. REFRESH_PROPERTY=x, string array, or integer [. SET_VALUE=value]

WIDGET_SLIDER: [. GET_VALUE=value]
[. SET_SLIDER_MAX=value]
[. SET_SLIDER_MIN=value [. SET_VALUE=value]
[. TAB_MODE=value]
WIDGET_CONTROL - continued

WIDGET_TABLE: [.alignment={0 | 1 | 2}]
[.all_table_events] [.am_pm={string, string}]
[.background_color={array}]
[.column_labels={array}]
[.column_widths={array}]
[.context_events]
[.days_of_week={array}]
[.delete_columns={not for row_major mode}]
[.delete_rows={not for column_major mode}]
[.disjoint_selection] [.editable]
[.edit_cell={integer, integer}]
[.font={string}]
[.foreground_color={array}] [.format={variable}]
[.get_value={variable}]
[.ignore_accelerators={string, array}]
[.insert_columns={variable}]
[.insert_rows={value}]
[.kbrd_focus_events]
[.months={array}]
[.row_labels={array}]
[.row_heights={array}]
[.set_table_select={left, top, right, bottom}]
[.set_table_view={integer, integer}]
[.set_text_select={integer, integer}]
[.set_value={variable}]
[.tab_mode={variable}]
[.table_blank={cells}]
[.table_disjoint_selection]
[.table_xsize={columns}]
[.table_ysize={rows}]
[.use_table_select [.use_text_select={left, top, right, bottom}]
[.use_text_select]
WIDGET_TEXT: [.append] [.context_events] [.editable]
[.get_value={variable}]
[.ignore_accelerators={string, array}]
[.input_focus] [.kbrd_focus_events]
[.no_newline] [.set_text_select={integer, integer}]
[.set_text_top_line={integer, integer}]
[.set_value={variable}]
[.tab_mode={variable}]
[.use_text_select]
WIDGET_TREE: [.context_events]
[.get_value={variable}]
[.set_drag_notify={string}]
[.set_draggable] [.set_drop_events]
[.set_mask] [.set_tree_bitmap={array}]
[.set_tree_expanded]
[.set_tree_index={variable}]
[.set_tree_select={0 | 1 | widget ID | array of widget IDs}]
[.set_value={variable}]
[.tab_mode={variable}]

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] [.classname={string}]
[.color_model] [.colors={integer, string}]
[.drag_notify={string}]
[.drop_events]
[.event_func={string}]
[.event_pro={string}]
[.expose_events]
[.frame={width}]
[.func_get_value={string}]
[.graphics_level={2}]
[.group_leader={widget ID}]
[.ignore_accelerators={variable}]
[.keyboard_events={0 | 1 | 2}]
[.kill_notify={string}]
[.motion_events]
[.no_copy]
[.notify_realize={string}]
[.pro_set_value={string}]
[.renderers={0 | 1}]
[.resource_name={string}]
[.retain={0 | 1 | 1}]
[.scr_xsize={width}]
[.scr_ysize={height}]
[.scroll_controller]
[.sensitive]
[.tooltip={string}]
[.tracking_events]
[.uname={string}]
[.units={0 | 1 | 2}]
[.value={variable}]
[.viewport_events]
[.wheel_events]
[.x_scroll_size={width}]
[.y_offset={value}]
[.y_scroll_size={height}]
[.yscroll={value}]
[.y_size={variable}]

WIDGET_DROPLIST - Creates droplist widgets.

Result = WIDGET_DROPLIST(Parent [.dynamic_resize]
[.event_func={string}]
[.event_pro={string}]
[.font={string}]
[.frame={variable}]
[.func_get_value={string}]
[.group_leader={widget ID}]
[.kill_notify={string}]
[.no_copy]
[.notify_realize={string}]
[.pro_set_value={string}]
[.resource_name={string}]
[.retain={0 | 1 | 2}]
[.scr_xsize={width}]
[.scr_ysize={height}]
[.scroll_controller]
[.sensitive]
[.tab_mode={variable}]
[.ttitle={string}]
[.tracking_events]
[.uname={string}]
[.units={0 | 1 | 2}]
[.value={variable}]
[.x_offset={variable}]
[.x_scroll_size={width}]
[.y_offset={variable}]
[.y_scroll_size={height}]

WIDGET_EVENT - Returns events for the widget hierarchy.

Result = WIDGET_EVENT(Widget_ID)
[.bad_id={variable}]
[.nowait]
[.save_hourglass]
UNIX Keywords: [.yield_to_tty]
WIDGET_INFO - Obtains information about widgets.

Result = WIDGET_INFO( [Widget_ID] )

All widgets: [.ACTIVE] [.ALL_CHILDREN]
[.CHILD] [.EVENT_FUNC] [.EVENT_PRO]
[.FINDBY_UNAME=string] [.FONTNAME]
[.GEOMETRY] [.KBRSDD_FOCUS_EVENTS]
[.MANAGED] [.MAP] [.IN_CHILDREN] [.NAME]
[.PARENT] [.PUSHBUTTON_EVENTS]
[.REALIZED] [.SENSITIVE] [.SIBLING]
[.STRING_SIZE={string [[string, font]]}]
[.SYSTEM_COLORS] [.TAB_MODE=value]
[.TRACKING_EVENTS] [.TYPE]
[.UNAME] [.UPDATE] [.VALID_ID]
[.VERSION] [.VISIBLE]

WIDGET_BASE: [.CONTEXT_EVENTS]
[.MODAL] [.TLB_ICONIFY_EVENTS]
[.TLB_KILL_REQUEST_EVENTS]
[.TLB_MOVE_EVENTS] [.TLB_SIZE_EVENTS]

WIDGET_BUTTON: [.BUTTON_SET] [.DYNAMIC_RESIZE] [.PUSHBUTTON_EVENTS]
[.TOOLTIP]


WIDGET_DRAW: [.DRAW_NOTIFY] [.DRAW_BUTTON_EVENTS]
[.DRAW_EXPOSE_EVENTS]
[.DRAW_KEYBOARD_EVENTS]
[.DRAW_MOTION_EVENTS]
[.DRAW_WHEEL_EVENTS] [.DROP_EVENTS]
[.TOOLTIP]

WIDGET_DROPLIST: [.DROPLIST_NUMBER] [.DROPLIST_SELECT] [.DYNAMIC_RESIZE]

WIDGET_LIST: [.CONTEXT_EVENTS]
[.LIST_MULTIPLE] [.LIST_NUMBER]
[.LIST_VISIBLE] [.LIST_SELECT] [.LIST_TOP]

WIDGET_PROPERTIESHEET:
[.CONTEXT_EVENTS] [.COMPONENT=obref]
[.MULTIPLE_PROPERTIES]
[.PROPERTY_VALID=string]
[.PROPERTY_VALUE=string]
[.PROPERTYSET_NSELECTED]
[.PROPERTYSET_SELECTED]

WIDGET_SLIDER: [.SLIDER_MIN_MAX]

WIDGET_TAB: [.TAB_CURRENT]
[.TAB_MULTILINE] [.TAB_NUMBER]

WIDGET_INFO - continued

WIDGET_TABLE: [.COLUMN_WIDTHS]
[.CONTEXT_EVENTS] [.ROW_HEIGHTS]
[.STRING_SIZE] [.TABLE_ALL_EVENTS]
[.TABLE_BACKGROUND_COLOR]
[.TABLE_DISJOINT_SELECTION]
[.TABLE_EDITABLE] [.TABLE_EDIT_CELL]
[.TABLE_FONT]
[.TABLE_FOREGROUND_COLOR]
[.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]
[.DRAGGABLE] [.DRAG_NOTIFY]
[.DROP_EVENTS] [.MASK] [.TREE_BITMAP]
[.TREE_DRAG_SELECT] [.TREE_EXPANDED]
[.TREE_FOLDER] [.TREE_INDEX]
[.TREE_ROOT] [.TREE_SELECT]

WIDGET_LABEL - Creates label widgets.

Result = WIDGET_LABEL( Parent [, .ALIGN_CENTER]
[.ALIGN_LEFT] [.ALIGN_RIGHT]
[.DYNAMIC_RESIZE] [.FONT=string]
[.FRAME=width] [.FUNC_GET_VALUE=string]
[.GROUP_LEADER=widget_id]
[.KILL_NOTIFY=string] [.NO_COPY]
[.NOTIFY_REALIZE=string]
[.PRO_SET_VALUE=string]
[.RESOURCE_NAME=string] [.SCR_XSIZE=width]
[.SCR_YSIZE=height] [.SENSITIVE]
[.SUNKEN_FRAME] [.TRACKING_EVENTS]
[.UNAME=string] [.UNITS={0 | 1 | 2}]
[.VALUE=value] [.VALUE=value]
[.OFFSET=value] [.XSIZE=value]
[.OFFSET=value] [.YSIZE=value]

WIDGET_LIST - Creates list widgets.

Result = WIDGET_LIST( Parent [.CONTEXT_EVENTS]
[.EVENT_FUNC=string]
[.EVENT_PRO=string] [.FONT=string]
[.FRAME=width] [.FUNC_GET_VALUE=string]
[.GROUP_LEADER=widget_id]
[.KILL_NOTIFY=string] [.MULTIPLE]
[.NO_COPY] [.NOTIFY_REALIZE=string]

WIDGET_LIST - continued
[.PRO_SET_VALUE=string]
[.RESOURCE_NAME=string] [.SCR_XSIZE=width]
[.SCR_YSIZE=height] [.SENSITIVE]
[.TAB_MODE=value] [.TRACKING_EVENTS]
WIDGET_RTABLE - Creates property sheet widgets, which exposes the properties of an IDL object. This widget transparently handles property value changes.


WIDGET_TAB - Creates tab widgets.

WIDGET_TREE - Creates tree widgets.


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


WINDOW - Creates window for the display of graphics or text.


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


WRITE_GIF - Writes a Graphics Interchange Format (GIF) file.


WRITE_IMAGE - Writes 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 - Writes a JPEG file.

WRITE_JPEG[ , Filename] [, UNIT=lin] [, Image [, /ORDER] [, /PROGRESSIVE] [, QUALITY=0 to 100]] [, TRUE={1 | 2 | 3}] )


WRITE_NRF - Writes NCAR Raster Interchange Format rasterfile.

WRITE_NRF, File, Image [, R, G, B] )

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


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

WRITE_PPM, Filename, Image [, JASC] )

WRITE_SPR - Writes row-indexed sparse array structure to a file.

WRITE_SPR, AS, Filename )

WRITE_SRF - Writes Sun Raster File (SRF).


WRITE_SYLK - Writes SYLK (Symbolic Link) spreadsheet file.


WRITE_SPR - Writes Sun Raster File (SRF).


WRITE_SYLK - Writes SYLK (Symbolic Link) spreadsheet file.


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

WRITE_PPM, Filename, Image [, JASC] )

WRITE_SPR - Writes row-indexed sparse array structure to a file.

WRITE_SPR, AS, Filename )

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 audio stream to the named .WAV file.

**WRITE_WALL, WRITEU** - Writes unformatted binary data to a file.

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

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

**XMANAGER** - Provides event loop manager for IDL widgets.

**XMING_TMPL** - Template for creating widgets.

**XMOD** - Displays tool for viewing XMANAGER widgets.

**XOBJVIEW** - Displays object viewer widget.

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

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

XOBJVIEW_ROTATE - Programmatically rotate the object currently displayed in XOBJVIEW.
XOBJVIEW_ROTATE, Axis, Angle [, /PREMULTIPLY]

XOBJVIEW_WRITE_IMAGE - Write the object currently displayed in XOBJVIEW to an image file.
XOBJVIEW_WRITE_IMAGE, Filename, Format
[. DIMENSIONS=x, y]}

XPALETTE - Displays palette used to create and modify color tables.
XPALETTE [, /BLOCK] [, GROUP=widget_id]
[. UPDATECALLBACK=’procedure_name’
[. UPDATECDBDATA=value]]

XPCOLOR - Adjusts the value of the current foreground plotting color. /PCOLOR

XPCOLOR [, GROUP=widget_id ]

XPLT3D - Utility for creating and interactively manipulating 3D plots.
XPLT3D, X, Y, Z [, /BLOCK] [, COLOR=[r, g, b]]
[. /DOUBLE_VIEW] [, GROUP=widget_id]
[. LINESTYLE=1 | 2 | 3 | 4 | 5 | 6] [, /MODAL]
[. NAME=string [, /OVERPLOT]
[. SYMBOL=’style’ [, /TEST]
[. THICK=points (1.0 to 10.0)] [, TITLE=string]
[. XRANGE=[min, max]] [, YRANGE=[min, max]]
[. ZRANGE=[min, max]] [, XTITLE=string]
[. YTITLE=string]] [, /T3D]

XREGISTERED - Returns registration status of a given widget.
Result = XREGISTERED(Widget [, /NOSHOW] )

XROI - Utility for interactively creating and obtaining information about ROIs.
[. /FLOATING] [, GROUP=widget_id] [, /MODAL]
[. REGIONS_IN=value] [, REGIONS_OUT=value]
[. REJECTED=variable] [, RENDERER=[0 | 1]]
[. ROI_COLOR=[r, g, b] or variable]
[. ROI_GEOMETRY=variable]
[. ROI_SELECT_COLOR=[r, g, b] or variable]
[. STATISTICS=variable] [, TITLE=string]
[. TOOLS=’string or string array’ (valid values are ‘Freehand Draw’, ‘Polygon Draw’, and ‘Selection’)]
[. X_SCROLL_SIZE=value]
[. Y_SCROLL_SIZE=value]

XSQ_TEST - Computes Chi-square goodness-of-fit test.
Result = XSQ_TEST( Obsfreq, Exfreq
[. EXCELL=variable] [, OBCELL=variable]
[. RESIDUAL=variable] )

XSURFACE - Provides GUI to SURFACE and SHADE_SURF.
XSURFACE, Data [, /BLOCK] [, GROUP=widget_id]

XVAREDIT - Provides widget-based editor for IDL variables.
XVAREDIT, Var [, NAME=variable_name’ (ignored if variable is a structure)] [, GROUP=widget_id]
[. X_SCROLL_SIZE=columns]
[. Y_SCROLL_SIZE=rows]

XVOLUME - Utility for viewing and interactively manipulating volumes and isosurfaces.
XVOLUME, Vol [, /BLOCK] [, GROUP=widget_id]
[. /INTERPOLATE] [, /MODAL] [, RENDERER=[0 | 1]] [, /REPLACE] [, /SCALE=value] [, /TEST]
[. XSIZE=pixels] [, YSIZE=pixels]

XYOUTS - Draws text on currently-selected graphics device.
XYOUTS, [X, Y] String [, ALIGNMENT=value (0.0 to 1.0)] [, /INTERP]
[. CHARSIZE=value] [, /T3D]
[. CHARTHICK=value]
[. TEXT_AXES=0 | 1 | 2 | 3 | 4 | 5]
[. WIDTH=value]

Graphics Keywords: [ CLIP=[X0, Y0, X1, Y1]]
[. COLOR=value] [, /DEVICE | /NORMAL]
[. FONT=integer]
[. ORIENTATION=ccw_degrees_from_horizontal]
[. /NOCLIP] [, /T3D]
[. Z=value]

Z

ZOOM - Zooms portions of the display.
ZOOM [, /CONTINUOUS] [, FACT=integer] [, /INTERP]
[. /KEEP] [, /NEW_WINDOW] [, XSIZE=value]
[. YSIZE=value] [, ZOOM_WINDOW=value]

ZOOM_24 - Zooms portions of true-color (24-bit) display.
ZOOM_24 [, FACT=integer] [, /RIGHT] [, XSIZE=value]
[. YSIZE=value]

Scientific Data Formats

See the online version of this document, quickref.pdf located on your IDL CD-ROM for scientific data format routine information.
Objects

This section lists all IDL objects and their methods. In addition to the syntax conventions discussed in "IDL Syntax Conventions" on page 28, 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_IDLBridge - Creates and controls a synchronous or an asynchronous child IDL process that can operate sequentially or in parallel with the main IDL process.

Properties: [], CALLBACK{Get, Init, Set}=

IDL_IDLBridge::Abort - Aborts the current asynchronous child IDL process.

IDL_IDLBridge::Cleanup - Performs all cleanup on the object.

IDL_IDLBridge::Execute - Causes the child IDL process to execute a given IDL statement in synchronous or asynchronous mode.

IDL_IDLBridge::GetProperty - Retrieves the value of a property or group of properties for the bridge.

IDL_IDLBridge::GetVar - Copies the value of a variable from the main IDL process to the child process under the given name.

IDL_IDLBridge::Init - Initializes an IDL bridge object that lets you create and control a child IDL session.

IDL_IDLBridge::OnCallback - When an asynchronous command ens, this property is automatically called. If the CALLBACK property is set, the callback procedure specified by the property is executed.

IDL_IDLBridge::SetProperty - Sets the value of a property or group of properties for the bridge.
IDL IDLBridge::Status - Returns the state of an asynchronous child process (one executed with the NOWAIT keyword set).
Result = Obj->IDL IDLBridge::Status([ERROR=ErrorString])

IDL Savefile - Object that provides complete query and restore capabilities for IDL SAVE file.
Properties: None.

IDL Savefile::Cleanup - Performs all cleanup on the object.
Obj->IDL Savefile::Cleanup

IDL Savefile::Contents - Returns a structure variable of type IDL_SAVEFILE_CONTENTS containing information about the associated SAVE file and its contents.
Result = Obj->IDL Savefile::Contents()

IDL Savefile::Init - Initializes the Savefile object.
Obj = OBJ_NEW('IDL Savefile' [, FileName ] [, FILENAME=string ] [, /RELAXED_STRUCTURE_ASSIGNMENT])
Result = Obj->IDL Savefile::Init ([, FileName ] [, FILENAME=string ] [, /RELAXED_STRUCTURE_ASSIGNMENT ])

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.
Result = Obj->IDL_Savefile::Size(SaveItem)

IDLanROI - Represents a region of interest. Superclass of IDLanROI.

IDLanROI::SetProperty - Sets the value of a property or group of properties for the region.
Obj->IDLanROI::SetProperty[, PROPERTY=value]
### Alphabetical List of IDL Routines

- **IDLanROI::Translate** - Modifies the vertices for the region by applying a translation.
  
  \[ \text{Obj} \rightarrow \text{IDLanROI::Translate}, \{T_x, T_y, T_z\} \]

- **IDLanROIGroup** - This object is an analytical representation of a group of regions of interest. Subclass of IDLgrROIGroup. Subclass of IDLgrROIGroup.

  **Properties:**
  - [ALL] Get=[variable]
  - [ROIGroup_XRANGE] Get=[variable]
  - [ROIGroup_YRANGE] Get=[variable]
  - [ROIGroup_ZRANGE] Get=[variable]

- **IDLanROIGroup::Add** - Adds a region to the region group.
  
  \[ \text{Obj} \rightarrow \text{IDLanROIGroup::Add}, \text{ROI} \]

- **IDLanROIGroup::Cleanup** - Performs all cleanup for the object.
  
  \[ \text{OBJ\_DESTROY} \] or \[ \text{Obj} \rightarrow \text{IDLanROIGroup::Cleanup} \]

- **IDLanROIGroup::ContainsPoints** - Determines whether the given points (in data coordinates) are contained within the closed polygon regions within this group.
  
  \[ \text{Result} = \text{Obj} \rightarrow \text{IDLanROIGroup::ContainsPoints}(X[, Y[, Z]]) \]

- **IDLanROIGroup::ComputeMask** - Prepares a 2-D mask for this group of regions.
  
  \[ \text{Result} = \text{Obj} \rightarrow \text{IDLanROIGroup::ComputeMask}(\text{[,, INITIALIZE}=[-1] 0 1]) \]

- **IDLanROIGroup::ComputeMesh** - Triangulates a surface mesh with optional capping from the stack of regions contained within this group.
  
  \[ \text{Result} = \text{Obj} \rightarrow \text{IDLanROIGroup::ComputeMesh}(\text{Vertices, Conn [, CAPPED}=[0] 1 2]) \]

- **IDLanROIGroup::GetProperty** - Retrieves the value of a property or group of properties for the region group.
  
  \[ \text{Obj} \rightarrow \text{IDLanROIGroup::GetProperty}[\text{,, PROPERTY}=[\text{variable}]} \]

- **IDLanROIGroup::Init** - Initializes a region of interest group object.
  
  \[ \text{Obj} = \text{OBJ\_NEW}('\text{IDLanROIGroup}') \text{ or } \text{Result} = \text{Obj} \rightarrow \text{IDLanROIGroup::Init}() \]

- **IDLanROIGroup::Rotate** - Modifies the vertices for all regions within the group by applying a rotation.
  
  \[ \text{Obj} \rightarrow \text{IDLanROIGroup::Rotate}, \text{Azis}, \text{Angle}[, \text{CENTER}=[x, y, z]] \]

- **IDLanROIGroup::Scale** - Modifies the vertices for the region by applying a scale.
  
  \[ \text{Obj} \rightarrow \text{IDLanROIGroup::Scale}, \text{Sx}, \{\text{Sy}, \text{Sz}\} \]

- **IDLanROIGroup::Translate** - Modifies the vertices of all regions within the group by applying a translation.
  
  \[ \text{Obj} \rightarrow \text{IDLanROIGroup::Translate}, \{T_x, T_y, T_z\} \]

- **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 IDLcomIDispatch is created when the object is instantiated.

- **IDLcomDispatch::GetProperty** - Gets properties for an IDispatch interface.
  
  \[ \text{IDLcomDispatch} \rightarrow \text{GetProperty}, \text{PROPERTY}=[\text{variable}, \{arg0, arg1, \ldots\}] \]

- **IDLcomDispatch::Init** - Initializes a COM object and establish a link between the resulting IDL object and the IDispatch interface.
  
  \[ \text{Obj} = \text{OBJ\_NEW}('\text{IDLcomDispatchSIDTYPESID}') \]

- **IDLcomDispatch::SetProperty** - Sets properties for an IDispatch interface.
  
  \[ \text{IDLcomDispatch} \rightarrow \text{SetProperty}, \text{PROPERTY}=[\text{value}] \]

- **IDLffDICOM** - Contains the data for one or more images embedded in a DICOM part 10 file. No superclasses. No subclasses.

  **Property:**
  - [,, /VERBOSE] [,, Init]

- **IDLffDICOM::Cleanup** - Destroys the IDLffDICOM object.
  
  \[ \text{OBJ\_DESTROY} \] or \[ \text{Obj} \rightarrow \text{IDLffDICOM::Cleanup} \]

- **IDLffDICOM::DumpElements** - Dumps a description of the DICOM data elements of IDLffDICOM object to the screen or to a file.
  
  \[ \text{Obj} \rightarrow \text{IDLffDICOM::DumpElements}[,, \text{Filename}] \]

- **IDLffDICOM::GetChildren** - Finds the member element references of a DICOM sequence.
  
  \[ \text{array} = \text{Obj} \rightarrow \text{IDLffDICOM::GetChildren}(\text{Reference}) \]

- **IDLffDICOM::GetDescription** - Takes optional DICOM group and element arguments and returns array of STRING descriptions.
  
  \[ \text{array} = \text{Obj} \rightarrow \text{IDLffDICOM::GetDescription}(\text{[Group [, Element]}[, \text{REFERENCE}=[\text{list} of \text{element} \text{references}])} \]

- **IDLffDICOM::GetElement** - Takes optional DICOM group and/or element arguments and returns an array of DICOM Element numbers for those parameters.
  
  \[ \text{array} = \text{Obj} \rightarrow \text{IDLffDICOM::GetElement}(\text{[Group [, Element]}[, \text{REFERENCE}=[\text{list} of \text{element} \text{references}])} \]

- **IDLffDICOM::GetGroup** - Takes optional DICOM group and/or element arguments and returns an array of DICOM Group numbers for those parameters.
  
  \[ \text{array} = \text{Obj} \rightarrow \text{IDLffDICOM::GetGroup}(\text{[Group [, Element]}[, \text{REFERENCE}=[\text{list} of \text{element} \text{references}])} \]

- **IDLffDICOM::GetLength** - Takes optional DICOM group and/or element arguments and returns an array of LONGs.
  
  \[ \text{array} = \text{Obj} \rightarrow \text{IDLffDICOM::GetLength}(\text{[Group [, Element]}[, \text{REFERENCE}=[\text{list} of \text{element} \text{references}])} \]

- **IDLffDICOM::GetParent** - Finds the parent references of a set of elements in a DICOM sequence.
  
  \[ \text{array} = \text{Obj} \rightarrow \text{IDLffDICOM::GetParent}(\text{ReferenceList}) \]

- **IDLffDICOM::GetPreamble** - Returns the preamble of a DICOM v3.0 Part 10 file.
  
  \[ \text{array} = \text{Obj} \rightarrow \text{IDLffDICOM::GetPreamble}() \]
Alphabetical List of IDL Routines

IDLffDICOM::GetReference - Takes optional DICOM group and/or element arguments and returns an array of references to the elements matching in the object.

IDLffDXF::GetReference - Takes optional DXF 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 (Value Representation) STRINGs for those parameters.

IDLffDICOM::Write - Writes a file for the DICOM object information this object contains.

IDLffJPEG2000 - Object class used for reading and writing JPEG2000 files. No superclasses. No subclasses.

IDLffJPEG2000::GetData - Returns image data from the IDLffJPEG2000 object.

IDLffJPEG2000::GetProperty - Retrieves the value of a property or group of properties for the IDLffJPEG2000 object.

IDLffJPEG2000::GetProperty - Retrieves the value of a property or group of properties for the IDLffJPEG2000 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 object information this object contains.

IDLffDXF::Cleanup - Performs all cleanup on the object.

IDLffDXF::Init - Initializes the DXF 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::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 object information this object contains.

IDLffDXF::Init - Initializes the DXF 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::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 object information this object contains.

IDLffDXF::Init - Initializes the DXF 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::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 object information this object contains.

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 object information this object contains.

IDLffDXF::Init - Initializes the DXF 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.
IDLffJPEG2000::GetUUID - Retrieves the properties of a tile in an IDLffJPEG2000 object.
Obj->[IDLffJPEG2000::GetTileProperty [, TileComponent]] [ , N_LEVELS=variable]
[ , N_INDEX=variable] [ , PROGRESSION=variable]
[ , REVERSIBLE=variable]
[ , TILE_DIMENSIONS=variable]
[ , TILE_OFFSET=variable] [ , YCC=variable]

IDLffJPEG2000::GetTileProperty - Retrieves the properties of a tile in an IDLffJPEG2000 object.
Obj->[IDLffJPEG2000::GetTileProperty [ , TileComponent]] [ , N_LEVELS=variable]
[ , N_INDEX=variable] [ , PROGRESSION=variable]
[ , REVERSIBLE=variable]
[ , TILE_DIMENSIONS=variable]
[ , TILE_OFFSET=variable] [ , YCC=variable]

IDLffJPEG2000::Init - Initializes an IDLffJPEG2000 object.
Result = OBJ_NEW('IDLffJPEG2000', Filename
[ , PERSISTENT=0 [1] , PROPERTY=value])

IDLffJPEG2000::SetUUID - Allows you to add UUID boxes when creating a new JPEG2000 file.
Obj->[IDLffJPEG2000::SetUUID, uuid, Data]

IDLffLangCat - Object class is used to find and load an XML language catalog. No superclasses. No subclasses.
Properties: [ , APP_NAME = Get, Init, Set]=string
[ , APP_PATH = Get, Init, Set]=string
[ , AVAILABLE_LANGUAGES = Get]=string
[ , DEFAULT_KEYS = Get]=string
[ , DEFAULT_LANGUAGE = Get, Init]=string
[ , DEFAULT_N_KEYS = Get]=variable
[ , FILENAME = Get, Init]=string
[ , KEYS = Get]=variable
[ , LANGUAGE = Get, Init, Set]=string
[ , N_KEYS = Get]=variable
[ , VERBOSE = Get, Init, Set]=byte

IDLffLangCat::AppendCatalog - Adds keys from a file or files to those used to build the language catalog.
Result = Obj->[IDLffLangCat::]AppendCatalog
( [ , APP_NAME=applicationName]
[ , FILENAME=.filename]
[ , APP_PATH=applicationPath])

IDLffLangCat::Cleanup - Performs all cleanup operations on the language catalog object.
OBJ_DESTROY, Obj or
Obj->[IDLffLangCat::]Cleanup

IDLffLangCat::GetProperty - Retrieves a property or group of properties for an IDLffLangCat object.
Obj->[IDLffLangCat::]GetProperty
[ , PROPERTY=value]

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=value]

IDLffMJPEG2000 - Object class used for reading and writing Motion JPEG2000 animations. No superclasses. No subclasses.
Properties: [ , BIT_DEPTH = Get, Init, Set]=integer
[ , BIT_RATE = Init, Set]=integer
[ , CURRENT_FRAME = Get, Init, Set]=string
[ , DIMENSIONS = Get, Init, Set]=width, height
[ , DURATION = Get]=value
[ , FILENAME = Get]=string
[ , FRAME_BUFFER_LENGTH = Get, Init, Set]=integer
[ , FRAMES_IN_BUFFER = Get]=integer
[ , FRAMES_PERIOD = Get, Init, Set]=integer
[ , N_COMPONENTS = Get, Init, Set]=integer
[ , N_FRAMES = Get]=variable
[ , N_LAYERS = Get, Init, Set]=integer
[ , N_TILES = Get]=variable
[ , PALLETTE = Get, Init, Set]=array
[ , PROGRESSION = Get, Init, Set]=string
[ , REVERSIBLE = Get, Init, Set]=variable
[ , Scan_MODE = Get, Init, Set]=variable
[ , STATE = Get]=value
[ , TIMESCALE = Get, Init, Set]=variable
[ , WRITE = Get, Init, Set]=variable
[ , XML = Get, Init, Set]=variable
[ , YCC = Get, Init, Set]=variable

IDLffMJPEG2000::Cleanup - Deletes all IDLffMJPEG2000 objects.
OBJ_DESTROY, Obj or
Obj->[IDLffMJPEG2000::]Cleanup

IDLffMJPEG2000::Commit - Shuts down the processing thread and closes the MJ2 file.
Result = Obj->[IDLffMJPEG2000::]Commit(Wait)

IDLffMJPEG2000::GetData - Returns frame, component, tile or region data from the MJ2 file.
Result = Obj->[IDLffMJPEG2000::]GetData( FrameNumber
[ , COMPONENT=value] , DISCARD_LEVELS=value)
IDLffShape::GetEntity - Returns an array of entity structures from a Shapefile.
  \[Result = Obj->IDLffShape::GetEntity( [Index] [, ALL] [, N_ATTRIBUTES] )\]

IDLffShape::GetProperty - Returns the values of properties associated with a Shapefile object.
  \[Obj->IDLffShape::GetProperty [ATTRIBUTE_INFO=variable] [ATTRIBUTE_NAMES=variable] [ENTITY_TYPE=variable] [FILENAME=variable] [IS_OPEN=variable] [N_ATTRIBUTES=variable] [N_ENTITIES=variable] [N_RECORDS=variable]\]

IDLffShape::Init - Initializes or constructs a Shapefile object.
  \[Result = OBJ_NEW('IDLffShape' [, PROPERTY=value])\]

IDLffShape::Open - Opens a specified Shapefile.
  \[Result = Obj->IDLffShape::Open( 'Filename' [, /DBF_ONLY] [, /UPDATE] [ENTITY_TYPE=value])\]

IDLffShape::PutEntity - Inserts an entity into the Shapefile object.
  \[Obj->IDLffShape::PutEntity, Data\]

IDLffShape::SetAttributes - Modifies the attributes for a specified entity in a Shapefile object.
  \[Obj->IDLffShape::SetAttributes, Index, Attribute_Num, Value or Obj->IDLffShape::SetAttributes, Index, Attributes\]

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.

IDLffXMLDOMAttr::GetName - Returns the attribute node’s name.
  \[Result = Obj->IDLffXMLDOMAttr::GetName()\]

IDLffXMLDOMAttr::GetSpecified - Returns a scalar integer indicating how the attribute node’s value was set.
  \[Result = Obj->IDLffXMLDOMAttr::GetSpecified()\]

IDLffXMLDOMAttr::GetValue - Returns the attribute node’s value.
  \[Result = Obj->IDLffXMLDOMAttr::GetValue()\]

IDLffXMLDOMAttr::SetValue - Sets the attribute node’s value.
  \[Obj->IDLffXMLDOMAttr::SetValue, Value\]

IDLffXMLDOMCDATASection - Used to escape blocks of text in an XML document containing text that would otherwise be regarded as markup. Subclass of IDLffXMLDOMNode, IDLffXMLDOMCharacterData, and IDLffXMLDOMText.

IDLffXMLDOMCDATASection::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.

IDLffXMLDOMCharacterData - Extends the IDLffXMLDOM class with methods for accessing character data in the DOM tree. Subclass of IDLffXMLDOMNode.

IDLffXMLDOMCharacterData::AppendData - Appends a string to the node’s character data.
  \[Obj->IDLffXMLDOMCharacterData::AppendData, String\]

IDLffXMLDOMCharacterData::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.

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.
  \[Result = Obj->IDLffXMLDOMCharacterData::SubstringData(Offset, Count)\]

IDLffXMLDOMCharacterData::SubstringData (Offset, Count)
IDLffXMLDOMComment - Represents the content of a comment (characters between "<!--" and "-->") in an XML document. Subclass of IDLffXMLDOMCharacterData and IDLffXMLDOMText.

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. Should not be subclassed.

Result = Obj->[IDLffXMLDOMDocument::]Cleanup

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.

Properties: [:, NODE_DESTRUCTION_POLICY[Get, Init, Set]=0 | 1]

Destroys both the accessing object in the IDL tree and any objects created by that object; does not modify the actual DOM tree.

OBJ_DESTROY, Obj

or

Obj->[IDLffXMLDOMDocument::]Cleanup

IDLffXMLDOMDocument::CreateAttribute - Creates and names an attribute node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateAttribute(Name)

IDLffXMLDOMDocument::CreateCDATASection - Creates and fills a CDATASection node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateCDATASection(String)

IDLffXMLDOMDocument::CreateComment - Creates and fills a comment node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateComment(String)

IDLffXMLDOMDocument::CreateDocumentFragment - Creates a document fragment node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateDocumentFragment()

IDLffXMLDOMDocument::CreateElement - Creates and names with a tag name an element node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateElement(TagName)

IDLffXMLDOMDocument::CreateEntityReference - Creates and names an entity reference node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateEntityReference(Name)

IDLffXMLDOMDocument::CreateNodeIterator - Creates and names an instance of an IDLffXMLDOMNodeIterator object.

Result = Obj->[IDLffXMLDOMDocument::]CreateNodeIterator(Name [, FILTER_NAME=string] [, FILTER_USERDATA=variable] [, WHAT_TO_SHOW=value])

IDLffXMLDOMDocument::CreateProcessingInstruction - Creates and stores two strings in a ProcessingInstruction node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateProcessingInstruction(Target, Data)

IDLffXMLDOMDocument::CreateTextNode - Creates and fills a text node owned by the XML document.

Result = Obj->[IDLffXMLDOMDocument::]CreateTextNode(String)

IDLffXMLDOMDocument::CreateTreeWalker - Creates an instance of an IDLffXMLDOMTreeWalker object.

Result = Obj->[IDLffXMLDOMDocument::]CreateTreeWalker(RootNode [, FILTER_NAME=string] [, FILTER_USERDATA=variable] [, WHAT_TO_SHOW=value])

IDLffXMLDOMDocument::GetDoctype - Creates an instance of IDLffXMLDOMDocumentType.

Result = Obj->[IDLffXMLDOMDocument::]GetDoctype()

IDLffXMLDOMDocument::GetDocumentElement - Creates an instance of IDLffXMLDOMElement.

Result = Obj->[IDLffXMLDOMDocument::]GetDocumentElement()

IDLffXMLDOMDocument::GetElementsByTagName - Creates an IDLffXMLDOMNodeList object containing all element nodes in the XML document with the specified tag name.

Result = Obj->[IDLffXMLDOMDocument::]GetElementsByTagName(Tagname)

IDLffXMLDOMDocument::Init - Initializes the object.


or


(Only in a subclass’s Init method)

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.

IDLffXMLDOMDocument::GetEntities - References a DocumentType 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.

IDLffXMLDOMDocument::Save - Serializes the current DOM document and writes it to an output source.

IDLffXMLDOMDocumentType::GetEntities - References the external and internal entities declared in the DTD.

IDLffXMLDOMDocumentType::GetNotations - Returns the notations declared in the DTD.

IDLffXMLDOMElement::GetAttribute - Returns the value of the named attribute.

IDLffXMLDOMElement::GetAttributeNode - Creates a named IDLffXMLDOMAttr object.

IDLffXMLDOMElement::GetElementsByTagName - Creates an IDLffXMLDOMNodeList object containing all element nodes in the XML document with the specified tag name.

IDLffXMLDOMElement::GetTagNode - Returns the element's name.

IDLffXMLDOMElement::GetElementsByTagName - References a DocumentType 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.

IDLffXMLDOMEntity::GetPublicId - Returns the entity's public ID if specified.

IDLffXMLDOMEntity::GetSystemId - Returns the entity's system ID if specified.

IDLffXMLDOMEntityReference::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.

IDLffXMLDOMNamedNodeMap::GetLength - Returns the number of nodes in the named node map.
IDLffXMLDOMNode::GetNodeValue - Returns the node’s value, depending on the subclass type.

Result = Obj->[IDLffXMLDOMNode::]GetNodeValue()

IDLffXMLDOMNode::GetOwnerDocument - Returns an object reference to the IDLffXML DOMDocument used to create the node.

Result = Obj->[IDLffXMLDOMNode::]GetOwnerDocument()

IDLffXMLDOMNode::GetParentNode - Creates an IDLffXML DOM node that refers to the parent in the DOM tree.

Result = Obj->[IDLffXMLDOMNode::]GetParentNode()

IDLffXMLDOMNode::GetPreviousSibling - Creates an IDLffXML DOM node that refers to the previous sibling in the DOM tree.

Result = Obj->[IDLffXMLDOMNode::]GetPreviousSibling()

IDLffXMLDOMNode::HasChildNodes - Returns a value indicating whether the calling node has children.

Result = Obj->[IDLffXMLDOMNode::]HasChildNodes()

IDLffXMLDOMNode::InsertBefore - Inserts a new node into the calling node’s children before a reference node.

Result = Obj->[IDLffXMLDOMNode::]InsertBefore(NewChild, RefChild())

IDLffXMLDOMNode::RemoveChild - Removes an existing child node from the calling node’s child list.

Result = Obj->[IDLffXMLDOMNode::]RemoveChild(OldChild)

IDLffXMLDOMNode::ReplaceChild - Replaces an existing child node with a new node in the calling node’s child list.

Result = Obj->[IDLffXMLDOMNode::]ReplaceChild(NewChild, OldChild)

IDLffXMLDOMNode::SetNodeValue - Sets the node’s value to the contents of a string.

Result = Obj->[IDLffXMLDOMNode::]SetNodeValue(NodeValue)

IDLffXMLDOMNode::NodeIterator - Allows iterative navigation of an IDLffXML DOM tree. No superclasses.

IDLffXMLDOMNode::Node::NextNode - Returns an object reference to the DOM tree node that appears after the iterator’s current position in document-order traversal.

Result = Obj->[IDLffXMLDOMNode::]NextNode()

IDLffXMLDOMNode::Node::PreviousNode - Returns an object reference to the DOM tree node that appears before the iterator’s current position in document-order traversal.

Result = Obj->[IDLffXMLDOMNode::]PreviousNode()
Alphabetical List of IDL Routines

IDLffXMLDOMNodeList - Container for nodes. No superclasses.
IDLffXMLDOMNodeList::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.

IDLffXMLDOMNodeList::GetLength - Returns the number of nodes in the node list.
Result = Obj->[IDLffXMLDOMNodeList::GetLength()]

IDLffXMLDOMNodeList::GetItem - Returns an object reference to the node that contains the indexed item.
Result = Obj->[IDLffXMLDOMNodeList::GetItem(Index)]

IDLffXMLDOMNode - References a node in the XML document. Subclass of IDLffXMLDOMNode.
IDLffXMLDOMNode::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.

IDLffXMLDOMNode::GetPublicID - Returns the notation’s public ID.
Result = Obj->[IDLffXMLDOMNode::GetPublicID()]

IDLffXMLDOMNode::GetSystemID - Returns the notation’s system ID.
Result = Obj->[IDLffXMLDOMNode::GetSystemID()]

IDLffXMLDOMNotation - References a notation in the DTD. Subclass of IDLffXMLDOMNode.
IDLffXMLDOMNotation::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 - References a processing instruction node in the XML document. Subclass of IDLffXMLDOMNode.
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::GetTarget - Returns the target of the processing instruction.
Result = Obj->[IDLffXMLDOMText::GetTarget()]

IDLffXMLDOMText::GetData - Returns the content of the processing instruction.
Result = Obj->[IDLffXMLDOMText::GetData()]

IDLffXMLDOMText::IsIgnorableWhitespace - Indicates whether the text node contains ignorable whitespace.
Result = Obj->[IDLffXMLDOMText::IsIgnorableWhitespace()]

IDLffXMLDOMTreeWalker - Allows tree-walking navigation of an IDLffXMLDOM tree. No superclasses.
IDLffXMLDOMTreeWalker::Cleanup - Destroys both the node-iterator object and any objects created by that object; does not modify the actual DOM tree. Should not be subclassed.

IDLffXMLDOMTreeWalker::FirstChild - Returns an object reference to the DOM tree node that is the first child of the node to which the walker is currently pointing.
Result = Obj->[IDLffXMLDOMTreeWalker::FirstChild()]

IDLffXMLDOMTreeWalker::GetCurrentNode - Returns an object reference to the DOM tree node to which the walker is currently pointing.
Result = Obj->[IDLffXMLDOMTreeWalker::GetCurrentNode()]

IDLffXMLDOMTreeWalker::LastChild - Returns an object reference to the DOM tree node that is the last child of the node to which the walker is currently pointing.
Result = Obj->[IDLffXMLDOMTreeWalker::LastChild()]

IDLffXMLDOMTreeWalker::NextNode - Returns an object reference to the DOM tree node visited next after the walker’s current node in document-order traversal.
Result = Obj->[IDLffXMLDOMTreeWalker::NextNode()]

IDLffXMLDOMTreeWalker::NextSibling - Returns an object reference to the DOM tree node that is the next sibling of the node to which the walker is currently pointing.
Result = Obj->[IDLffXMLDOMTreeWalker::NextSibling()]

IDLffXMLDOMTreeWalker::ParentNode - Returns an object reference to the DOM tree node that is the parent of the node to which the walker is currently pointing.
Result = Obj->[IDLffXMLDOMTreeWalker::ParentNode()]

IDLffXMLDOMTreeWalker::PreviousNode - Returns an object reference to the DOM tree node visited before the walker’s current node in document-order traversal.
Result = Obj->[IDLffXMLDOMTreeWalker::PreviousNode()]

IDLffXMLDOMTreeWalker::PreviousSibling - Returns an object reference to the DOM tree node that is the previous sibling of the node to which the walker is currently pointing.
Result = Obj->[IDLffXMLDOMTreeWalker::PreviousSibling()]

IDLffXMLDOMNodeWalker::SetCurrentNode - Sets the walker’s current node to the specified node.
Obj->[IDLffXMLDOMNodeWalker::SetCurrentNode(Current Node)]
IDLffXMLSAX - Represents an XML SAX Level 2 parser. No super-classes. In order to use this class, you must write your own subclass.

Properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ , FILENAME{Get}=variable ]</td>
<td></td>
</tr>
<tr>
<td>[ , NAMESPACE_PREFIXES{Get, Init, Set} ]</td>
<td></td>
</tr>
<tr>
<td>[ , PARSER_LOCATION{Get}=variable ]</td>
<td></td>
</tr>
<tr>
<td>[ , PARSER_PUBLICID{Get}=variable ]</td>
<td></td>
</tr>
<tr>
<td>[ , SCHEMA_CHECKING{Get, Init, Set}]=[0,1,2]</td>
<td></td>
</tr>
<tr>
<td>[ , VALIDATION_MODE{Get, Init, Set}]=[0,1,2]</td>
<td></td>
</tr>
</tbody>
</table>

IDLffXMLSAX::AttributeDecl - Called when the parser detects an <!ATTLIST ... > declaration in a DTD.

IDLffXMLSAX::AttributeDecl( eName, aName, Type, Mode, Value )

IDLffXMLSAX::Attributes - Called when the parser detects an <!ENTITY ... > declaration in a DTD.

IDLffXMLSAX::Attributes( eName, aName, URI, SystemID, PublicID, Mode, value )

IDLffXMLSAX::BeginCDATA - Called when the parser detects the beginning of a <!-- ... --> text section.

IDLffXMLSAX::BeginCDATA() or IDLffXMLSAX::BeginCDATA(Chars)

IDLffXMLSAX::BeginDocument - Called when the parser begins processing a document, and before any data is processed.

IDLffXMLSAX::BeginDocument()

IDLffXMLSAX::BeginDTD - Called when the parser detects the end of a Document Type Definition (DTD).

IDLffXMLSAX::BeginDTD() or IDLffXMLSAX::BeginDTD( Name, Model )

IDLffXMLSAX::BeginElement - Called when the parser detects the end of an element.

IDLffXMLSAX::BeginElement( Name, URI, Local, qName )

IDLffXMLSAX::BeginEntity - Called when the parser detects the end of an internal or external entity expansion.

IDLffXMLSAX::BeginEntity( Name )

IDLffXMLSAX::BeginPrefixMapping - Called when a previously declared prefix mapping goes out of scope.

IDLffXMLSAX::BeginPrefixMapping( Prefix )

IDLffXMLSAX::Cleanup - Performs all cleanup on the object.

IDLffXMLSAX::Cleanup()

IDLffXMLSAX::Comment - Called when the parser detects a comment section of the form <!- - ... -->.

IDLffXMLSAX::Comment() or IDLffXMLSAX::Comment(Chars)

IDLffXMLSAX::ElementDecl - Called when the parser detects an <!ELEMENT ... > declaration in the DTD.

IDLffXMLSAX::ElementDecl( Name, Model )

IDLffXMLSAX::EndCDATA - Called when the parser detects the end of a <!--CDATA[...]--> text section.

IDLffXMLSAX::EndCDATA()

IDLffXMLSAX::EndElement - Called when the parser detects the end of an element.

IDLffXMLSAX::EndElement( URI, Local, qName )

IDLffXMLSAX::EndEntity - Called when the parser detects the end of an internal or external entity expansion.

IDLffXMLSAX::EndEntity( Name )

IDLffXMLSAX::EndPrefixMapping - Called when a previously declared prefix mapping goes out of scope.

IDLffXMLSAX::EndPrefixMapping( Prefix )

IDLffXMLSAX::Error procedure - Called when the parser detects and error that is not expected to be fatal.

IDLffXMLSAX::Error( SystemID, LineNumber, ColumnNumber, Message )

IDLffXMLSAX::FatalError - Called when the parser detects a fatal error.

IDLffXMLSAX::FatalError( SystemID, LineNumber, ColumnNumber, Message )

IDLffXMLSAX::GetProperty - Used to get the values of various properties of the parser.

IDLffXMLSAX::GetProperty( PROPERTY=variable )

IDLffXMLSAX::IgnoreableWhitespace - Called when the parser detects whitespace that separates elements in an element content model.

IDLffXMLSAX::IgnoreableWhitespace( Chars )

IDLffXMLSAX::Init - Initializes an XML parser object.

IDLffXMLSAX::Init( PROPERTY=value ) or Result = Obj->IDLffXMLSAX::Init( PROPERTY=value )

IDLffXMLSAX::InternalEntityDecl - Called when the parser detects an <!ENTITY ... > declaration in a DTD for (parsed) internal entities.

IDLffXMLSAX::InternalEntityDecl( Name, Value )

IDLffXMLSAX::NotationDecl - Called when the parser detects a <!NOTATION ... > declaration in a DTD.

IDLffXMLSAX::NotationDecl( Name, Model )

IDLffXMLSAX::ParseFile - Parses the specified XML file.

IDLffXMLSAX::ParseFile( InPut [/URL] [/XML_STRING] )

IDLffXMLSAX::ProcessingInstruction - Called when the parser detects a processing instruction.

IDLffXMLSAX::ProcessingInstruction( Target, Data )

IDLffXMLSAX::setProperty - Used to set the values of various properties of the parser.

IDLffXMLSAX::setProperty( PROPERTY=value )

IDLffXMLSAX::SkippedEntity - Called when the parser skips an entity and validation is not being performed.

IDLffXMLSAX::SkippedEntity( Name )

IDLffXMLSAX::StartCDATA - Called when the parser detects the beginning of a <!--CDATA[...]--> text section.

IDLffXMLSAX::StartCDATA()

IDLffXMLSAX::StartDocument - Called when the parser begins processing a document, and before any data is processed.

IDLffXMLSAX::StartDocument()
### Alphabetical List of IDL Routines

**IDLffXMLSAX::StartElement** - Called when the parser detects the beginning of an element.

```
Obj->IDLffXMLSAX::StartElement, URI, Local, qName, attName, attValue
```

**IDLffXMLSAX::StartEntity** - Called when the parser detects the start of an internal or external entity expansion.

```
Obj->IDLffXMLSAX::StartEntity, SystemID, PublicID, 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, Dec, 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
  - **ALPHA_CHANNEL**: [Get, Init, Set] = value
  - **AM_PM**: [Get, Init, Set] = array
  - **CLIP_PLANES**: [Get, Init, Set] = array
  - **COLOR**: [Get, Init, Set] = index or RGB_vector
  - **CRANGE**: [Get] = variable
    - **DAYS_OF_WEEK**: [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]
  - **DIRECTION**: [Get, Init, Set] = [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8]
  - **EXACT**: [Get, Init, Set] = [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8]
  - **GRIDSTYLE**: [Get, Init, Set] = integer
    - **GRIDSTYLE** = [get, init, set] = [integer] = [0 to 6]
  - **LOCATION**: [Get, Init, Set] = [x, y, z]
  - **LOG**: [Get, Init, Set] = [integer]
  - **MINOR**: [Get, Init, Set] = [integer]
  - **MONTHS**: [Get, Init, Set] = [integer]
  - **NOTEXT**: [Get, Init, Set] = [integer]
  - **PARENT**: [Get] = variable
  - **RANGE**: [Get, Init, Set] = [min, max]
  - **REGISTER_PROPERTIES**: [Get, Init, Set] = [integer]
  - **SHADER**: [Get, Init, Set] = [integer]
  - **SUBTICKLEN**: [Get, Init, Set] = value
  - **TEXTALIGNMENTS**: [Get, Init, Set] = [horizontal
    - **TEXTBASELINE**: [Get, Init, Set] = [integer]
    - **TEXTPOS**: [Get, Init, Set] = [0 | 1]
    - **TEXTUPDIR**: [Get, Init, Set] = [vector]
    - **THICK**: [Get, Init, Set] = [point]
    - **TICKDIR**: [Get, Init, Set] = [vector]
    - **TICKFORMAT**: [Get, Init, Set] = [string or array of strings]

**IDLgrProperties** - continued

- **TICKFRMRTDATA**: [Get, Init, Set] = [value]
  - **TICKINTERVAL**: [Get, Init, Set] = [value]
  - **TICKLAYOUT**: [Get, Init, Set] = [scalar]
  - **TICKLEN**: [Get, Init, Set] = [value]
  - **TICKTEXT**: [Get, Init, Set] = [object or vector]
  - **TICKUNITS**: [Get, Init, Set] = [string or a vector of strings]
  - **TICKVALUES**: [Get, Init, Set] = [vector]
  - **TITLE**: [Get, Init, Set] = [object]
    - **USE_TEXT_COLOR**: [Get, Init, Set] = [object]
    - **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]

**IDLgrAxis::Cleanup** - Performs all cleanup on the object.

```
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(), [PATH=object(s)], [TOP=object(s)]
```

**IDLgrAxis::GetProperty** - Retrieves the value of a property or group of properties for the axis.

```
Object->IDLgrAxis::GetProperty, [PROPERTY=variable]
```

**IDLgrAxis::Init** - Initializes an axis object.

```
Obj = Obj->IDLgrAxis::Init(), [Direction], [PROPERTY=value] or
Result = Obj->IDLgrAxis::Init([Direction], [PROPERTY=value])
```

**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] = [object]
  - **DIMENSIONS**: [Get, Init, Set] = [width, height]
  - **GRAPHICS_TREE**: [Get, Init, Set] = [object]
  - **IMAGE_DATA**: [Get] = object
    - **N_COLORS**: [Get, Init] = [integer]
    - **QUALITY**: [Get, Init, Set] = [0 | 1 | 2]
    - **REGISTER_PROPERTIES**: [Get, Init, Set]
  - **RESOLUTION**: [Get, Init] = [object]
  - **SCREEN_DIMENSIONS**: [Get] = [object]
  - **UNITS**: [Get, Init] = [0 | 1 | 2 | 3]
  - **ZBUFFER_DATA**: [Get] = [variable]
IDLgrBuffer::Cleanup - Performs all cleanup on the object.

IDLgrBuffer::GetFontnames - Returns the list of available fonts that can be used in IDLgrFont objects.

IDLgrBuffer::GetProperty - Returns the value of a property or group of properties for the buffer.

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).

IDLgrBuffer::Init - Initializes the buffer object.

IDLgrBuffer::Read - Reads an image from a buffer.

IDLgrBuffer::Select - Returns a list of objects selected at a specified location.

IDLgrBuffer::Draw - Draws picture to this graphics destination.

IDLgrBuffer::Erase - Erases this graphics destination.

IDLgrBuffer::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.

IDLgrBuffer::GetTextDimensions - Retrieves the dimensions of a text object that will be rendered in the buffer.

IDLgrBuffer::PickData - Maps a point in the 2D device space of the buffer to a point in the 3D data space of an object tree.

IDLgrBuffer::QueryRequiredTiles - Returns an array of named structures containing information regarding which tile data is needed for display. Used with a tiled IDLgrImage object.

IDLgrBuffer::Read - Reads an image from a buffer.

IDLgrBuffer::SetProperty - Sets the value of a property or group of properties for the buffer.

IDLgrClipboard - A destination object representing the native clipboard.

IDLgrClipboard::Cleanup - Performs all cleanup on the object.

IDLgrClipboard::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.

IDLgrClipboard::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.

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).

IDLgrClipboard::Select - Returns a list of objects selected at a specified location.

IDLgrClipboard::Draw - Draws a picture to a graphics destination.

IDLgrClipboard::Cleanup - Performs all cleanup on the object.

IDLgrClipboard::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.

IDLgrClipboard::GetDeviceInfo - Returns information that allows IDL applications to make decisions for optimal performance.

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).

IDLgrClipboard::Select - Returns a list of objects selected at a specified location.
Alphabetical List of IDL Routines

IDLgrClipboard::GetProperty - Retrieves the value of a property or group of properties for the clipboard buffer.

Obj->IDLgrClipboard::GetProperty
{ PROPERTY=variable }

IDLgrClipboard::GetTextDimensions - Retrieves the dimensions of a text object that will be rendered in the clipboard buffer.

Result = Obj->IDLgrClipboard::GetTextDimensions( TextObj [, DESCENT=variable, PATH=objref(s) ] )

IDLgrClipboard::Init - Initializes the clipboard object.

Obj = OBJ_NEW('IDLgrClipboard', [ PROPERTY=value ])
or Result = Obj->IDLgrClipboard::Init( [PROPERTY=value] )

IDLgrClipboard::QueryRequiredTiles - Returns an array of named structures containing information regarding which tile data is needed for display. Used with a tiled IDLgrImage object.

Result = Obj->IDLgrClipboard::QueryRequiredTiles( View, Image [, COUNT=variable ] [, ALL_VISIBLE=variable ] )

IDLgrClipboard::SetProperty - Sets the value of a property or group of properties for the clipboard buffer.

Obj->IDLgrClipboard::SetProperty
{ PROPERTY=value }

IDLgrColorbar - Consists of a color-ramp with an optional framing box and annotation axis.

Properties: [ , ALL[Get]=variable ]
{ BLUE_VALUES[Get, Init, Set]=vector }
{ COLOR[Get, Init, Set]=index or RGB vector }
{ DIMENSIONS[Get, Init, Set]=[dx, dy] }
{ GREEN_VALUES[Get, Init, Set]=vector }
{ HIDE[Get, Init, Set]=integer }, { MAJOR[Get, Init, Set]=integer }
{ PALETTE[Get, Init, Set]=objref }
{ PARENT[Get]=variable }, { RED_VALUES[Get, Init, Set]=vector }
{ SHOW_AXIS[Get, Init, Set]=[0 | 1 | 2] }
{ SHOW_OUTLINE[Get, Init, Set]=variable }
{ SUBTICKLEN[Get, Init, Set]=minorTickLength]/majorTickLength }
{ THICK[Get, Init, Set]=points[1.0 to 10.0] }, { THREED[Get, Init]=variable }
{ TICKFORMAT[Get, Init, Set]=string }
{ TICKFRMTDATA[Get, Init, Set]=value }
{ TICKLEN[Get, Init, Set]=value }, { TICKTEXT[Get, Init, Set]=objref(s) }
{ TICKVALUES[Get, Init, Set]=vector }
{ TITLE[Get, Init, Set]=objref }
{ XCOORD_CONV[Get, Init, Set]=vector }
{ XRANGE[Get]=variable }, { YCOORD_CONV[Get, Init, Set]=vector }
{ YRANGE[Get]=variable }

IDLgrColorbar::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY,Obj or Obj->IDLgrColorbar::Cleanup

IDLgrColorbar::ComputeDimensions - Retrieves the dimensions of a colorbar object for the given destination object.

Result = Obj->IDLgrColorbar::ComputeDimensions( DestinationObj [, PATH=objref(s) ] )

IDLgrColorbar::GetProperty - Retrieves the value of a property or group of properties for the colorbar.

Obj->IDLgrColorbar::GetProperty
{ PROPERTY=variable }

IDLgrColorbar::Init - Initializes the colorbar object.

Obj = OBJ_NEW('IDLgrColorbar', [ aRed, aGreen, aBlue ] [, PROPERTY=value ])
or Result = Obj->IDLgrColorbar::Init([aRed, aGreen, aBlue ] [, PROPERTY=value] )

IDLgrColorbar::SetProperty - Sets the value of a property or group of properties for the colorbar.

Obj->IDLgrColorbar::SetProperty
{ PROPERTY=value }

IDLgrContour - Draws a contour plot from an array or unstructured point data. No superclasses. No subclasses.

Properties: [ , ALL[Get]=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_NOGAPS[Get, Init, Set]=vector }, { C_LABEL_OBJECTS[Get, Init, Set]=array of object references }
{ C_LABEL_SHOW[Get, Init, Set]=vector of integers }[, { C_LINESTYLE[Get, Init, Set]=array 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 }]
{ DATA_VALUES[Get, Init, Set]=vector or 2D array }[, { DAYS_OF_WEEK[Get, Init, Set]=vector of seven strings }]
{ DEPTH_OFFSET[Get, Init, Set]=value }[, { DEPTH_TEST_DISABLE[Get, Init, Set]=value }]
{ DEPTH_TEST_FUNCTION[Get, Init, Set]=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 ][, { DEPTH_WRITE_DISABLE[Get, Init, Set]=value }]
{ /DOUBLE_GEM[Init]=1 }, { /DOUBLE_DATA[Init] }[, { /DOWNHILL[Get, Init, Set]=1 }
{ /FILL[Get, Init, Set]=1 }[, { GEOM[Get]=variable }]
{ GEOMX[Init, Set]=vector or 2D array }[, { GEOMY[Init, Set]=vector or 2D array }]
{ GEOMZ[Init, Set]=scalar, vector, or 2D array }[, { HIDE[Get, Init, Set]=1 }]
{ LABEL_FORMAT[Get, Init, Set]=string }[, { LABEL_FRMTDATA[Init]=value }]
{ LABEL_UNITS[Get, Init, Set]=string }[, { MAX_VALUE[Get, Init, Set]=value }]

IDL Quick Reference

IDLgrContour::GetProperty - Retrieves the value of a property or group of properties for the contour object.
Obj->IDLgrContour::GetProperty [. PROPERTY=variable]

IDLgrContour::Init - Initializes the contour object.
Obj = OBJ_NEW(IDLgrContour') [. PROPERTY=variable] or
Result = Obj->IDLgrContour::Init [. PROPERTY=variable]

IDLgrContour::SetProperty - Sets the value of a property or group of properties for the contour object.
Obj->IDLgrContour::SetProperty [. PROPERTY=variable]

IDLgrContour::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrContour::Cleanup

IDLgrContour::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object
Result = Obj->IDLgrContour::GetCTM () . DESTINATION=objref [. PATH=objref(s)] [. TOP= wilderness]

IDLgrContour::GetLabelInfo - Retrieves information about the labels for a contour
Obj->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=variable] or
Result = Obj->IDLgrContour::Init [. VALUES=variable]

IDLgrContour::SetProperty - Sets the value of a property or group of properties for the contour.
Obj->IDLgrContour::SetProperty [. PROPERTY=variable]

IDLgrFilterChain - Provides a way to apply a sequence of image filtering shader programs in the order that they are added to this specialized container object (if there is sufficient graphics hardware support).
Properties: [. FORCE_FILTER=Get, Init]

IDLgrFilterChain::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrFilterChain::Cleanup

IDLgrFilterChain::GetProperty - Retrieves the value of a property or group of properties for the filter chain object.
Obj->IDLgrFilterChain::GetProperty [. PROPERTY=variable]

IDLgrFilterChain::Init - Initializes the filter chain object.
Obj = OBJ_NEW(IDLgrFilterChain') [. PROPERTY=variable] or
Result = Obj->IDLgrFilterChain::Init [. PROPERTY=variable]

IDLgrFilterChain::SetProperty - Sets the value of a property or group of properties for the filter chain object.
Obj->IDLgrFilterChain::SetProperty [. PROPERTY=variable]

IDLgrFont - Represents a typeface, style, weight, and point size that may be associated with text objects.
Properties: [. ALL=variable] [. SIZE=Get, Init, Set=points] [. SUBSTITUTE=Get, Init, Set=('Helvetica' | 'Courier' | 'Times' | 'Symbol' | 'Hershey')] [. THICK=Get, Init, Set=points(1.0 to 10.0)] [. VALUE=Get, Init, Set=variable]

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.
Properties: [. ALL=vector] [. ALPHA_CHANNEL=variable] [. BLEND_FUNCTION=Get, Init, Set=vector] [. CHANNEL=Get, Init, Set=hexadecimal bitmap] [. CLIP_PLANES=Get, Init, Set=array] [. DATA=Get, Init, Set=nxm, 2nxm, 3nxm, or 4nxm array of image data] [. DEPTH_OFFSET=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]] [. DEPTH_WRITE_DISABLE=Get, Init, Set=[0 | 1 | 2]] [. DIMENSIONS=Get, Init, Set=[width, height]] [. GRAYSCALE=Get, Init, Set=variable] [. IMAGE_ID=Get, Init, Set=variable] [. INTERLEAVE=Get, Init, Set=[0 | 1 | 2]]
### IDL Quick Reference

- **IDLgrImage::SetProperty** - Sets the value of the property or group of properties for the image.
  
  \[ \text{Obj} \rightarrow \text{IDLgrImage::SetProperty}, \text{PROPERTY} = \text{value} \]
  
  - \[ \text{/NO\_COPY} \]

- **IDLgrImage::ReadFilteredData** - Returns the image data after the application of one or more IDLgrShader programs to the image.
  
  \[ \text{data} = \text{Obj} \rightarrow \text{IDLgrImage::ReadFilteredData}, \text{Dest} \]
  
  - \[ \text{DATA\_TYPE} = \text{value} \]
  
  - \[ \text{READ\_POSITION} = \text{value} \]
  
  - \[ \text{TileInfo} = \text{tileinf} \]

- **IDLgrImage::SetTileData** - Stores tile data in the tile data cache.
  
  \[ \text{Obj} \rightarrow \text{IDLgrImage::SetTileData}, \text{TileInfo}, \text{TileData} \]
  
  - \[ \text{NO\_FREE} = \text{value} \]

- **IDLgrImage::TileDataLoaded** - Queries an image tile to determine whether data is loaded.
  
  \[ \text{Result} = \text{Obj} \rightarrow \text{IDLgrImage::TileData}, \text{TileInfo} \]

- **IDLgrLegend::Init** - Initializes the legend object.
  
  \[ \text{Obj} \rightarrow \text{IDLgrLegend::Init}, \text{aItemNames} \]
  
  - \[ \text{PROPERTY} = \text{value} \]

- **IDLgrLegend::GetProperty** - Retrieves the value of the property or group of properties for the legend.
  
  \[ \text{Obj} \rightarrow \text{IDLgrLegend::GetProperty}, \text{PROPERTY} = \text{value} \]
  
  - \[ \text{/NO\_COPY} \]

- **IDLgrLegend::ComputeDimensions** - Retrieves the dimensions of a legend object for the given destination object.
  
  \[ \text{Result} = \text{Obj} \rightarrow \text{IDLgrLegend::ComputeDimensions}, \text{DestinationObj}, \text{PATH} = \text{objref(s)} \]

- **IDLgrLegend::SetProperty** - Sets the value of the property or group of properties for the legend.
  
  \[ \text{Obj} \rightarrow \text{IDLgrLegend::SetProperty}, \text{PROPERTY} = \text{value} \]
  
  - \[ \text{/NO\_COPY} \]

### Alphabetic List of IDL Routines

- **IDL Quick Reference**

  - **IDLgrImage::SetProperty** - Sets the value of the property or group of properties for the image.
    
    \[ \text{Obj} \rightarrow \text{IDLgrImage::SetProperty}, \text{PROPERTY} = \text{value} \]
    
    - \[ \text{/NO\_COPY} \]

  - **IDLgrImage::TileDataLoaded** - Queries an image tile to determine whether data is loaded.
    
    \[ \text{Result} = \text{Obj} \rightarrow \text{IDLgrImage::TileData}, \text{TileInfo} \]

  - **IDLgrLegend::Init** - Initializes the legend object.
    
    \[ \text{Obj} \rightarrow \text{IDLgrLegend::Init}, \text{aItemNames} \]
    
    - \[ \text{PROPERTY} = \text{value} \]

  - **IDLgrLegend::GetProperty** - Retrieves the value of the property or group of properties for the legend.
    
    \[ \text{Obj} \rightarrow \text{IDLgrLegend::GetProperty}, \text{PROPERTY} = \text{value} \]
    
    - \[ \text{/NO\_COPY} \]

  - **IDLgrLegend::ComputeDimensions** - Retrieves the dimensions of a legend object for the given destination object.
    
    \[ \text{Result} = \text{Obj} \rightarrow \text{IDLgrLegend::ComputeDimensions}, \text{DestinationObj}, \text{PATH} = \text{objref(s)} \]

  - **IDLgrLegend::SetProperty** - Sets the value of the property or group of properties for the legend.
    
    \[ \text{Obj} \rightarrow \text{IDLgrLegend::SetProperty}, \text{PROPERTY} = \text{value} \]
    
    - \[ \text{/NO\_COPY} \]

### Alphabetical List of IDL Routines

- **IDLgrImage::SetProperty** - Sets the value of the property or group of properties for the image.
  
  \[ \text{Obj} \rightarrow \text{IDLgrImage::SetProperty}, \text{PROPERTY} = \text{value} \]
  
  - \[ \text{/NO\_COPY} \]

- **IDLgrImage::TileDataLoaded** - Queries an image tile to determine whether data is loaded.
  
  \[ \text{Result} = \text{Obj} \rightarrow \text{IDLgrImage::TileData}, \text{TileInfo} \]

- **IDLgrLegend::Init** - Initializes the legend object.
  
  \[ \text{Obj} \rightarrow \text{IDLgrLegend::Init}, \text{aItemNames} \]
  
  - \[ \text{PROPERTY} = \text{value} \]

- **IDLgrLegend::GetProperty** - Retrieves the value of the property or group of properties for the legend.
  
  \[ \text{Obj} \rightarrow \text{IDLgrLegend::GetProperty}, \text{PROPERTY} = \text{value} \]
  
  - \[ \text{/NO\_COPY} \]
IDLgrLegend::SetProperty - Sets the value of a property or group of properties for the legend.

IDLgrLight::Cleanup - Performs all cleanup on the object.

IDLgrLight::GetProperty - Returns the 4 x 4 graphics transform matrix from the current object.

IDLgrModel::Add - Adds a child to this Model.

IDLgrModel::Cleanup - Performs all cleanup on the object.

IDLgrModel::Construct - Creates an object.

IDLgrModel::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object.

IDLgrModel::GetProperty - Retrieves the value of a property or group of properties for the model.

IDLgrModel::Translate - Translates model by specified offsets.

IDLgrModel::SetProperty - Sets the value of a property or group of properties for the light.

IDLgrModel::Scale - Scales model by the specified scaling factors.

IDLgrModel::SetTransform - Sets the current transform matrix for the model object to the identity matrix.

IDLgrModel::Visibility - Controls the visibility of the model.

IDLgrModel::Zoom - Changes the zoom of the model.

IDLgrModel::::Add - Adds a child to this Model.

IDLgrModel::::Construct - Creates an object.

IDLgrModel::::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object.

IDLgrModel::::GetProperty - Retrieves the value of a property or group of properties for the model.

IDLgrModel::::Translate - Translates model by specified offsets.

IDLgrModel::::SetProperty - Sets the value of a property or group of properties for the light.

IDLgrModel::::Scale - Scales model by the specified scaling factors.

IDLgrModel::::SetTransform - Sets the current transform matrix for the model object to the identity matrix.

IDL grModel::::Visibility - Controls the visibility of the model.

IDLgrModel::::Zoom - Changes the zoom of the model.
IDLgrMPEG::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrMPEG::Cleanup

IDLgrMPEG::GetProperty - Retrieves the value of a property or group of properties for the MPEG object.
Obj->IDLgrMPEG::GetProperty
[, PROPERTY=variable]

IDLgrMPEG::Init - Initializes the MPEG object.
Obj = OBJ_NEW(IDLgrMPEG[, PROPERTY=value]) or
Result = Obj->IDLgrMPEG::[Init(PROPERTY=value)]

IDLgrMPEG::Put - Puts a given image into the MPEG sequence at the specified frame.
Obj->IDLgrMPEG::[Put, Image[, Frame]]

IDLgrMPEG::Save - Encodes and saves an MPEG sequence to a file.
Obj->IDLgrMPEG::[Save [, FILENAME=string]]

IDLgrMPEG::SetProperty - Sets the value of a property or group of properties for the MPEG object.
Obj->IDLgrMPEG::[SetProperty[, PROPERTY=value]]

IDLgrPalette - Represents a color lookup table that maps indices to red, green, and blue values.
Properties: [ ALL[Get]=variable]
[ , BLUE_VALUES[Get, Init, Set]=vector]
[ , BOTTOM.Stretch[Get, Init, Set]=value[0 to 100]]
[ , GAMMA[Get, Init, Set]=value[0.1 to 10.0]]
[ , GREEN_VALUES[Get, Init, Set]=vector]
[ , N.COLORS[Get]=variable]
[ , NAME[Get, Init, Set]=string]
[ , RED_VALUES[Get, Init, Set]=vector]
[ , TOP.Stretch[Get, Init, Set]=value[0 to 100]]
[ , UVируется value[0 to 100]]
[ , V.value=value]

IDLgrPalette::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrPalette::Cleanup

IDLgrPalette::GetRGB - Returns the RGB values contained in the palette at the given index.
Result = Obj->IDLgrPalette::GetRGB(Index)

IDLgrPalette::GetProperty - Retrieves the value of a property or group of properties for the palette.
Obj->IDLgrPalette::GetProperty
[, PROPERTY=variable]

IDLgrPalette::Init - Initializes a palette object.
Obj = OBJ_NEW(IDLgrPalette, Red, aGreen, aBlue
[ , PROPERTY=variable]) or
Result = Obj->IDLgrPalette::[Init(aRed, aGreen, aBlue]
[ , PROPERTY=variable])

IDLgrPalette::LoadCT - Loads one of the IDL predefined color tables into an IDL.grPalette object.
Obj->IDLgrPalette::LoadCT, TableNum
[ , FILENAME=colortable filename]

IDLgrPalette::NearestColor - Returns the index of the color in the palette that best matches the given RGB values.
Result = Obj->IDLgrPalette::NearestColor(Red, Green, Blue)

IDLgrPalette::SetRGB - Sets the color values at a specified index in the palette to the specified Red, Green and Blue values.
Obj->IDLgrPalette::SetRGB, Index, Red, Green, Blue

IDLgrPalette::SetProperty - Sets the value of a property or group of properties for the palette.
Obj->IDLgrPalette::[SetProperty[, PROPERTY=value]]

IDLgrPattern - Describes which pixels are filled and which are left blank when an area is filled.
Properties: [, ALL[Get]=variable]
[, NAME[Get, Init, Set]=string]
[, ORIENTATION[Get, Init, Set]=ccw degrees from horizontal]
[, PATTERN[Get, Init, Set]=32 x 32 bit array]
[, SPACING[Get, Init, Set]=points]
[, STYLE[Get, Init, Set]={0 | 1 | 2}]
[, THICK[Init, Set]=points[1.0 to 10.0]]
[, U.VALUE[Get, Init, Set]=value]

IDLgrPattern::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or Obj->IDLgrPattern::Cleanup

IDLgrPattern::GetProperty - Retrieves the value of a property or group of properties for the pattern.
Obj->IDLgrPattern::GetProperty
[ , PROPERTY=value]

IDLgrPattern::Init - Initializes the pattern object.
Obj = OBJ_NEW(IDLgrPattern[, Style]
[ , PROPERTY=value]) or
Result = Obj->IDLgrPattern::[Init[Style]
[ , PROPERTY=value])

IDLgrPattern::SetProperty - Sets the value of a property or group of properties for the pattern.
Obj->IDLgrPattern::[SetProperty[, PROPERTY=value]]

IDLgrPlot - Creates set of polylines connecting data points in 2D space.
Properties: [, ALL[Get]=variable]
[, ALPHA_CHANNEL[Get, Init, Set]=value]
[, CLIP_PLANES[Get, Init, Set]=array]
[, COLOR[Get, Init, Set]=index or RGB vector]
[, FONT_COLORS[Get, Init, Set]=vector]
[, DATA[Get]=variable]
[, DATAX[Init, Set]=vector]
[, DATAY[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}]
[, DEPTH.WRITE_DISABLE[Get, Init, Set]={0 | 1 | 2}]
[, DOUBLE[Get, Init, Set]=variable]
[, /HIDE[Get, Init, Set]]
[, /HISTOGRAM[Get, Set]]
[, LINESS[Get, Init, Set]=integer or two-element vector]
[, MAX.VALUE[Get, Init, Set]=value]
[, MIN.VALUE[Get, Init, Set]=value]
[, NSUM[Get, Init, Set]=value]
[, PALETTE[Get, Init, Set]=objref]
[, PARENT[Get]=variable]
[, /POLAR[Get, Init, Set]]
[, /REGISTER_PROPERTIES[Get, Init, Set]]
[, /RESET_DATA[Get, Init, Set]]
[, SHADER[Get, Init, Set]}

IDL Quick Reference

IDLgrPlot
IDLgrPlot::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or 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=objcies(s) ] [ , TOP=objcie to IDLgrModel object ] )

IDLgrPlot::GetProperty - Retrieves the value of the property or group of properties for the plot.
Obj->IDLgrPlot::GetProperty[ , PROPERTY=variable]

IDLgrPlot::GetVertexAttributeData - Retrieves the value of the attribute variable specified in the Name argument (a case-sensitive string).
Obj->IDLgrPlot::GetVertexAttributeData[ , Name, Value]

IDLgrPlot::Init - Initializes the plot object.
Obj = OBJ_NEW(‘IDLgrPlot’ [, X, Y] [ , PROPERTY=value] ) or
Result = Obj->IDLgrPlot::Ini(([X] Y) [ , PROPERTY=value] )

IDLgrPlot::SetProperty - Sets the value of the property or group of properties for the plot.
Obj->IDLgrPlot::SetProperty[ , PROPERTY=value]

IDLgrPlot::SetVertexAttributeData - Sets the value of the vertex attribute data variable specified in the Name argument (a case-sensitive string).
Obj->IDLgrPlot::SetVertexAttributeData[ , Name, Value]

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=vector]
[ , 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] ]
[ , DIFFUSE=Get, Init, Set=RGB vector] [ , DOUBLE=Get, Init, Set=value]
[ , EMMISION=Get, Init, Set=RGB vector]

IDLgrPolygon::Cleanup - Performs all cleanup on the object.
OBJ_DESTROY, Obj or 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=objcie [ , PATH=objcies(s) ] [ , TOP=objcie to IDLgrModel object ] )

IDLgrPolygon::GetProperty - Retrieves the value of the property or group of properties for the polygons.
Obj->IDLgrPolygon::GetProperty[ , PROPERTY=variable]

IDLgrPolygon::GetMultiTextureCoord - Retrieves the texture coordinates associated with the specified Unit argument.
Obj->IDLgrPolygon::GetMultiTextureCoord[ , Unit, TexCoord]

IDLgrPolygon::GetVertexAttributeData - Retrieves the value of the attribute variable specified in the Name argument (a case-sensitive string).
Obj->IDLgrPolygon::GetVertexAttributeData[ , Name, Value]

IDLgrPolygon::Init - Initializes the polygons object.
Obj = OBJ_NEW(‘IDLgrPolygon’ [, X, Y, Z]) [ , PROPERTY=value] ) or
Result = Obj->IDLgrPolygon::Ini((X, Y, Z) [ , PROPERTY=value] )
**IDLgrPolygon::SetMultiTextureCoord** - Sets the texture coordinates associated with the specified Unit argument.

```
Obj->[IDLgrPolygon::]SetMultiTextureCoord, Unit, TexCoord
```

**IDLgrPolygon::SetProperty** - Sets the value of the property or group of properties for the polygons.

```
Obj->[IDLgrPolygon::]SetProperty [, PROPERTY=value]
```

**IDLgrPolygon::SetVertexAttributeData** - Sets the value of the vertex attribute data variable specified in the Name argument (a case-sensitive string).

```
Obj->[IDLgrPolygon::]SetVertexAttributeData, Name, Value
```

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

```
Properties: 
[ , ALL[Get]=variable ]
[ , ALPHA_CHANNEL[Get, Init, Set]=value ]
[ , CLIP_PLANES[Get, Init, Set]=array ]
[ , COLOR[Get, Init, Set]=vector ]
[ , DATA[Get, Init, Set]=array ]
[ , DEPTH_TEST_DISABLE[Get, Init, Set]=true ]
[ , DEPTH_TEST_FUNCTION[Get, Init, Set]=0 ]
[ , DEPTH_TEST_DISABLE[Get, Init, Set]=false ]
[ , DOUBLE[Get, Init, Set]=true ]
[ , HIDE[Get, Init, Set]=false ]
[ , LABEL_NOGAPS[Get, Init, Set]=false ]
[ , LABEL_OFFSETS[Get, Init, Set]=vector ]
[ , LABEL_OBJECTS[Get, Init, Set]=objref ]
[ , LABEL_OFFSETS[Get, Init, Set]=vector ]
[ , LABEL_OBJECTS[Get, Init, Set]=objref ]
[ , LABEL_USE_VERTEX_COLOR[Get, Init, Set]=false ]
[ , LINESSTYLE[Get, Init, Set]=vector ]
[ , REGISTER_PROPERTIES[Get, Init, Set]=false ]
[ , RESET_DATA[Get, Init, Set]=false ]
[ , SHADER[Get, Init, Set]=false ]
[ , SHADING[Get, Init, Set]=false ]
[ , SHARE_DATA[Get, Init, Set]=false ]
[ , SYMBOL[Get, Init, Set]=false ]
[ , THICK[Get, Init, Set]=false ]
[ , THICKNESS[Get, Init, Set]=false ]
[ , USE_LABEL_COLOR[Get, Init, Set]=false ]
[ , USE_LABEL_ORIENTATION[Get, Init, Set]=false ]
[ , USE_TEXT_ALIGNMENT[Get, Init, Set]=false ]
[ , XCOORD_CONV[Get, Init, Set]=false ]
[ , YCOORD_CONV[Get, Init, Set]=false ]
[ , XCOORD_CONV[Get, Init, Set]=false ]
[ , YCOORD_CONV[Get, Init, Set]=false ]
[ , XVECTOR[Get]=variable ]
[ , YVECTOR[Get]=variable ]
[ , ZVECTOR[Get]=variable ]
```

**IDLgrPolyline::GetCTM** - Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.

```
Result = Obj->[IDLgrPolyline::]GetCTM( 
[ , DESTINATION=objref ]
[ , PATH=objref ]
[ , TOP=objref ]
)
```

**IDLgrPolyline::GetProperty** - Retrieves the value of a property or group of properties for the polylines.

```
Obj->[IDLgrPolyline::]GetProperty 
[ , PROPERTY=variable ]
```

**IDLgrPolyline::GetVertexAttributeData** - Retrieves the value of the attribute variable specified in the Name argument (a case-sensitive string).

```
Obj->[IDLgrPolyline::]GetVertexAttributeData, Name, Value
```

**IDLgrPrinter** - Represents a hardcopy graphics destination.

```
Properties: 
[ , ALL[Get]=variable ]
[ , COLOR_MODEL[Get, Init, Set]=true ]
[ , COLORS[Get, Init, Set]=vector ]
[ , DIMENSIONS[Get]=array ]
[ , GRAPHICS_TREE[Get, Init, Set]=objref ]
[ , LANDSCAPE[Get, Init, Set]=false ]
[ , N_COLORS[Get, Init, Set]=integer ]
[ , N_COPY[Get, Init, Set]=integer ]
[ , NAME[Get, Init, Set]=string ]
[ , PALETTE[Get, Init, Set]=objref ]
[ , PRINT_QUALITY[Get, Init, Set]=integer ]
[ , REGISTER_PROPERTIES[Get, Init, Set]=false ]
```

**IDLgrPrinter::Cleanup** - Performs all cleanup on the object.

```
OBJDESTROY, Obj or Obj->[IDLgrPrinter::]Cleanup
```

**IDLgrPrinter::Draw** - Draws a picture to this graphics destination.

```
Obj->[IDLgrPrinter::]Draw [, Picture ]
[ , VECTOR_SORTING=0 ]
[ , VECTOR_TEXT_RENDER_METHOD=0 ]
[ , VECTOR=0 ]
```

**IDLgrPolyline::Cleanup** - Performs all cleanup on the object.

```
OBJDESTROY, Obj or Obj->[IDLgrPolyline::]Cleanup
```

**IDLgrPolyline::Init** - Initializes the polylines object.

```
Obj = OBJ_NEW('IDLgrPolyline' [, X [, Y [, Z]]] 
[ , PROPERTY=value ]
```

**IDLgrPolyline::SetProperty** - Sets the value of a property or group of properties for the polylines.

```
Obj->[IDLgrPolyline::]SetProperty [, PROPERTY=variable ]
```

**IDLgrPolyline::SetVertexAttributeData** - Sets the value of the vertex attribute data variable specified in the Name argument (a case-sensitive string).

```
Obj->[IDLgrPolyline::]SetVertexAttributeData, Name, Value
```

**IDLgrScene, IDLgrViewgroup, or IDLgrView**

```
[ , /LANDSCAPE[Get, Init, Set]=false ]
[ , /N_COLORS[Get, Init, Set]=integer ]
[ , /N_COPY[Get, Init, Set]=integer ]
[ , /NAME[Get, Init, Set]=string ]
[ , /PALETTE[Get, Init, Set]=objref ]
[ , /PRINT_QUALITY[Get, Init, Set]=integer ]
[ , /REGISTER_PROPERTIES[Get, Init, Set]=false ]
[ , /RESOLUTION[Get]=variable ]
[ , /UNITS[Get, Init, Set]=true ]
```

**IDLgrPrinter::Cleanup** - Performs all cleanup on the object.

```
OBJDESTROY, Obj or Obj->[IDLgrPrinter::]Cleanup
```

**IDLgrPrinter::Draw** - Draws a picture to this graphics destination.

```
Obj->[IDLgrPrinter::]Draw [, Picture ]
[ , VECTOR_SORTING=0 ]
[ , VECTOR_TEXT_RENDER_METHOD=0 ]
[ , VECTOR=0 ]
```
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).  
Return = Obj->IDLgrPrinter::GetContiguousPixels()

IDLgrPrinter::GetFontnames - Returns the list of available fonts that can be used in IDLgrFont objects.  
Return = Obj->IDLgrPrinter::GetFontnames( FamilyName [, IDL_FONTS=[{0 1 2}] [, STYLE=s|sa|st] )

IDLgrPrinter::GetProperty - Retrieves the value of a property or group of properties for the printer.  
Obj->IDLgrPrinter::GetProperty[ , PROPERTY=value]

IDLgrPrinter::GetTextDimensions - Retrieves the dimensions of a text object that will be rendered on the printer.  
Result = Obj->IDLgrPrinter::GetTextDimensions( TextObj [, DESCENT=value] [, PATH=objref|s] )

IDLgrPrinter::Init - Initializes the printer object.  
Obj = OBJ_NEW('IDLgrPrinter' [, PROPERTY=value]) or Result = Obj->IDLgrPrinter::Init([PROPERTY=value])

IDLgrPrinter::NewDocument - Closes current document (page or group of pages), which causes pending output to be sent to the printer, finishing the printer job.  
Obj->IDLgrPrinter::NewDocument

IDLgrPrinter::NewPage - Issues new page command to printer.  
Obj->IDLgrPrinter::NewPage

IDLgrPrinter::QueryRequiredTiles - Returns an array of named structures containing information regarding which tile data is needed for display. Used with a tiled IDLgrImage object.  
Result = Obj->IDLgrPrinter::QueryRequiredTiles( View, Image [, COUNT=value] [, ALL_VISIBLE=value])

IDLgrPrinter::SetProperty - Sets the value of a property or group of properties for the printer.  
Obj->IDLgrPrinter::SetProperty[ , PROPERTY=value]

IDLgrROI - Object graphics representation of a region of interest.  

IDLgrROI::Cleanup - Performs all cleanup for the object.  
OBJ_DESTROY, Obj or Obj -> IDLgrROI::Cleanup

IDLgrROI::GetProperty - Retrieves the value of a property or group of properties for the Object Graphics region.  
Obj -> IDLgrROI::GetProperty[ , PROPERTY=value]

IDLgrROI::Init - Initializes an Object Graphics region of interest.  
Obj = OBJ_NEW('IDLgrROI' [, X | Y | Z]) [ , PROPERTY=value] or Result = Obj->IDLgrROI::Init([X | Y | Z]) [ , PROPERTY=value]

IDLgrROI::PickVertex - Picks a vertex of the region that, when projected onto the given destination device, is nearest to the given 2D device coordinate.  
Result = Obj->IDLgrROI::PickVertex( Dest, View, Point [, PATH=objref] )

IDLgrROI::SetProperty - Sets the value of a property or group of properties for the Object Graphics region.  
Obj->IDLgrROI::SetProperty[ , PROPERTY=value]

IDLgrROIGroup - Object Graphics representation of a group of regions of interest.  

IDLgrROIGroup::Add - Adds a region to the region group.  
Obj -> IDLgrROIGroup::Add, ROI

IDLgrROIGroup::Cleanup - Performs all cleanup for the object.  
OBJ_DESTROY, Obj or Obj -> IDLgrROIGroup::Cleanup

IDLgrROIGroup::GetProperty - Retrieves the value of a property or group of properties for the region group.  
Obj -> IDLgrROIGroup::GetProperty[ , PROPERTY=value]

IDLgrROIGroup::Init - Initializes an Object Graphics region of interest group object.  
Obj = OBJ_NEW('IDLgrROIGroup' [, PROPERTY=value]) or
**Alphabetical List of IDL Routines**

**IDLgrROIGroup**

- **Add** - Picks a region within the group that, when projected onto the given destination device, is nearest to the given 2D device coordinate.
  
  Result = Obj -> [IDLgrROIGroup::Add( [ , PROPERTY=value ])]

- **GetByName** - Finds contained objects by name and returns the object reference to the named object.
  
  Result = Obj -> [IDLgrROIGroup::GetByName( Name )]

- **Init** - Initializes the scene object.
  
  Obj = OBJ_NEW('IDLgrScene' [ , PROPERTY=value ])

- **Remove** - Removes an object from the ROI group.
  
  Obj -> [IDLgrROIGroup::Remove( Child_object )]

- **SetProperty** - Sets the value of a property or group of properties for the region group.
  
  Obj -> [IDLgrROIGroup::SetProperty [ , PROPERTY=value ]]

**IDLgrScene**

- **Add** - Verifies that the added item is an instance of an IDLgrView or IDLgrViewgroup object.
  
  Obj -> [IDLgrScene::Add( View [, POSITION=index ])]

- **Cleanup** - Performs all cleanup on the object.
  
  OBJ_DESTROY, Obj or Obj -> [IDLgrScene::Cleanup]

- **GetProperty** - Retrieves the value of a property or group of properties for the scene.
  
  Obj -> [IDLgrScene::GetProperty [ , PROPERTY=value ]]

- **Init** - Initializes the scene.
  
  Obj = OBJ_NEW('IDLgrScene')

- **Remove** - Removes an object from the scene.
  
  Obj -> [IDLgrScene::Remove( Child_object )]

- **SetProperty** - Sets the value of one or more properties for the scene.
  
  Obj -> [IDLgrScene::SetProperty [ , PROPERTY=value ]]

**IDLgrShader**

- **Add** - Verifies that the added item is an instance of an IDLgrShader or IDLgrShaderBytscl object.
  
  Obj -> [IDLgrShader::Add( Child_object )]

- **Cleanup** - Performs all cleanup on the object.
  
  OBJ_DESTROY, Obj or Obj -> [IDLgrShader::Cleanup]

- **GetProperty** - Retrieves the value of a property or group of properties for the shader.
  
  Obj -> [IDLgrShader::GetProperty [ , PROPERTY=value ]]

- **Init** - Initializes the shader object.
  
  Obj = OBJ_NEW('IDLgrShader')

- **SetUniformVariable** - Sets the value of a uniform variable specified in the Name argument (a case-sensitive string).
  
  Result = Obj -> [IDLgrShader::SetUniformVariable( Name, Value )]

**IDLgrShaderBytscl**

- **Add** - Verifies that the added item is an instance of an IDLgrShaderBytscl object.
  
  Obj -> [IDLgrShaderBytscl::Add( Child_object )]

- **Cleanup** - Performs all cleanup on the object.
  
  OBJ_DESTROY, Obj or Obj -> [IDLgrShaderBytscl::Cleanup]

- **GetProperty** - Retrieves the value of a property or group of properties for the byte-scale shader.
  
  Obj -> [IDLgrShaderBytscl::GetProperty [ , PROPERTY=value ]]

- **Init** - Initializes the byte-scale shader object.
  
  Obj = OBJ_NEW('IDLgrShaderBytscl')
IDLgrShaderConvol3 - Performs a convolution operation (smooth, sharpen, edge detection, or custom) on image data as it is drawn to the screen using a shader (with sufficient graphics hardware support).

**Properties:**
- `BASE_BLEND_FACTOR` (Get, Init, Set=value)
- `CONVOL_BLEND_FACTOR` (Get, Init, Set=value)
- `KERNEL` (Get, Init, Set=value)

**Methods:**
- `Init`
- `Cleanup`
- `setProperty`
- `getProperty`

IDLgrShaderConvol3::Init - Initializes the convolution shader object.

```
Obj = OBJ_NEW('IDLgrShaderConvol3'
[PROPERTY=value])
```

IDLgrShaderConvol3::Cleanup - Performs all cleanup on the object.

```
OBJ_DESTROY, Obj or Obj->[IDLgrShaderConvol3::]Cleanup
```

IDLgrShaderConvol3::setProperty - Sets the value of a property or group of properties for the convolution shader.

```
Obj->[IDLgrShaderConvol3::]setProperty
[PROPERTY=variable]
```

IDLgrShaderConvol3::getProperty - Retrieves the value of a property or group of properties for the convolution shader.

```
Obj->[IDLgrShaderConvol3::]getProperty
[PROPERTY=variable]
```

IDLgrSurface - A shaded vector representation of a mesh grid.

**Properties:**
- `ALL` (Get=value)
  - `ALPHA_CHANNEL` (Get, Init, Set=value)
  - `BOTTOM` (Get, Set) = index or RGB vector
  - `CLIP_PLANES` (Get, Init, Set) = array
  - `COLOR` (Get, Init, Set) = index or RGB vector
  - `DATAX` (Init, Set) = vector or 2D array
  - `DATAY` (Init, Set) = vector or 2D array
  - `DATAZ` (Init, Set) = 2D array
  - `DEPTH_OFFSET` (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]
  - `DIFFUSE` (Get, Init, Set) = RGB vector
  - `DOUBLE` (Get, Init, Set) = variable
  - `EMMISION` (Get, Init, Set) = RGB vector
  - `EXTENDED_LEGEND` (Get, Init, Set)
  - `HIDDEN_LINES` (Get, Init, Set)
  - `HIDE` (Get, Init, Set)
  - `LINESTYLE` (Get, Init, Set) = variable
  - `MAX_VALUE` (Get, Init, Set) = variable
  - `MIN_VALUE` (Get, Init, Set) = variable
  - `PALETTE` (Get, Init, Set) = variable
  - `PARENT` (Get) = variable
  - `REGISTER_PROPERTIES` (Get, Init, Set)
  - `RESET_DATA` (Init, Set)
  - `SHADE` (Get, Init, Set) = index of darkest pixel, index of brightest pixel
  - `SHADING` (Get, Init, Set) = objref
  - `SHARE_DATA` (Init, Set) = objref
  - `SHININESS` (Get, Init, Set) = variable
  - `SHOW_SKIRT` (Get, Init, Set)
  - `SKIRT` (Get, Init, Set) = Z value
  - `SPECULAR` (Get, Init, Set) = RGB vector
  - `STYLE` (Get, Init, Set) = [0 | 1 | 2 | 3 | 4 | 5 | 6]
  - `TEXTURECOORD` (Get, Init, Set) = array
  - `TEXTURE_HIGHER` (Get, Init, Set)
  - `TEXTURE_INTERP` (Get, Init, Set)

**Methods:**
- `Init`
- `Cleanup`
- `setProperty`
- `getProperty`
**Alphabetical List of IDL Routines**

**Obj->[IDLgrSurface::]SetVertexAttributeData** - Sets the value of the vertex attribute data variable specified in the *Name* argument (a case-sensitive string).

**Properties:**
- **ALL** [Get] = variable
- **COLOR** [Get, Init, Set] = index or RGB vector
- **DATA** [Get, Init, Set] = integer or objref
- **NAME** [Get, Init, Set] = string
- **SIZE** [Get, Init, Set] = vector
- **THICK** [Get, Init, Set] = points [1.0 to 10.0]
- **UVALUE** [Get, Init, Set] = value

**Obj->[IDLgrSymbol::]SetVertexAttributeData,**

**Name,**

**Value**

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

**Properties:**
- **AlphaChannel** [Get, Set] = value
- **Color** [Get, Init, Set] = index or RGB vector
- **Data** [Get, Init, Set] = integer or objref
- **Name** [Get, Init, Set] = string
- **Size** [Get, Init, Set] = vector
- **Thick** [Get, Init, Set] = points [1.0 to 10.0]
- **UValue** [Get, Init, Set] = value

**Obj->[IDLgrSymbol::]Cleanup,**

**IDLgrSymbol::GetProperty** - Retrieves the value of a property or group of properties for the symbol.

**Obj->[IDLgrSymbol::]GetProperty,**

**PROPERTY** = variable

**Obj->[IDLgrSymbol::]Init** - Initializes the plot symbol.

**Obj = OBJ_NEW('IDLgrSymbol',**

**[,**

**Data**,**

**PROPERTY** = value**

**]) or**

**Result = Obj->[IDLgrSymbol::]Init([Data**

**,**

**PROPERTY** = value**

**])**

**Obj->[IDLgrSymbol::]SetProperty** - Sets the value of a property or group of properties for the symbol.

**Obj->[IDLgrSymbol::]Cleanup,**

**IDLgrTessellator** - Converts a simple concave polygon (or a simple polygon with “holes”) into a number of simple convex polygons (general triangles).

**Obj->[IDLgrTessellator::]AddPolygon,**

**X** [,**

**Y** [,**

**Z**]**

**,**

**AUXDATA = array of auxiliary data** [,**

**/INTERIOR**]

**,**

**POLYGON = array of polygon descriptions**

**Obj->[IDLgrTessellator::]Cleanup,**

**IDLgrTessellator::Init** - Initializes the tessellator object.

**Obj = OBJ_NEW('IDLgrTessellator') or**

**Result = Obj->[IDLgrTessellator::]Init()**

**IDLgrTessellator::Reset** - Resets the object’s internal state.

**Obj->[IDLgrTessellator::]Reset**

**IDLgrTessellator::Tessellate** - Performs the actual tessellation.

**Result = Obj->[IDLgrTessellator::]Tessellate(Vertices,**

**Poly**, **AUXDATA = variable** [,**

**/QUIET** ]

**])**

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

**Properties:**
- **AlphaChannel** [Get, Init, Set] = value [0.0 to 1.0]
- **Color** [Get, Init, Set] = index or RGB vector
- **DepthTestDisable** [Get, Init, Set] = {0 | 1 | 2}
- **DepthTestFunction** [Get, Init, Set] = {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}
- **DepthWriteDisable** [Get, Init, Set] = {0 | 1 | 2}
- **EnableFormatting** [Get, Init, Set]
- **FillBackgroundColor** [Get, Init, Set]
- **FillColor** [Get, Init, Set] = index or RGB vector
- **Font** [Get, Init, Set] = objref
- **Kerning** [Get, Init, Set]
- **Locations** [Get, Init, Set] = array
- **NAME** [Get, Init, Set] = string
- **/Onglass** [Get, Init, Set]
- **/Palette** [Get, Init, Set] = objref
- **/Parent** [Get, variable]
- **/RegisterProperties** [Get, Init, Set]
- **RendererMethod** [Get, Init, Set] = {0 | 1 | 2}
- **Shader** [Get, Init, Set] = objref
- **Strings** [Get, Init, Set] = string or vector of strings
- **UDIR** [Get, Init, Set] = vector
- **UValue** [Get, Init, Set] = value
- **VerticalAlignment** [Get, Init, Set] = value [0.0 to 1.0]
- **XcoIndexConv** [Get, Init, Set] = vector
- **Xrange** [Get] = variable
- **YcoIndexConv** [Get, Init, Set] = vector

**Obj->[IDLgrText::]Cleanup,**

**Obj->[IDLgrText::]GetProperty,**

**PROPERTY** = variable
IDLgrText::Cleanup - Performs all cleanup on the object.

IDLgrText::GetCTM - Returns the 4 x 4 graphics transform matrix from the current object upward through the graphics tree.

IDLgrText::GetProperty - Retrieves the value of a property or group of properties for the text.

IDLgrText::Init - Initializes the text object.

IDLgrText::SetProperty - Sets the value of a property or group of properties for the text.

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} = RGB vector] [ , DEPTH_CUE {Get, Init, Set} = {0 | 1 | 2 | 3}] [ , /TRANSPARENT {Get, Init, Set} = variable] [ , 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, d0, d0, d0] ] [ , DATA1 {Get, Init, Set} = [d1, d1, d1] ] [ , DATA2 {Get, Init, Set} = [d2, d2, d2] ] [ , DATA3 {Get, Init, Set} = [d3, d3, d3] ] [ , DEPTH_CUE {Get, Init, Set} = {bright, zdim} ] [ , /HIDE {Get, Init, Set} = 0 | 1 | 2 | 3] ] [ , /INTERPOLATE {Get, Init, Set} = variable] [ , /LIGHTING_MODEL {Get, Init, Set} = variable] [ , OPACITY_TABLE0 {Get, Init, Set} = 256-element byte array] [ , OPACITY_TABLE1 {Get, Init, Set} = 256-element byte array] [ , RENDER_STEP {Get, Init, Set} = x, y, z] [ , RGB_TABLE0 {Get, Init, Set} = 256 x 3-element byte array] [ , RGB_TABLE1 {Get, Init, Set} = 256 x 3-element byte array] [ , /REGISTER_PROPERTIES {Get, Init, Set} = variable] [ , SHADER {Get, Init, Set} = objref] [ , /TWO_SIDED {Get, Init, Set} = variable] [ , VALID_DATA {Get} = variable]

IDLgrViewgroup - A simple container object that contains one or more IDLgrView objects. An IDLgrScene can contain one or more of these objects.

Properties: [ , ALL {Get} = variable] [ , /HIDE {Get, Init, Set} = variable] [ , PARENT {Get} = variable] [ , /REGISTER_PROPERTIES {Get, Init, Set} = variable] [ , /TWO_SIDED {Get, Init, Set} = variable] [ , VALID_DATA {Get} = variable]
IDLgrVolume::PickVoxel - Computes the coordinates of the voxel projected to a location specified by the 2D device coordinates point, $(x_i, y_i)$, and the current Z-buffer.

Result = Obj->IDLgrVolume::PickVoxel( Win, View, Point, [PATH=objcref[]])

IDLgrVolume::SetProperty - Sets the value of a property or group of properties for the volume.

Obj->IDLgrVolume::SetProperty, PROPERTY=value

[ , /NO_COPY]
IDLgrWindow - Represents an on-screen area on a display device that serves as a graphics destination.

**Properties:**
- [ , ALL=variable
- [ COLOR_MODEL=variable
- [ DIMENSIONS=variable
- [ DISPLAY_NAME=variable
- [ IMAGE_DATA=variable
- [ GRAPHICS_TREE=variable
- [ IDLgrWindow::GetDeviceInfo = objref of type IDLgrScene, IDLgrViewgroup, or IDLgrView
- [ LOCATION=variable
- [ MINIMUM_VIRTUAL_DIMENSIONS=variable
- [ N_COLORS=variable
- [ PACKAGE=variable
- [ RENDERER=variable
- [ SCREEN_DIMENSIONS=variable
- [ TITLE=variable
- [ VIRTUAL_DIMENSIONS=variable
- [ VISIBLE_LOCATION=variable
- [ VISIBLE_LOCATION=variable

**Methods:**
- [ IDLgrWindow::Cleanup() - Performs all cleanup on the object.
- [ IDLgrWindow::Draw() - Draws the specified scene or view object to this graphics destination.
- [ IDLgrWindow::GetDimensions() - Returns a two-element vector, [width, height], representing the visible dimensions (in device units) of this window.
- [ IDLgrWindow::GetFontnames() - Returns the list of available fonts that can be used in IDLgrFont objects.
- [ IDLgrWindow::GetProperty() - Retrieves the value of a property or group of properties for the window.
- [ IDLgrWindow::GetTextDimensions() - Retrieves the dimensions of a text object that will be rendered in the window.

**Example:**
```
IDLgrWindow::GetDimensions() = [width, height]
```

**Properties:**
- [ , ALL=variable
- [ COLOR_MODEL=variable
- [ DIMENSIONS=variable
- [ DISPLAY_NAME=variable
- [ IMAGE_DATA=variable
- [ GRAPHICS_TREE=variable
- [ IDLgrWindow::GetDeviceInfo = objref of type IDLgrScene, IDLgrViewgroup, or IDLgrView
- [ LOCATION=variable
- [ MINIMUM_VIRTUAL_DIMENSIONS=variable
- [ N_COLORS=variable
- [ PACKAGE=variable
- [ RENDERER=variable
- [ SCREEN_DIMENSIONS=variable
- [ TITLE=variable
- [ VIRTUAL_DIMENSIONS=variable
- [ VISIBLE_LOCATION=variable

**Methods:**
- [ IDLgrWindow::Cleanup() - Performs all cleanup on the object.
- [ IDLgrWindow::Draw() - Draws the specified scene or view object to this graphics destination.
- [ IDLgrWindow::GetDimensions() - Returns a two-element vector, [width, height], representing the visible dimensions (in device units) of this window.

**Example:**
```
IDLgrWindow::GetDimensions() = [width, height]
```

**Properties:**
- [ , ALL=variable
- [ COLOR_MODEL=variable
- [ DIMENSIONS=variable
- [ DISPLAY_NAME=variable
- [ IMAGE_DATA=variable
- [ GRAPHICS_TREE=variable
- [ IDLgrWindow::GetDeviceInfo = objref of type IDLgrScene, IDLgrViewgroup, or IDLgrView
- [ LOCATION=variable
- [ MINIMUM_VIRTUAL_DIMENSIONS=variable
- [ N_COLORS=variable
- [ PACKAGE=variable
- [ RENDERER=variable
- [ SCREEN_DIMENSIONS=variable
- [ TITLE=variable
- [ VIRTUAL_DIMENSIONS=variable
- [ VISIBLE_LOCATION=variable

**Methods:**
- [ IDLgrWindow::Cleanup() - Performs all cleanup on the object.
- [ IDLgrWindow::Draw() - Draws the specified scene or view object to this graphics destination.
- [ IDLgrWindow::GetDimensions() - Returns a two-element vector, [width, height], representing the visible dimensions (in device units) of this window.

**Example:**
```
IDLgrWindow::GetDimensions() = [width, height]
```

**Properties:**
- [ , ALL=variable
- [ COLOR_MODEL=variable
- [ DIMENSIONS=variable
- [ DISPLAY_NAME=variable
- [ IMAGE_DATA=variable
- [ GRAPHICS_TREE=variable
- [ IDLgrWindow::GetDeviceInfo = objref of type IDLgrScene, IDLgrViewgroup, or IDLgrView
- [ LOCATION=variable
- [ MINIMUM_VIRTUAL_DIMENSIONS=variable
- [ N_COLORS=variable
- [ PACKAGE=variable
- [ RENDERER=variable
- [ SCREEN_DIMENSIONS=variable
- [ TITLE=variable
- [ VIRTUAL_DIMENSIONS=variable
- [ VISIBLE_LOCATION=variable

**Methods:**
- [ IDLgrWindow::Cleanup() - Performs all cleanup on the object.
- [ IDLgrWindow::Draw() - Draws the specified scene or view object to this graphics destination.
- [ IDLgrWindow::GetDimensions() - Returns a two-element vector, [width, height], representing the visible dimensions (in device units) of this window.

**Example:**
```
IDLgrWindow::GetDimensions() = [width, height]
```

**Properties:**
- [ , ALL=variable
- [ COLOR_MODEL=variable
- [ DIMENSIONS=variable
- [ DISPLAY_NAME=variable
- [ IMAGE_DATA=variable
- [ GRAPHICS_TREE=variable
- [ IDLgrWindow::GetDeviceInfo = objref of type IDLgrScene, IDLgrViewgroup, or IDLgrView
- [ LOCATION=variable
- [ MINIMUM_VIRTUAL_DIMENSIONS=variable
- [ N_COLORS=variable
- [ PACKAGE=variable
- [ RENDERER=variable
- [ SCREEN_DIMENSIONS=variable
- [ TITLE=variable
- [ VIRTUAL_DIMENSIONS=variable
- [ VISIBLE_LOCATION=variable

**Methods:**
- [ IDLgrWindow::Cleanup() - Performs all cleanup on the object.
- [ IDLgrWindow::Draw() - Draws the specified scene or view object to this graphics destination.
- [ IDLgrWindow::GetDimensions() - Returns a two-element vector, [width, height], representing the visible dimensions (in device units) of this window.

**Example:**
```
IDLgrWindow::GetDimensions() = [width, height]
```
IDLgrWindow::SetCurrentCursor - 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->[IDLgrWindow::]SetCurrentCursor , CursorName

IDLgrWindow::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->[IDLgrWindow::]OnMouseDown , X, Y, ButtonMask,Modifiers

IDLgrWindow::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->[IDLgrWindow::]OnMouseMotion , X, Y, Modifiers

IDLgrWindow::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->[IDLgrWindow::]OnMouseUp , X, Y, ButtonMask

IDLgrWindow::OnResize - Handles notification (from the native window device) that a resize event has occurred.

Obj->[IDLgrWindow::]OnResize, Width, Height

IDLgrWindow::OnWheel - Handles notification (from the native window device) that a mouse wheel event has occurred.

Obj->[IDLgrWindow::]OnWheel, X, Y, Delta, Modifiers

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=object(s)] [, PICK_STATUS=variable])

IDLgrWindow::QueryRequiredTiles - Returns an array of named structures containing information regarding which tile data is needed for display. Used with a tiled IDLgrImage object.

Result = Obj->[IDLgrWindow::]QueryRequiredTiles (View, Image [, COUNT=variable] [, ALL_VISIBLE=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] [, HOTSPOTS=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]

IDLgrWindow::SetProperty - Sets the value of a property or group of properties for the window.

Obj->[IDLgrWindow::]SetProperty [, PROPERTY=value]

IDLgrWindow::Show - Expases or hides a window.

Obj->[IDLgrWindow::]Show, Position

IDLgrWindow::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).

Obj->[IDLgrWindow::]ZoomIn

IDLgrWindow::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).

Obj->[IDLgrWindow::]ZoomOut

IDLitCommand - The base functionality for the iTools command buffer system.

Properties: [, SKIP_REDO{Get, Init, Set}=0 | 1] [, SKIP_UNDO{Get, Init, Set}=0 | 1] [, TARGET_IDENTIFIER{Get, Init, Set}=string] [, OPERATION_IDENTIFIER{Get, Init, Set}=string]

IDLitCommand::AddItem - Adds the specified data item to the data dictionary associated with this object.

Result = Obj->[IDLitCommand::]AddItem( StrItem, 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( StrItem, 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=variable])

IDLitCommand::SetProperty - Sets the value of a property or group of properties for the command object.

Obj->[IDLitCommand::]SetProperty [, PROPERTY=value]
<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLitCommandSet</td>
<td>A container for IDLitCommand objects, which allows a group of commands to be managed as a single item.</td>
</tr>
<tr>
<td>IDLitCommandSet::Cleanup</td>
<td>Performs all cleanup on the object.</td>
</tr>
<tr>
<td>IDLitCommandSet::GetSize</td>
<td>Returns an approximate value for the amount of memory being used by the items contained by this command set.</td>
</tr>
<tr>
<td>IDLitCommandSet::Init</td>
<td>Initializes the object.</td>
</tr>
<tr>
<td>IDLitCommandSet::NotifyBridge</td>
<td>Broadcasts a notification message to the Export Bridge object that is wrapping the IDLitComponent object.</td>
</tr>
<tr>
<td>IDLitCommandSet::QueryProperty</td>
<td>Checks whether a property identifier is registered, or retrieves a list of all registered properties.</td>
</tr>
<tr>
<td>IDLitCommandSet::RegisterProperty</td>
<td>Registers a property as belonging to the component.</td>
</tr>
<tr>
<td>IDLitCommandSet::Restore</td>
<td>Performs any transitional work required after an object has been restored from a SAVE file.</td>
</tr>
<tr>
<td>IDLitCommandSet::SetProperty</td>
<td>Sets the value of an IDLitComponent property or properties.</td>
</tr>
<tr>
<td>IDLitCommandSet::SetPropertyByIdentifier</td>
<td>Sets the value of an IDLitComponent property.</td>
</tr>
<tr>
<td>IDLitCommandSet::UpdateComponentVersion</td>
<td>Updates the value of the COMPONENT_VERSION property for the specified object to the current release of IDLitComponent.</td>
</tr>
<tr>
<td>IDLitContainer</td>
<td>A specialization of the IDLit_Container class that manages a collection of IDLitComponents and provides methods for working with the Identifier system of the iTools framework.</td>
</tr>
<tr>
<td>IDLitContainer::Add</td>
<td>Adds items to the container object.</td>
</tr>
<tr>
<td>IDLitContainer::AddByIdentifier</td>
<td>Adds an object to the container hierarchy in the position specified by the Identifier argument.</td>
</tr>
<tr>
<td>IDLitContainer::Cleanup</td>
<td>Performs all cleanup on the object.</td>
</tr>
</tbody>
</table>

**IDL Quick Reference**
IDL Quick Reference

IDLitContainer::FindIdentifiers - Retrieves the full identifiers for items within an iTool container.
Result = Obj->IDLitContainer::FindIdentifiers([Pattern] [ , COUNT=variable] [ , /LEAF_NODES] )

IDLitContainer::Get - Retrieves items from the container.
Result = Obj->[IDLitContainer::Get[ , /ALL] [ , COUNT=variable] [ , ISA=string or array of strings] [ , POSITION=index or array of indices] [ , /SKIP_PRIVATE])

IDLitContainer::GetByIdentifier - Retrieves an object from a container hierarchy using the specified identifier to locate the object.
Result = Obj->[IDLitContainer::GetByIdentifier( Identifier)]

IDLitContainer::_Init - Initializes the object.
Obj = OBJ_NEW('IDLitContainer') or
Result = Obj->[IDLitContainer::Init()]

IDLitContainer::Remove - Removes items from the container.
Obj->[IDLitContainer::]Remove, Components [ , /NO_NOTIFY]?

IDLitContainer::RemoveByIdentifier - Removes an object from a container hierarchy using the specified identifier to locate the object.
Result = Obj->[IDLitContainer::]RemoveByIdentifier( Identifier)

IDLitData - 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}] [ , /READ_ONLY{Init}] [ , /TYPE{Get, Init}=string]

IDLitData::AddDataObserver - Specifies an object (the Observer) that will be notified when the contents of the data object are changed.
Obj->[IDLitData::]AddDataObserver, Observer

IDLitData::Cleanup - Performs all cleanup operations on the object.
OBJ_DESTROY, Obj or Obj->[IDLitData::]Cleanup

IDLitData::Copy - Returns an exact copy of the data object and its contents, including registered property values.
Result = Obj->[IDLitData::]Copy()

IDLitData::GetByType - Returns all contained objects of the specified iTool data type.
Result = Obj->[IDLitData::]GetByType([Type] [ , COUNT=variable])

IDLitData::GetData - Retrieves the data stored in the object.
Result = Obj->[IDLitData::]GetData([Data], Identifier [ , NAN=variable] [ , /NO_COPY])

IDLitData::GetProperty - Retrieves the value of an IDLitData property.
Obj->[IDLitData::]GetProperty[ , PROPERTY=variable]

IDLitData::GetSize - Returns an approximate value for the amount of memory being used by the data object.
Result = Obj->[IDLitData::]GetSize()

IDLitData::Init - Initializes the IDLitData object.
Obj = OBJ_NEW('IDLitData', [ Data] [ , PROPERTY=value]) or
Result = Obj->[IDLitData::]Init([Data] [ , PROPERTY=value])

IDLitData::NotifyDataChange - Is called when a data object has been changed; it is part of the notification process that allows data updates to be reflected by visualizations that use the data. It works in conjunction with the NotifyDataComplete method to provide a two-pass change notification system that minimizes the number of operations performed when a data object changes.
Obj->[IDLitData::]NotifyDataChange

IDLitData::NotifyDataComplete - Is called after a data object has been changed; it is part of the notification process that allows data updates to be reflected by visualizations that use the data. It works in conjunction with the NotifyDataChange method to provide a two-pass change notification system that minimizes the number of operations performed when a data object changes.
Obj->[IDLitData::]NotifyDataComplete

IDLitData::RemoveDataObserver - Unregisters an object that has been registered as an observer of this data object.
Obj->[IDLitData::]RemoveDataObserver, Observer

IDLitData::SetData - Copies the data from an IDL variable or expression into the data object, and notifies all its observers that the data has changed.
Result = Obj->[IDLitData::]SetData([Data], Identifier [ , /NO_COPY] [ , /NULL])

IDLitData::SetProperty - Sets the value of an IDLitData property
Obj->[IDLitData::]SetProperty[ , PROPERTY=value]

IDLitDataContainer - 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 composite data types.

IDLitDataContainer::Add - Adds items to the data container object.
Obj->[IDLitDataContainer::]Add, Data [ , /NO_NOTIFY]

IDLitDataContainer::Cleanup - Performs all cleanup operations on the object.
OBJ_DESTROY, Obj or Obj->[IDLitDataContainer::]Cleanup

IDLitDataContainer::GetByIdentifier - Retrieves the data value contained in the data object specified by the Identifier argument.
Result = Obj->[IDLitDataContainer::]GetByIdentifier([Data], Identifier [ , /NO_COPY])

IDLitDataContainer::GetIdentifiers - Retrieves the object identifiers for all data and data container objects contained in the data container object.
Result = Obj->[IDLitDataContainer::]GetIdentifiers([Pattern] [ , /LEAF])
IDLitDataContainer::GetProperty - Retrieves the value of an IDLitDataContainer property
  Result = Obj->IDLitDataContainer::GetProperty([, PROPERTY=value])

IDLitDataContainer::Init - Initializes the IDLitDataContainer object.
  Obj = OBJ_NEW(IDLitDataContainer [, Data] [, PROPERTY=value]) or
  Result = Obj->IDLitDataContainer::Init([Data] [, PROPERTY=value])

IDLitDataContainer::SetData - Stores data in the IDLitData object specified by Identifier.
  Result = Obj->IDLitDataContainer::SetData(Data, Identifier [, NO_COPY] [, /NULL])

IDLitDataContainer::SetProperty - Sets the value of an IDLitDataContainer property.
  Obj->IDLitDataContainer::SetProperty [, PROPERTY=value]

IDLitDataOperation - A subclass to IDLitOperation that automates data access and automatically records information for the undo-redo system.
  Properties: [, WITHIN_UI [GET]=1 | 0]

IDLitDataOperation::Cleanup - Performs all cleanup on the object.
  OBJ_DESTROY, Obj or
  Obj->IDLitDataOperation::Cleanup

IDLitDataOperation::DoExecuteUI - Provides a way for the iTool developer to request user input before performing an operation.
  Result = Obj->IDLitDataOperation::DoExecuteUI()

IDLitDataOperation::Execute - Contains the execution logic for the operation.
  Result = Obj->IDLitDataOperation::Execute(Data)

IDLitDataOperation::GetProperty - Retrieves the value of a property or group of properties of an operation object.
  Obj->IDLitDataOperation::GetProperty [, PROPERTY=value]

IDLitDataOperation::Init - Initializes the IDLitDataOperation object and sets properties that define the behavior the operation provides.
  Obj = OBJ_NEW(IDLitDataOperation' [, PROPERTY=value]) or
  Result = Obj->IDLitDataOperation::Init([, PROPERTY=value])

IDLitDataOperation::SetProperty - Sets the value of a property or group of properties for the operation.
  Obj->IDLitDataOperation::SetProperty [, PROPERTY=value]

IDLitDataOperation::UndoExecute - IS called when a user selects the Unde operation after executing an IDLitDataOperation that sets the value of the REVERSIBLE_OPERATION property to 1.
  Result = Obj->IDLitDataOperation::UndoExecute(Data)

IDLitDataContainer - A representation of an on-screen area on a display device that serves as a graphics destination. In contrast to IDLgrWindow and IDLitWindow, the IDLitDirectWindow uses Direct Graphics. This object is only for use with the Export Bridges.
  Properties: [, WINDOW_INDEX=integer]

IDLitDirectWindow::Cleanup - Performs all cleanup on the object.
  OBJ_DESTROY, Obj or
  Obj->IDLitDirectWindow::Cleanup

IDLitDirectWindow::GetProperty - Retrieves the value of an IDLitDirectWindow property.
  Obj->IDLitDirectWindow::GetProperty[, PROPERTY=value]

IDLitDirectWindow::Init - Initializes the window object.
  Obj = OBJ_NEW(IDLitDirectWindow' [, PROPERTY=value]) or
  Result = Obj->IDLitDirectWindow::Init([, PROPERTY=value])

IDLitDirectWindow::MakeCurrent - Sets the current graphics device to the window device and sets the current output window to the IDLitDirectWindow's window
  Obj->IDLitDirectWindow::MakeCurrent

IDLitDirectWindow::OnEnter - Handles notification from the window that a mouse enter event has occurred.
  Obj->IDLitDirectWindow::OnEnter

IDLitDirectWindow::OnExit - Handles notification from the window that a mouse exit event has occurred.
  Obj->IDLitDirectWindow::OnExit

IDLitDirectWindow::OnExpose - Handles notification from the window that an expose event has occurred.
  Obj->IDLitDirectWindow::OnExpose, X, Y, Width, Height

IDLitDirectWindow::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->IDLitDirectWindow::OnKeyboard, IsASCII, Character, KeySymbol, X, Y, Press, Release, Modifiers

IDLitDirectWindow::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->IDLitDirectWindow::OnMouseDown, X, Y, ButtonMask,Modifiers, NumClicks

IDLitDirectWindow::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->IDLitDirectWindow::OnMouseMotion, X, Y, Modifiers

IDLitDirectWindow::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->IDLitDirectWindow::OnMouseUp, X, Y, ButtonMask
IDLitDirectWindow::OnResize - Handles notification (from the native window device) that a resize event has occurred.
Obj->IDLitDirectWindow::OnResize(), Width, Height

IDLitDirectWindow::OnWheel - Handles notification (from the native window device) that a mouse wheel event has occurred.
Obj->IDLitDirectWindow::OnWheel, X, Y, Delta, Modifiers

IDLitDirectWindow::SetProperty - Sets the value of an IDLDirectWindow property.
Obj->IDLitDirectWindow::SetProperty(), PROPERTY=value

IDLitMessaging - An interface providing common methods to send or trigger messaging and error actions, which may occur during execution.

IDLitMessaging::AddOnNotifyObserver - Is used to register a specified iTool component object to receive messages generated by the DoOnNotify method of another specified iTool component object.
Obj->IDLitMessaging::AddOnNotifyObserver, IdObserver, IdSubject

IDLitMessaging::DoOnNotify - Is used to broadcast a notification message to an IDL component objects that are observing the source of the message.
Obj->IDLitMessaging::DoOnNotify, IdOriginator, IdMessage, Value

IDLitMessaging::ErrorMessage - Is used to display an error message to the user.
Obj->IDLitMessaging::ErrorMessage(), StrMessage, SEVERITY=integer, TITLE=string

IDLitMessaging::GetTool - Returns an object reference to the iTool object associated with the object on which it is called.
Result = Obj->IDLitMessaging::GetTool()

IDLitMessaging::ProbeStatusMessage - Is used to display a status message to the user, which is displayed in a data-specific region of the user interface.
Obj->IDLitMessaging::ProbeStatusMessage(), StrMessage

IDLitMessaging::ProgressBar - Is used to display a progress bar to the user and update the displayed values.
Result = Obj->IDLitMessaging::ProgressBar(), StrMessage, CANCEL=string, PERCENT=value / SHUTDOWN

IDLitMessaging::PromptUserText - Is used to prompt the iTool user with a question and retrieve a text answer.
Result = Obj->IDLitMessaging::PromptUserText(), StrPrompt, Answer, TITLE=string

IDLitMessaging::PromptUserYesNo - Is used to prompt the user with a yes or no question and return an answer.
Result = Obj->IDLitMessaging::PromptUserYesNo(), StrPrompt, Answer, TITLE=string

IDLitMessaging::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.
Obj->IDLitMessaging::RemoveOnNotifyObserver, IdObserver, IdSubject

IDLitMessaging::SignalError - Is used to signal an error in the system.
Obj->IDLitMessaging::SignalError(), StrMessage, CODE=string, SEVERITY=integer

IDLitMessaging::StatusMessage - Is used to display a status message to the user.
Obj->IDLitMessaging::StatusMessage(), StrMessage, SEGMENT_IDENTIFIER=string

IDLitManipulator - The base functionality of the iTools manipulator system.

Properties: [\ BUTTON_EVENTS\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ DESCRIPTION\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ DISABLE\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ DRAG\ QUALITY\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ KEYBOARD_EVENTS\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ MOTION_EVENTS\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ OPERATION_IDENTIFIER\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ PARAMETER_IDENTIFIER\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ TRANSIENT\ DEFAULT\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ TRANSIENT\ MOTION\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ TYPES\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ VIEWS\ ONLY\ ]\ \ \ \ Get\ Init\ Set\ ]
\ \ \ \ [\ VISUAL\ TYPE\ ]\ \ \ \ Get\ Init\ Set\ ]

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( )

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 IDLitManipulator property.
Obj->IDLitManipulator::GetProperty(), PROPERTY=value

IDLitManipulator::Init - Initializes the IDLitManipulator object.
Obj = OBJ_NEW('IDLitManipulator', [
\ PROPERTY=value\ ])

IDLitManipulator::PromptUserText - Is used to prompt the iTool user with a question and retrieve a text answer.
Result = Obj->IDLitManipulator::PromptUserText()

IDLitManipulator::PromptUserYesNo - Is used to prompt the user with a yes or no question and return an answer.
Result = Obj->IDLitManipulator::PromptUserYesNo()
**IDL Quick Reference**

**Alphabetical List of IDL Routines**

**IDLitManipulator::OnKeyboard** - Is used when a keyboard event occurs on the target window (Win).

*Result = 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.

*Result = Obj->[IDLitManipulator::OnLoseCurrentManipulator]*

**IDLitManipulator::OnMouseDown** - Is used when a mouse down event occurs on the target window.

*Result = Obj->[IDLitManipulator::OnMouseDown, Win, X, Y, IButton, KeyMods, NCicks]*

**IDLitManipulator::OnMouseMotion** - Manages the setting of the cursor on the window if no mouse button is down.

*Result = Obj->[IDLitManipulator::OnMouseMotion, Win, X, Y, KeyMods]*

**IDLitManipulator::OnMouseUp** - Is used when a mouse up event occurs on the target window.

*Result = Obj->[IDLitManipulator::OnMouseUp, Win, X, Y, IButton]*

**IDLitManipulator::OnWheel** - Handles notification (from the native window device) that a mouse wheel event has occurred.

*Result = Obj->[IDLitManipulator::OnWheel, X, Y, Delta, Modifiers]*

**IDLitManipulator::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.

*Result = Obj->[IDLitManipulator::RecordUndoValues()]*

**IDLitManipulator::RegisterCursor** - Defines the appearance and name of the cursor associated with the manipulator.

*Result = Obj->[IDLitManipulator::RegisterCursor, ArrCursor, Name [, /DEFAULT]]*

**IDLitManipulator::SetCurrentManipulator** - Sets the manipulator as the current manipulator in the system.

*Result = Obj->[IDLitManipulator::SetCurrentManipulator]*

**IDLitManipulator::SetProperty** - Sets the value of an IDLitManipulator property

*Result = Obj->[IDLitManipulator::SetProperty [, PROPERTY=value]]*

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

*Property: [ /AUTO_SWITCH {Get, Init, Set}]*

**IDLitManipulatorContainer::Add** - Is used to add a new manipulator to the container.

*Result = Obj->[IDLitManipulatorContainer::Add, Manipulator]*

**IDLitManipulatorContainer::GetCurrent** - Is used to get the object reference of the current manipulator of the container.

*Result = Obj->[IDLitManipulatorContainer::GetCurrent()]*

**IDLitManipulatorContainer::GetCurrentManipulator** - Is used to get the current manipulator of the system.

*Result = Obj->[IDLitManipulatorContainer::] GetCurrentManipulator([ /IDENTIFIER])*

**IDLitManipulatorContainer::GetProperty** - Retrieves the value of an IDLitManipulatorContainer property

*Result = Obj->[IDLitManipulatorContainer::]GetProperty [, PROPERTY=variable]*

**IDLitManipulatorContainer::Init** - Initializes the IDLitManipulatorContainer object.

*Result = Obj = OBJ_NEW('IDLitManipulatorContainer' [, PROPERTY=value]) or Result = Obj->[IDLitManipulatorContainer::]Init( [PROPERTY=value])*

**IDLitManipulatorContainer::OnKeyboard** - Is used when a keyboard event occurs on the target window (Win).

*Result = Obj->[IDLitManipulator::OnKeyboard, Win, IsASCII, Character, Key Value, X, Y, Press, Release, KeyMods]*

**IDLitManipulatorContainer::OnMouseMotion** - Manages the setting of the cursor on the window if no mouse button is down.

*Result = Obj->[IDLitManipulatorContainer::OnMouseMotion, Win, X, Y, KeyMods]*

**IDLitManipulatorContainer::OnMouseUp** - Is used when a mouse up event occurs on the target window.

*Result = Obj->[IDLitManipulatorContainer::OnMouseUp, Win, X, Y, IButton]*

**IDLitManipulatorContainer::OnWheel** - Handles notification (from the native window device) that a mouse wheel event has occurred.

*Result = Obj->[IDLitManipulatorContainer::OnWheel, X, Y, Delta, Modifiers]*

**IDLitManipulatorContainer::SetCurrentManipulator** - Is used to set a manipulator within the container to be the current manipulator.

*Result = Obj->[IDLitManipulatorContainer::]SetCurrentManipulator, Identifier*

**IDLitManipulatorContainer::SetCurrentManipulator** - Is used to set the current child manipulator in a manipulator hierarchy.

*Result = Obj->[IDLitManipulatorContainer::] SetCurrentManipulator, Identifier*

**IDLitManipulatorContainer::SetProperty** - Sets the value of an IDLitManipulatorContainer property

*Result = Obj->[IDLitManipulator::]SetProperty [, PROPERTY=value]*
**IDL Quick Reference**

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

### IDLManipulatorManager::Add
- Is used to add manipulators to the manipulator manager.
  
  ```
  Obj->IDLManipulatorManager::Add, Manipulator
  [,.DEFAULT]
  ```

### IDLManipulatorManager::AddManipulatorObserver
- Is used to add an observer to the manipulator system.
  
  ```
  Obj->IDLManipulatorManager::AddManipulatorObserver, Observer
  ```

### IDLManipulatorManager::GetDefaultManipulator
- Returns a reference to the manipulator that was most recently added as the default manipulator.
  
  ```
  Result = Obj->IDLManipulatorManager::GetDefaultManipulator()
  ```

### IDLManipulatorManager::Init
- Initializes the IDLManipulatorManager object.
  
  ```
  Obj = OBJ_NEW('IDLManipulatorManager'
  [,.PROPERTY=value]) or
  Result = Obj->IDLManipulatorManager::Init([.
  PROPERTY=value])
  ```

### IDLManipulatorManager::RemoveManipulatorObserver
- Is used to remove a manipulator observer from the manipulator object.
  
  ```
  Obj->IDLManipulatorManager::RemoveManipulatorObserver, Observer
  ```

### IDLManipulatorVisual
- 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.
  
  ```
  OBJ_DESTROY, Obj or
  Obj->IDLManipulatorVisual::Cleanup
  ```

### IDLManipulatorVisual::GetProperty
- Retrieves the value of an IDLManipulatorVisual property.
  
  ```
  Obj->IDLManipulatorVisual::GetProperty
  ```

### IDLManipulatorVisual::Init
- Initializes the IDLManipulatorVisual object.
  
  ```
  Obj = OBJ_NEW('IDLManipulatorVisual'
  [,.PROPERTY=value]) or
  Result = Obj->IDLManipulatorVisual::Init([.
  PROPERTY=value])
  ```

### IDLManipulatorVisual::SetProperty
- Sets the value of an IDLManipulatorVisual property.
  
  ```
  Obj->IDLManipulatorVisual::SetProperty
  [,.PROPERTY=value]
  ```

### 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.
  
  Properties: [.EXPENSIVE_COMPUTATION{Get, Init}] [.REVERSIBLE_OPERATION{Get, Init}] [.SHOW_EXECUTION_UI{Get, Init}] [.TYPES{Get, Init}=string or string array]

### IDLOperation::Cleanup
- Performs all cleanup on the object.
  
  ```
  OBJ_DESTROY, Obj or
  Obj->IDLOperation::Cleanup
  ```

### IDLOperation::DoAction
- Is called when an operation is requested by the iTools system, either as the result of a user action such as selection of a menu item or toolbar button, or by another operation.
  
  ```
  Result = Obj->IDLOperation::DoAction(Tool)
  ```

### IDLOperation::GetProperty
- Retrieves the value of a property or group of properties of an operation object.
  
  ```
  Obj->IDLOperation::GetProperty
  [,.PROPERTY=value]
  ```

### IDLOperation::Init
- Initializes the IDLOperation object and sets properties that define the behavior the operation provides.
  
  ```
  Obj = OBJ_NEW('IDLOperation'
  [,.PROPERTY=value]) or
  Result = Obj->IDLOperation::Init([.
  PROPERTY=value])
  ```

### IDLOperation::QueryAvailability
- Determines whether an operation is applicable for the selected data and/or visualization.
  
  ```
  Result = Obj->IDLOperation::QueryAvailability(Tools, SetTypes)
  ```

### IDLOperation::RecordFinalValues
- Records the information needed to redo an operation.
  
  ```
  Result = Obj->IDLOperation::RecordFinalValues(CommandSets, Targets, IdProperty)
  ```

### IDLOperation::RecordInitialValues
- Records the information needed to undo an operation.
  
  ```
  Result = Obj->IDLOperation::RecordInitialValues(CommandSet, Targets, IdProperty)
  ```

### 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).
  
  ```
  Result = Obj->IDLOperation::RedoOperation(CommandSet)
  ```

### IDLOperation::SetProperty
- Sets the value of a property or group of properties for the operation.
  
  ```
  Obj->IDLOperation::SetProperty
  [,.PROPERTY=value]
  ```

### 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).
  
  ```
  Result = Obj->IDLOperation::UndoOperation(CommandSet)
  ```
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.

RESULT OBJ_DESTROY, Obj or Obj->IDLParameter:::Cleanup()

IDLParameter::GetParameter - Retrieves the IDLitData object associated with a registered parameter.

RESULT = Obj->IDLParameter:::GetParameter(Name [, ALL] [, COUNT=variable])

IDLParameter::GetParameterAttribute - Retrieves the value of an attribute associated with a registered parameter.

RESULT = Obj->IDLParameter:::GetParameterAttribute, ParamName [, ATTRIBUTE=variable]

IDLParameter::GetParameterSet - Returns a reference to the IDLitParameterSet object associated with the visualization object.

RESULT = Obj->IDLParameter:::GetParameterSet([, /DEEP_COPY])

IDLParameter::Init - Initializes object instance fields that contain parameter data.

RESULT = Obj->IDLParameter:::Init()

IDLParameter::OnDataChangeUpdate - Is called when a data value has been updated or a new data object has been associated with the visualization object.

RESULT = Obj->IDLParameter:::OnDataChangeUpdate, Data, ParameterName

IDLParameter::OnDataDisconnect - Is called when a data value has been disconnected from a parameter.

RESULT = Obj->IDLParameter:::OnDataDisconnect, ParameterName

IDLParameter::QueryParameter - Checks whether a parameter is registered, or retrieves the list of all registered parameters.

RESULT = Obj->IDLParameter:::QueryParameter(ParameterName [, COUNT=variable])

IDLParameter::RegisterParameter - Registers a parameter with the visualization object.


IDLParameter::SetData - Is used to set data in this interface, associating a data object with a given parameter.

RESULT = Obj->IDLParameter:::SetDataData [, /BY_VALUE] [, PARAMETER_NAME=string] [, NO_UPDATE]

IDLParameter::SetParameterAttribute - Sets one or more parameter attributes for a registered parameter.

RESULT = Obj->IDLParameter:::SetParameterAttribute, ParamName [, ATTRIBUTE=variable]

IDLParameter::SetParameterSet - Is used to associate an IDLitParameterSet object with the visualization object’s parameter interface.

RESULT = Obj->IDLParameter:::SetParameterSet(ParameterSet)

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

IDLParameterSet::Add - Is used to add data to the parameter set.

RESULT = Obj->IDLParameterSet:::Add, Data [, PARAMETER_NAME=string] [, /DEEP_COPY] [, /PRESERVE_LOCATION]

IDLParameterSet::Cleanup - Performs all cleanup on the parameter set object.

RESULT = Obj->IDLParameterSet:::Cleanup()

IDLParameterSet::Copy - Returns a copy of the parameter set and its contents.

RESULT = Obj->IDLParameterSet:::Copy()

IDLParameterSet::Get - Is used to retrieve one or more IDLitData objects from the parameter set.

RESULT = Obj->IDLParameterSet:::Get([, ALL] [, COUNT=variable] [, NAME=variable] [, POSITION=integer])

IDLParameterSet::GetByName - Returns the IDLitData object associated with a specified named parameter.

RESULT = Obj->IDLParameterSet:::GetByParameter(Name [, COUNT=variable] [, NAME=variable])

IDLParameterSet::GetParameterName - Retrieves the name of a specified parameter using a provided data object.

RESULT = Obj->IDLParameterSet:::GetParameterName(Data, Name)

IDLParameterSet::Init - Initializes the IDLitParameterSet object.

RESULT = Obj->IDLParameterSet:::Init()

IDLParameterSet::Remove - Is used to remove a data item from the parameter set.

RESULT = Obj->IDLParameterSet:::Remove [, Items] [, ALL] [, POSITION=integer]

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

IDLReader::Cleanup - Performs all cleanup on the object, and should be called by the subclass ‘Cleanup method.

RESULT = Obj->IDLReader:::Cleanup

IDLReader::GetData - Is called by the system to retrieve the data from the current file.

RESULT = Obj->IDLReader:::GetData(Data)

IDLReader::GetFileExtensions - Is called by the system to retrieve the file extensions supported by this particular reader.

RESULT = Obj->IDLReader:::GetFileExtensions([COUNT=variable])
IDLitTool::ActivateManipulator - Activates a manipulator that has been registered with the tool.

Result = Obj->IDLitTool::ActivateManipulator(Identifier[, /DEFAULT])

IDLitTool::Add - Adds any item to the tool.

Obj = Obj->IDLitTool::Add(Item)

IDLitTool::AddService - Adds a service to the tool.

Obj = 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 = 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 = Obj->IDLitTool::DisableUpdates()

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.

Result = Obj->IDLitTool::FindIdentifiers([Pattern]

[. /ANNOTATIONS] [. /COUNT=variable]

[. /DATA_MANAGER] [. /FILE_READERS]

[. /FILE_WRITERS] [. /LEAF_NODES]

[. /MANIPULATORS] [. /OPERATIONS]

[. /VISUALIZATIONS] )

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] [. COUNT=variable])

IDLitTool::GetFileWriter - Retrieves a file writer registered with the tool object.

Result = Obj->IDLitTool::GetFileReaders(Identifier

[. /ALL] [. COUNT=variable])

IDLitTool::GetManipulators - Retrieves the manipulators registered with the tool object.

Result = Obj->IDLitTool::GetManipulators(

[COUNT=variable])

IDLitTool::GetOperations - Retrieves the operations registered with the tool object.

Result = Obj->IDLitTool::GetOperations(Identifier

[. /IDENTIFIER=string] [. COUNT=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(

[COUNT=variable])

IDLitTool::GetService - Retrieves a service that has been registered with the tool.

Result = Obj->IDLitTool::GetService(IdService)
IDLTool::GetVisualization - Retrieves a visualization registered with the tool object.
Result = Obj->IDLTool::GetVisualizations(Name, [ , ALL ] , [ , COUNT=variable ])

IDLTool::RegisterCustomization - Registers a customization component with the tool.
Obj->IDLTool::RegisterCustomization, Name, ClassName [, /DEFAULT] [, DESCRIPTION=string] [, ICON=string] [, IDENTIFIER=string] [, PROXY=string]

IDLTool::RegisterFileWriter - Registers a file writer component with the tool.
Obj->IDLTool::RegisterFileWriter, Name, ClassName [, /DEFAULT] [, DESCRIPTION=string] [, ICON=string] [, IDENTIFIER=string] [, PROXY=string]

IDLTool::RegisterOperation - Registers an operation component with the tool.

IDLTool::RegisterStatusBarSegment - Registers a status message bar segment with the tool.
Obj->IDLTool::RegisterStatusBarSegment, Name, ClassName [, IDENTIFIER=string] [, NORMALIZED_WIDTH=scalar]
IDLitUI::GetTool - Returns an object reference to the iTool with which the user interface is associated.

Result = Obj->IDLitUI::GetTool()

IDLitUI::GetWidgetByName - 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.

Result = Obj->IDLitUI::GetWidgetByName(Name)

IDLitUI::Init - Initializes the IDLitUI object

Obj = OBJ_NEW(IDLitUI, oTool) or
Result = Obj->IDLitUI::Init(oTool)

IDLitUI::RegisterUIService - Registers a user interface service with the user interface.

Result = Obj->IDLitUI::RegisterUIService(Name, Callback)

IDLitUI::RegisterWidget - Registers an IDL widget hierarchy with the user interface object.

Result = Obj->IDLitUI::RegisterWidget(wID, Name, Callback[, /FLOATING])

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.

Result = Obj->IDLitUI::RemoveOnNotifyObserver, IdObserver, IdSubject

IDLitUI::setProperty - Sets the value of an IDLitUI property.

Result = Obj->IDLitUI::setProperty[ PROPERTY=value]

IDLitUI::UnRegisterUIService - Unregisters a user interface service with the user interface object.

Result = Obj->IDLitUI::UnRegisterUIService, Name

IDLitUI::UnRegisterWidget - Unregisters a widget with the user interface object.

Result = Obj->IDLitUI::UnRegisterWidget, Name

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

Properties:
[ CENTER_OF_ROTATION ] [Get, Init, Set=[x, y] or [x, y, z]]

 IDLitVisualization::GetCenterRotation - Returns the center of rotation for this visualization.
 Result = Obj->IDLitVisualization::GetCenterRotation

 IDLitVisualization::GetCenterRotation([ /NO_TRANSFORM ] [X_RANGE=[xmin, xmax]] [Y_RANGE=[ymin, ymax]] [Z_RANGE=[zmin, zmax]])

 IDLitVisualization::GetDefaultSelectionVisual - Returns an object that serves as the default selection visual for this visualization.

Result = Obj->IDLitVisualization::GetDefaultSelectionVisual()

 IDLitVisualization::GetDefaultSelectionVisual - Returns an object that serves as the default selection visual for this visualization.

Result = Obj->IDLitVisualization::GetDefaultSelectionVisual()

IDLitVisualization::GetDataString - Retrieves a description of this visualization’s data at the given x, y, and z location.

Result = Obj->IDLitVisualization::GetDataString( XYZLocation)

IDLitVisualization::GetDataSpace - Returns a reference to the parent data space object within the graphics hierarchy that contains the visualization.

Result = Obj->IDLitVisualization::GetDataSpace([ /UNNORMALIZED])

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[ PROPERTY=value]
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.
Obj = OBJ_NEW('IDLitVisualization' [, PROPERTY=value] ) or
Result = Obj->[IDLitVisualization::]Init( {PROPERTY=value})

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 dataspace changes.
Obj->[IDLitVisualization::]On2DRotate, Notifier, IsRotated

IDLitVisualization::OnAxesRequestChange - Ensures that objects that use the visualization’s data are notified when the axes of a contained object are 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 changes to the visualization’s data are complete.
Obj->[IDLitVisualization::]OnDataComplete, Notifier

IDLitVisualization::OnDataRangeChange - Ensures that objects that use the visualization’s data are notified when 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 dataspace’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 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
```

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 105 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->IDLitWindow::DoHitTest(X, Y [, DIMENSIONS=width, height] [, /ORDER] [, SUB_HIT=variable] [, UNITS={0 | 1 | 2 | 3}])
```
IDLlitWindow::GetProperty - Retrieves the value of an IDLit-Window property

IDLlitWindow::OnMouseUp - Handles notification (from the native window device) that a mouse up event has occurred.

IDLlitWindow::OnMouseDown - Handles notification (from the native window device) that a mouse down event has occurred.

IDLlitWindow::OnMouseMotion - Handles notification (from the native window device) that a mouse motion event has occurred.

IDLlitWindow::OnMouseUp - Handles notification (from the native window device) that a mouse up event has occurred.

IDLlitWindow::OnMouseWheel - Handles notification (from the native window device) that a mouse wheel event has occurred.

IDLlitWindow::OnKeyPress - Handles notification (from the native window device) that a keyboard event has occurred.

IDLlitWindow::OnKeyRelease - Handles notification (from the native window device) that a keyboard event has occurred.

IDLlitWindow::OnInit - Initializes the window object.

IDLlitWindow::OnEnter - Handles notification from the window that a mouse enter event has occurred.

IDLlitWindow::OnExit - Handles notification from the window that a mouse exit event has occurred.

IDLlitWindow::OnExpose - Handles notification from the window that an expose event has occurred.

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

IDLlitWindow::OnResize - Handles notification (from the native window device) that a resize event has occurred.

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).

IDLlitWindow::Init - Initializes the window object.

IDLlitWindow::SetProperty - Sets the value of an IDLitWindow property.

IDLlitWindow::SetTimerInterval - Specifies the number of seconds between timer events.

IDLlitWindow::SetManipulatorManager - Sets the given IDLitManipulatorManager object as the current manager of this window’s manipulators.

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).

IDLlitWindow::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.

IDLlitWindow::GetProperty - Retrieves the value of an IDLit-Window property

IDLlitWindow::GetEventMask - Returns a bitwise mask representing the events that are enabled for this window.

Result = Obj->IDLlitWindow::GetEventMask('[BUTTON_EVENTS=variable]
[.KEYBOARD_EVENTS=variable]
[.MOUSE_EVENTS=variable]
[.TIMER_EVENTS=variable]
[.TRACKING_EVENTS=variable]
[.WHEEL_EVENTS=variable])

IDLlitWindow::GetEventMask - Returns a bitwise mask representing the events that are enabled for this window.

Result = Obj->IDLlitWindow::GetEventMask('[BUTTON_EVENTS=variable]
[.KEYBOARD_EVENTS=variable]
[.MOUSE_EVENTS=variable]
[.TIMER_EVENTS=variable]
[.TRACKING_EVENTS=variable]
[.WHEEL_EVENTS=variable])

IDLlitWindow::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->IDLlitWindow::GetSelectedItems('[.ALL] [.COUNT=.named variable])

IDLlitWindow::Init - Initializes the window object.

Obj = OBJ_NEW('IDLlitWindow'[,
PROPERTY=value]) or
Result = Obj->IDLlitWindow::Init('[
PROPERTY=value])

IDLlitWindow::OnEnter - Handles notification from the window that a mouse enter event has occurred.

Obj->IDLlitWindow::OnEnter

IDLlitWindow::OnExit - Handles notification from the window that a mouse exit event has occurred.

Obj->IDLlitWindow::OnExit

IDLlitWindow::OnExpose - Handles notification from the window that an expose event has occurred.

Obj->IDLlitWindow::OnExpose, X, Y, Width, Height

IDLlitWindow::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->IDLlitWindow::OnKeyboard, JsASCII, Character, KeySymbol, X, Y, Press, Release, Modifiers

IDLlitWindow::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->IDLlitWindow::OnMouseDown, X, Y, ButtonMask, Modifiers, NumClicks

IDLlitWindow::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->IDLlitWindow::OnMouseMotion, X, Y, Modifiers

IDLlitWindow::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->IDLlitWindow::OnMouseUp, X, Y, ButtonMask

IDLlitWindow::OnMouseWheel - Handles notification (from the native window device) that a mouse wheel event has occurred.

Obj->IDLlitWindow::OnMouseWheel, X, Y, Delta, Modifiers

IDLlitWindow::OnKeyPress - Handles notification (from the native window device) that a keyboard event has occurred.

Obj->IDLlitWindow::OnKeyPress, Object, [ALL]
[.PROPERTY=value]

IDLlitWindow::OnKeyRelease - Handles notification (from the native window device) that a keyboard event has occurred.

Obj->IDLlitWindow::OnKeyRelease, Object, [ALL]
[.PROPERTY=value]

IDLlitWindow::OnResize - Handles notification (from the native window device) that a resize event has occurred.

Obj->IDLlitWindow::OnResize, Width, Height

IDLlitWindow::OnScroll - Handles notification (from the native window device) that a scrolling event has occurred.

Obj->IDLlitWindow::OnScroll, X, Y

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

Obj->IDLlitWindow::OnTimer

IDLlitWindow::OnWheel - Handles notification (from the native window device) that a mouse wheel event has occurred.

Obj->IDLlitWindow::OnWheel, X, Y, Delta, Modifiers

IDLlitWindow::Remove - Removes the given object(s) from the window.

Obj->IDLlitWindow::Remove, Object, [ALL]
[.POSITION=index]

IDLlitWindow::RemoveWindowEventObserver - Removes the given object(s) from the list of observers that are notified of events that occur within this window.

Obj->IDLlitWindow::RemoveWindowEventObserver, Objects

IDLlitWindow::SetCurrentZoom - Sets the current zoom factor for this window by changing its virtual dimensions.

Obj->IDLlitWindow::SetCurrentZoom, ZoomFactor, [RESET]

IDLlitWindow::SetEventMask - Enables the given events within this window.

Obj->IDLlitWindow::SetEventMask([EventMask]
[.BUTTON_EVENTS], [.KEYBOARD_EVENTS]
[.MOUSE_EVENTS], [.TIMER_EVENTS]
[.TRACKING_EVENTS], [.WHEEL_EVENTS])

IDLlitWindow::SetManipulatorManager - Sets the given IDLitManipulatorManager object as the current manager of this window’s manipulators.

Obj->IDLlitWindow::SetManipulatorManager, Manager

IDLlitWindow::SetProperty - Sets the value of an IDLitWindow property.

Obj->IDLlitWindow::SetProperty, PROPERTY=value

IDLlitWindow::SetTimerInterval - Specifies the number of seconds between timer events.

Obj->IDLlitWindow::SetTimerInterval, PROPERTY=value

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).

Obj->IDLlitWindow::ZoomIn

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).

Obj->IDLlitWindow::ZoomOut
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

null

IDLitWriter::GetFileExtensions - Is called by the system to retrieve the file extensions supported by this particular writer.

Result = Obj->[IDLWriter::]GetFileExtensions([COUNT=count])

IDLitWriter::GetFilename - Is called by the system to retrieve the current filename associated with this writer.

Result = Obj->[IDLWriter::]GetFilename()

IDLitWriter::GetProperty - Retrieves the value of an IDLitWriter property.

Obj->[IDLWriter::]GetProperty(, PROPERTY=variable)

IDLitWriter::Init - Initializes the IDLitWriter object

null

IDLitWriter::IsA - Is called by the system to determine if the given file is of the type supported by this file writer.

Result = Obj->[IDLWriter::]IsA(Filename)

IDLitWriter::SetData - Is called by the system to set the data for the current file.

Result = Obj->[IDLWriter::]SetData(Data)

IDLitWriter::SetFilename - Is called by the system to set the current filename associated with this writer.

Obj->[IDLWriter::]SetFilename, Filename

IDLitWriter::SetProperty - Sets the value of an IDLitWriter property

Obj->[IDLWriter::]SetProperty(), PROPERTY=value

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.

null

IDLJavaObject::GetProperty - Retrieves properties (known as data members in Java) from the Java object that underlies the IDLJavaObject.

Obj->[IDLJavaObject::]GetProperty(, PROPERTY=variable)

IDLJavaObject::Init - Instantiates the given Java object and establishes a link between the resulting IDL object with the underlying Java object.

null

IDLJavaObject::SetProperty - Sets properties (known as data members in Java) for the Java object that underlies an instance of IDLJavaObject.

Obj->[IDLJavaObject::]SetProperty(, PROPERTY=value)

IDLnetOGCWCS - Provides the ability to connect to a remote Open Geospatial Consortium Web Coverage Service server and return coverage offering briefs and unrendered data containing geographic data.

Properties: [ , AUTHENTICATION {Get, Init, Set}=value ], CALLBACK_DATA {Get, Init, Set}=variable ], CALLBACK_FUNCTION {Get, Init, Set}=string ], CAPABILITIES_FILENAME {Get, Init, Set}=string ], CONNECT_TIMEOUT {Get, Init, Set}=value ], CONTENT_TYPE {Get}=string ], COVERAGE_FILENAME {Get, Init, Set}=string ], DESCRIBE_COVERAGE_FILENAME {Get, Init, Set}=string ], DESCRIPTOR_FILENAME {Get, Init, Set}=string ], DESCRIPTOR_SCHEMA {Get, Init, Set}=string ], DESCRIPTION {Get, Init, Set}=string ], DESCRIBE_COVERAGE {Get, Init, Set}=string ], DESCRIBE_DESCRIP

null

IDLnetOGCWCS::Cleanup - Performs all cleanup on the object.

null

IDLnetOGCWCS::DescribeCoverage - Retrieves an XML file containing one or more descriptive coverage offerings from a remote OGC WCS server. Return value contains names of coverage offerings.

Result = Obj->[IDLnetOGCWCS::]DescribeCoverage (, COUNT=variable ), NAMES=value ]

null

IDLnetOGCWCS::GetCapabilities - Retrieves an XML file containing coverage offering briefs from a remote OGC WCS server.

Result = Obj->[IDLnetOGCWCS::]GetCapabilities(, FROM_FILE=value ]

null

IDLnetOGCWCS::GetCoverage - Retrieves a data file containing unrendered data from a remote OGC WCS server.

Result = Obj->[IDLnetOGCWCS::]GetCoverage (Coverage_Request)

null
IDLnetOGCWCS::GetCoverageOffering - Returns a complex structure containing details about a single coverage.

Result = Obj->IDLnetOGCWCS::GetCoverageOffering ([INDEX=value])

IDLnetOGCWCS::GetCoverageOfferingBriefs - Provides access to coverage offering briefs returned by GetCapabilities.

Result = Obj->IDLnetOGCWCS::GetCoverageOfferingBriefs ([COUNT=variable], [NUMBER=value], [INDEX=value])

IDLnetOGCWCS::GetProperty - Retrieves the value of a property or group of properties.

Obj->IDLnetOGCWCS::GetProperty [PROP=variable]

IDLnetOGCWCS::GetServiceSection - Provides a structure that accesses Service section values contained in the XML file returned by GetCapabilities.

Result = Obj->IDLnetOGCWCS::GetServiceSection()

IDLnetOGCWCS::Init - Initializes an OGC WCS object.

Obj = OBJ_NEW(IDLnetOGCWCS [PROP=value]) or Result = Obj->[IDLnetOGCWCS::Init(PROP=value)] (Only in a subclass' Init method)

IDLnetOGCWCS::ParseUrl - Given a valid URL, populated required properties with portions of the URL string.

Obj->IDLnetOGCWCS::ParseUrl, URL

IDLnetOGCWCS::SetProperty - Sets the value of a property or group of properties for the object.

Obj->IDLnetOGCWCS::SetProperty [PROP=value]

IDLnetOGCWMS - Provides a structure for the object.

[SSL_VERIFY_PEER [Get, Init, Set]=Boolean]
[TIMEOUT [Get, Init, Set]=value]
[URL_HOSTNAME [Get, Init, Set]=string]

IDLnetOGCWMS::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, Obj or Obj->IDLnetOGCWMS::Cleanup (Only in subclass' Cleanup method)

IDLnetOGCWMS::GetCapabilities - Retrieves an XML file containing layer information from a remote OGC WCS server.

Result = Obj->IDLnetOGCWMS::GetCapabilities ([DISABLE_INHERITANCE], [FROM_FILE=value], [], [SCHEMA_CHECK=value])

IDLnetOGCWMS::GetFeatureInfo - Retrieves a information about a feature on a map if supported by the WMS server and if the information exists.

Result = Obj->IDLnetOGCWMS::GetFeatureInfo (Feature_Info_Request)

IDLnetOGCWMS::GetLayers - Provides access to layer details returned by GetCapabilities.

Result = Obj->IDLnetOGCWMS::GetLayers ([COUNT=variable], [NUMBER=value], [INDEX=value])

IDLnetOGCWMS::GetMap - Retrieves one or more layers as a rendered map image.

Result = Obj->IDLnetOGCWMS::GetMap (Map_Request)

IDLnetOGCWMS::GetProperty - Retrieves the value of a property or group of properties.

Obj->IDLnetOGCWMS::GetProperty [PROP=value]

IDLnetOGCWMS::GetServiceSection - Provides a structure that accesses Service section values contained in the XML file returned by GetCapabilities.

Result = Obj->IDLnetOGCWMS::GetServiceSection()

IDLnetOGCWMS::Init - Initializes an OGC WMS object.

Obj = OBJ_NEW(IDLnetOGCWMS [PROP=value]) or Result = Obj->[IDLnetOGCWMS::Init(PROP=value)] (Only in a subclass' Init method)

IDLnetOGCWMS::ParseUrl - Given a valid URL, populated required properties with portions of the URL string.

Obj->IDLnetOGCWMS::ParseUrl, URL
IDLnetURL::Cleanup - Performs all cleanup operations on the object, and closes all open connections when the object is destroyed.

```
IDLnetURL::Cleanup (In a lifecycle method only.)
```

IDLnetURL::CloseConnections - Closes all the existing connections to remote HTTP and FTP servers.

```
IDLnetURL::CloseConnections
```

IDLnetURL::FtpCommand - Executes FTP commands, as well as deletes files, makes directories, deletes directories, changes directories, and so on, on the remote FTP server.

```
Result = Obj->[IDLnetURL::FtpCommand( command, [URL=string], [/FTP_ASCII], [/FTP_EXPLICIT_SSL])]
```

IDLnetURL::GetProperty - Returns information describing the properties of an IDLnetURL object in the supplied variable/key-value pairs.

```
Obj->[IDLnetURL::GetProperty, PROPERTYNAME=value]
```

IDLnetURL::Init - Initializes an IDLnetURL object.

```
obj = Obj->[IDLnetURL::Init]
```

IDLnetURL::Put - Sends a file or a buffer of data to a remote HTTP or FTP server.

```
Result = Obj->[IDLnetURL::Put( data, [URL=string], [/VERBOSE], [/BUFFER], [/POST])]
```

IDLnetURL::SetProperty - Sets the properties of an object that is open for writing.

```
Result = Obj->[IDLnetURL::SetProperty, PROPERTYNAME=value]
```

IDLsysMonitorInfo::Cleanup - Performs all cleanup on the object.

```
IDLsysMonitorInfo::Cleanup (Only in subclass' Cleanup method)
```

IDLsysMonitorInfo::GetNumberOfMonitors - Retrieves the number of monitors available on the system as an IDL integer.

```
Result = Obj->[IDLsysMonitorInfo::GetNumberOfMonitors()]
```
IDLsysMonitorInfo::GetPrimaryMonitorIndex - Retrieves the index of the monitor that is considered the system's primary monitor as an IDL integer.

Result = Obj->[IDLsysMonitorInfo::]GetPrimaryMonitorIndex()

IDLsysMonitorInfo::GetProperty - Retrieves the value of a property or group of properties.

Obj->[IDLsysMonitorInfo::]GetProperty

IDLsysMonitorInfo::GetRectangles - Retrieves monitor pixel coordinates and dimensions as a [4,n] IDL long array. There is one row in the array for each monitor.

Result = Obj->[IDLsysMonitorInfo::]GetRectangles([, /EXCLUDE_TASKBAR])

IDLsysMonitorInfo::GetResolutions - Retrieves monitor resolutions (pixel sizes) as a [2,n] IDL float array of x and y cm/pixel.

Result = Obj->[IDLsysMonitorInfo::]GetResolutions()

IDLsysMonitorInfo::Init - Initializes a monitor information object.

Obj = OBJ_NEW(IDLsysMonitorInfo [PROPERTY=value]) or
Result = Obj->[IDLsysMonitorInfo::]Init([PROPERTY=value]) (Only in a subclass' Init method)

IDLsysMonitorInfo::IsExtendedDesktop - Returns a Boolean value (0 or 1 as an IDL integer) that indicates if the display system is configured as an extended desktop across two or more monitors.

Result = Obj->[IDLsysMonitorInfo::]IsExtendedDesktop()

IDLsysMonitorInfo::SetProperty - Sets the value of a property or group of properties for the object.

Obj->[IDLsysMonitorInfo::]SetProperty[, PROPERTY=value]

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{Init}={0 | 1 | 2}] [, /CONSTRAIN{Init}] [, MOUSE{Init}={1 | 2 | 4}]

TrackBall::Init - Initializes the TrackBall object.

Obj = OBJ_NEW('TrackBall', Center, Radius [, PROPERTY=value]) or
Result = Obj->[TrackBall::]Init(Center, Radius [, PROPERTY=value])

TrackBall::Reset - Resets the state of the TrackBall object.

Obj->[TrackBall::]Reset, Center, Radius [, AXIS={0 | 1 | 2}] [, /CONSTRAIN] [, MOUSE={1 | 2 | 4}]

TrackBall::Update - Updates the state of the TrackBall object based on the information contained in the input widget event structure.

Result = Obj->[TrackBall::]Update(sEvent [, MOUSE={1 | 2 | 4}] [, TRANSFORM=variable] [, /TRANSFORM] )
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[subscript_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.

Conditional Statements

- `IF...THEN...ELSE` - Conditionally executes a statement or block of statements.

Loop Statements

- `FOR...DO` - Executes one or more statements repeatedly, while incrementing or decrementing a variable with each repetition, until a condition is met.

Jump Statements

- `BREAK` - Immediately exits from a loop (FOR, WHILE, REPEAT), CASE, or SWITCH statement without resorting to GOTO statements.
- `CONTINUE` - Immediately starts the next iteration of the enclosing FOR, WHILE, or REPEAT loop.
- `GOTO` - Transfers program control to point specified by `label`. 

GOTO
Functions and Procedures

**COMPIL** E/OPT - Gives IDL compiler information that changes the default rules for compiling functions or procedures.

```
COMPIL**E/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).

```
FORWAR**D_FUNCTION Name1, Name2, ..., Name n
```

**FUNCTION** - Defines a function.

```
FUNCTION Function_Name, parameter 1, ..., parameter n
```

**PRO** - Defines a procedure.

```
PRO Procedure_Name, argument 1, ..., argument n
```

**Procedure_Name** - Calls a procedure.

```
Procedure_Name, argument 1, ..., argument n
```

**Result = FUNCTION (arg 1, ..., arg n)** - Calls a function.

Variable Scope

**COMMON** - Creates a common block.

```
COMMON Block_Name, Variable1, ..., Variable n
```
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,..., FileN]

To compile from a temporary file: .COMPILE -f TempFile

.CONTINUE - Continues execution of a stopped program.

.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 FileN]

.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

.RETURN - Continues execution until RETURN statement.

.RETURN

.RNEW - Erases main program variables and then does .RUN.

.RNEW [File1,...,FileN]

To save listing in a file: RNEW -L ListFile.lis File1

[. File2,..., FileN]

To display listing on screen: RNEW -T File1 [, File2,..., FileN]

.RUN - Compiles and executes IDL commands from files or keyboard.

.RUN [File1,..., FileN]

To save listing in a file: RUN -L ListFile.lis File1

[. File2,..., FileN]

To display listing on screen: RUN -T File1 [, File2,..., FileN]

.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
## 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>
<tr>
<td>Apostrophe (’)</td>
<td>• Delimits string constants</td>
</tr>
<tr>
<td></td>
<td>• Indicates part of octal or hex constant</td>
</tr>
<tr>
<td>Asterisk (*)</td>
<td>• Multiplication operator</td>
</tr>
<tr>
<td></td>
<td>• Array subscript range</td>
</tr>
<tr>
<td></td>
<td>• Pointer dereference (if in front of a valid pointer)</td>
</tr>
<tr>
<td>At Sign (@)</td>
<td>• Include character: Used at beginning of a line to cause the IDL compiler</td>
</tr>
<tr>
<td></td>
<td>to substitute the contents of the file whose name appears after the @</td>
</tr>
<tr>
<td></td>
<td>symbol for the line.</td>
</tr>
<tr>
<td></td>
<td>• In interactive mode, @ is used to execute a batch file.</td>
</tr>
<tr>
<td>Colon (:</td>
<td>• Ends label identifiers</td>
</tr>
<tr>
<td></td>
<td>• Separates start and end subscript ranges</td>
</tr>
<tr>
<td>Dollar Sign ($)</td>
<td>• Continue current command on the next line</td>
</tr>
<tr>
<td></td>
<td>• Issue operating system command if entered on a line by itself</td>
</tr>
<tr>
<td>Exclamation Point (!)</td>
<td>• First character of system variable names and font-positioning commands</td>
</tr>
<tr>
<td>Period (.)</td>
<td>• First character of executive commands</td>
</tr>
<tr>
<td></td>
<td>• Indicates floating-point numbers</td>
</tr>
<tr>
<td></td>
<td>• Indicates fields in a structure, such as in mystructure.field1</td>
</tr>
<tr>
<td>Question Mark (?)</td>
<td>• Invokes online help when entered at the IDL command line</td>
</tr>
<tr>
<td></td>
<td>• Part of the ?: ternary operator used in conditional expressions</td>
</tr>
<tr>
<td>Semicolon (;)</td>
<td>• First character of comment field. Everything after the semicolon is</td>
</tr>
<tr>
<td></td>
<td>ignored by IDL. Semicolon can be used as the first character or after an</td>
</tr>
<tr>
<td></td>
<td>an IDL command:</td>
</tr>
<tr>
<td></td>
<td>; This is a comment</td>
</tr>
<tr>
<td></td>
<td>COUNT = 5 ; Set variable COUNT to 5</td>
</tr>
</tbody>
</table>
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:*]</td>
<td>Elements $i$ through the end 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[*]</td>
<td>Column $i$ of a two-dimensional array.</td>
</tr>
<tr>
<td>Array[*]</td>
<td>The $j$th row of a two-dimensional array.</td>
</tr>
<tr>
<td>Array[ij: m:n]</td>
<td>Subarray of columns $i$ through $j$, rows $m$ through $n$.</td>
</tr>
<tr>
<td>Array[i:j:k]</td>
<td>A range of subscripts, written $[k_0:k_1:k_2]$, denoting every $k_2$th element within the range of subscripts $k_0$ through $k_1$ ($k_0$ must not be greater than $k_1$). $k_2$ is referred to as the subscript stride.</td>
</tr>
<tr>
<td>Array[i:*:k]</td>
<td>Every $k_2$th element from a given element to the last element of the dimension, written as $[k_0:*:k_2]$. $k_2$ is referred to as the subscript stride.</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
-> Object method invocation
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></td>
<td>~ (logical 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>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 $\pi$ ($\pi$). |
| IDTOR | Degrees to radians, $\pi/180 \approx 0.01745$. |
| !MAP | Read-only system variable used by MAP_SET. |
| !PI | Single-precision $\pi$ ($\pi$). |
| !RADEG | Radians to degrees, $180/\pi \approx 57.2958$. |
| !VALUES | Single- and double-precision NaN and Infinity values. |

Graphics System Variables

| ID | Information about current graphics device. |
| **Fields**: | FILL_DIST - line interval, in device coordinates |
| | FLARGS - longword of flags |
| N_COLORS | number of simultaneously available colors |
| NAME | string containing name of device |
| ORIGIN | pan/scroll offset (pan, scroll) |
| TABLE_SIZE | number of color table indices |
| UNIT | logical number of file open for output |
| WINDOW | index of currently open window |
| X_CH_SIZE, Y_CHAR_SIZE | width/height of rectangle that encloses the average character in current font, in device units (usually pixels) |
| X_PX_CM, Y_PX_CM | approx. number of pixels/cm |
| X_SIZE, Y_SIZE | total size of the display or window, in device units |
| X_VSIZE, Y_VSIZE | size of visible area of display or window |
| 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 ([$x_0$, $y_0$, $z_0$], [$x_f$, $y_f$, $z_f$]) |
| 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 134) |
| 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 ($x_0$, $y_0$, $x_1$, $y_1$) |
| PSYM | plotting symbol index (see “Plotting Symbols” on page 134) |
| REGION | normalized coords of plot region ($x_0$, $y_0$, $x_1$, $y_1$) |
| 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) |
| IX, IY, IZ | Axis structures for X, Y, and Z axes. |
| **Fields**: | CHARSIZE | character size of annotation when Hershey fonts are selected |
| CRANGE | output axis range |
| GRIDSTYLE | linestyle for tick marks/grids (see “Line Styles” on page 134) |
| MARGIN | 2-element array specifying plot window margins, in units of char size ([left or bottom, right or top]) |
| MINOR | number of minor tick marks |
| OMARGIN | 2-element array specifying plot window outer margins, in units of char size ([left or bottom, right or top]) |
| RANGE | 2-element vector specifying input axis range (min, max) |
| REGION | normalized coords of region (2-element floating-point array) |
| T | homogeneous 4 x 4 transformation matrix |
| S | 2-element array specifying scaling factors for conversion between data and normalized coords |
| STYLE | style of the axis encoded as bits in a longword. 1=exact, 2=extend, 4=no axis, 8=no box, 16=inhibit setting Y axis min to 0 when data are all greater than 0 (add values together for multiple effects) |
| THICK | thickness of axis line |
| TICKFORMAT | format string or string containing name of function that returns format string used to format axis tick mark labels |
**Alphabetical List of IDL Routines**

**IX, IY, IZ - continued**

- TICKINTERVAL - indicates the interval between major tick marks for the first axis level
- TICKLAYOUT - indicates the tick layout style to be used to draw each level of the axis
- TICKLEN - tick mark length, in normal coords
- TICKNAME - annotation for each tick (string array)
- TICKS - number of major tick intervals
- TICKUNITS - indicates the units to be used for axis tick labeling
- TICKV - data values for each tick mark (array)
- TITLE - string containing axis title
- TYPE - type of axis (0 for linear, 1 for logarithmic)
- WINDOW - normalized coords of axis end points (2-element floating-point array)

**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 (read-only).
  - CODE - long-integer containing error code of IDL-generated component of last error message (read-only).
  - 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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>!CPU</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>!DIR</strong></td>
<td>Location of the main IDL directory.</td>
</tr>
<tr>
<td><strong>!DLM_PATH</strong></td>
<td>Indicates where IDL looks for Dynamically Loadable Modules when started. Read-only.</td>
</tr>
<tr>
<td><strong>!EDIT_INPUT</strong></td>
<td>Enables/disables keyboard line editing.</td>
</tr>
<tr>
<td><strong>!HELP_PATH</strong></td>
<td>Lists directories IDL will search for online help files.</td>
</tr>
<tr>
<td><strong>!JOURNAL</strong></td>
<td>Logical unit number of journal output, or 0.</td>
</tr>
<tr>
<td><strong>!MORE</strong></td>
<td>Set to 0 to prevent paginating help text.</td>
</tr>
<tr>
<td><strong>!MAKE_DLL</strong></td>
<td>Used to configure how IDL uses the CALL_EXTERNAL, DLMs, and LINKIMAGE for the current platform.</td>
</tr>
<tr>
<td><strong>!PATH</strong></td>
<td>Search path for IDL routines. UNIX: colon-separated list of directories. Windows: semicolon-separated list of directories.</td>
</tr>
<tr>
<td><strong>!PROMPT</strong></td>
<td>String to be used for IDL prompt.</td>
</tr>
<tr>
<td><strong>!QUIET</strong></td>
<td>Suppresses informational messages if set to nonzero.</td>
</tr>
<tr>
<td><strong>!VERSION</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

**IDL Quick Reference**

132
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 labels for tickmark 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 marks.
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 OPLOT, 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 OPLOT, 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” on page 136
- “EOS Routines” on page 137
- “HDF Routines” on page 142
- “HDF5 Routines” on page 147
- “NetCDF Routines” on page 151
CDF Routines

CDF_ATTCREATE - Creates a new attribute.
Result = CDF_ATTCREATE( Id, Attribute_Name
[. /GLOBAL_SCOPE] [. /VARIABLE_SCOPE] )

CDF_ATTDELETE - Deletes attribute from specified CDF file.
CDF_ATTDDELETE, Id, Attribute [, EntryNum]
[. /VARIABLE]

CDF_ATTEXIST - Determines whether specified attribute exists.
Result = CDF_ATTEXIST( Id, Attribute [, EntryNum]
[. /VARIABLE] )

CDF_ATTGET - Reads an attribute entry from a CDF file.
CDF_ATTGET, Id, Attribute, EntryNum, Value
[. CDF_TYPE= variable [, /VARIABLE]]

CDF_ATTINQ - Obtains information about specified attribute.
CDF_ATTINQ, 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
[. /VARIABLE]

CDF_ATTRENAME - Renames an existing attribute.
CDF_ATTRENAME, 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}]

CDF_ATTRENAME, Id, OldAttr, NewName

CDF_ATTPUT - Writes an attribute entry to a CDF file.
CDF_ATTPUT, Id, Attribute, EntryNum, Value
[. /VARIABLE]

CDF_ATTNUM - Returns an attribute number.
Result = CDF_ATTNUM( Id, Attribute_Name )

CDF_ATTNQ - Obtains information about specified attribute.
CDF_ATTNQ, Id, Attribute, Name, Scope, MaxEntry
[. MaxZEntry]

CDF_ATTNAME - Renames an existing attribute.
CDF_ATTNAME, 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]
[. /VARIABLE]

CDF_ATTRENAME - Renames an existing attribute.
CDF_ATTRENAME, 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]
[. /VARIABLE]

CDF_ATTRENAME - Renames an existing attribute.
CDF_ATTRENAME, 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]
[. /VARIABLE]

CDF_ATTRENAME - Renames an existing attribute.
CDF_ATTRENAME, 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]
[. /VARIABLE]
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_ENCODE_EPOCH16 - Encodes CDF_EPOCH16 variable into a string.
Result = CDF_ENCODE_EPOCH16(Epoch16
[ , EPOCH=[0 | 1 | 2 | 3] ]

CDF_EPOCH - Computes/breaks down CDF_EPOCH values.
CDF_EPOCH, Epoch, Year [, , Month, Day, Hour, Minute, Second, Milli] [, /COMPUTE_EPOCH]
[ , /BREAKDOWN_EPOCH]

CDF_EPOCH_COMPARE - Compares two epoch (date and time) values and returns an integer value of 1, 0, or -1.
Result = CDF_EPOCH_COMPARE(Epoch1, epoch2

CDF_EPOCH_DIFF - Compares two epoch (date and time) values and returns the difference in milliseconds (default) or microseconds.
Result = CDF_EPOCH_DIFF(Epoch1, epoch2
[ , /MILLI_SECONDS] [ , /MICRO_SECONDS])

CDF_EPOCH16 - Computes/breaks down CDF_EPOCH16 values.
CDF_EPOCH16, Epoch, Year [, , Month, Day, Hour, Minute, Second, Milli, Micro, Nano, Pico] [, /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_OPEN - Opens an existing Common Data Format file.
Result = CDF_OPEN(Filename)

CDFPARSE_EPOCH - Parses input string into a double precision value properly formatted for use as CDF_EPOCH variable.
Result = CDFPARSE_EPOCH(Epoch_string)

CDFPARSE_EPOCH16 - Parses input string into a double precision value properly formatted for use as CDF_EPOCH16 variable.
Result = CDFPARSE_EPOCH16(Epoch_string)

EOS_Routines

EOS_EH_CONVANG - Converts angles between decimal degrees, radians, and packed degrees-minutes-seconds.
Result = EOS_EH_CONVANG(inAngle, code)

Scientific Data Formats
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 ID 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(gridID, fid, gridname)

EOS_GD_ATTRINFO - Returns number type and number of ele-
   ments (count) of a grid attribute.
   Result = EOS_GD_ATTRINFO(gridID, attrname, numbertype, count)

EOS_GD_CLOSE - Closes the HDF grid file.
   Result = EOS_GD_CLOSE(gridID, fid)

EOS_GD_COMPINFO - Returns the compression code and com-
   pression parameters for a given field.
   Result = EOS_GD_COMPINFO(gridID, fieldname, compcode, compparm)

EOS_GD_CREATE - Creates a grid within the file.
   Result = EOS_GD_CREATE(fid, 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_DEFCOMP - Sets the HDF field compression for subse-
   quent grid field definitions.
   Result = EOS_GD_DEFCOMP(gridID, compcode [
   [, compparm] )

EOS_GD_DEFDM - Defines dimensions used by field definition
   routines to establish size of the field.
   Result = EOS_GD_DEFDM(gridID, dimname, dim)

EOS_GD_DEFFIELD - Defines data fields to be stored in the grid.
   Result = EOS_GD_DEFFIELD(gridID, fieldname, dimlist, numbertype [, /MERGE] )

EOS_GD_DEFORIGIN - Defines the origin of the grid data.
   Result = EOS_GD_DEFORIGIN(gridID, origincode)

EOS_GD_DEFPIXREG - Defines whether the pixel center or pixel
   corner is used when requesting the location of a given pixel.
   Result = EOS_GD_DEFPIXREG(gridID, pixreg)

EOS_GD_DEFPROJ - Defines the GCTP projection and projection
   parameters of the grid.
   Result = EOS_GD_DEFPROJ(gridID, projcode, zonecode, spherecode, projparm)

EOS_GD_DEFTIME - Defines the tiling dimensions for fields
   defined following this function call.
   Result = EOS_GD_DEFTIME(gridID, tilecode [, , tilerank, tiledims] )

EOS_GD_DEFTIMEPERIOD - Defines a time period for a grid.
   Result = EOS_GD_DEFTIMEPERIOD(gridID, periodID, starttime, stoptime)

EOS_GD_DEFVRTREGION - Subsets on a monotonic field or con-
   tinuous elements of a dimension.
   Result = EOS_GD_DEFVRTREGION(gridID, regionID, vertObj, range)

EOS_GD_DETACH - Detaches from grid interface.
   Result = EOS_GD_DETACH(gridID)

EOS_GD_DIMINFO - Retrieves the size of the specified dimension.
   Result = EOS_GD_DIMINFO(gridID, dimname)

EOS_GD_DUPREGION - Copies information stored in current
   region or period to a new region or period.
   Result = EOS_GD_DUPREGION(regionID)

EOS_GD_EXTRACTREGION - Reads data into the data buffer
   from a subsetted region as defined by
   EOS_GD_DEFBOXREGION.
   Result = EOS_GD_EXTRACTREGION(gridID, regionID, fieldname, buffer)

EOS_GD_FIELDINFO - Retrieves information on a specific data
   field.
   Result = EOS_GD_FIELDINFO(gridID, fieldname, rank, dims, numbertype, dimlist)

EOS_GD_GETFILLVALUE - Retrieves fill value for specified
   field.
   Result = EOS_GD_GETFILLVALUE(gridID, fieldname, fillvalue)

EOS_GD_GETPIXELS - Returns the pixel rows and columns for
   specified longitude/latitude pairs.
   Result = EOS_GD_GETPIXELS(gridID, nLonLat, lonVal, latVal, 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_INQA TRS - Retrieves information about attributes
   defined in grid.
   Result = EOS_GD_INQATTRS(gridID, attrlist [, , LENGTH (OUT)=value] )

EOS_GD_INQDIMS - Retrieves information about dimensions
   defined in grid.
   Result = EOS_GD_INQDIMS(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, fieldname, 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 pixel registration code, zone code, spheroid code, and projection parameters of the grid.
Result = EOS_GD_PIXREGINFO(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, fieldname, 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, upleftpnt, loowrighpnt)

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_WRITEFIELDMETA - Writes field metadata for a grid field not defined by the Grid API.
Result = EOS_GD_WRITEFIELDMETA(gridID, fieldname, dimlist, numbertype)

EOS_GD_WRITETILE - Writes a single tile of data to a field.
Result = EOS_GD_WRITETILE(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=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, outRecs, outINRec, outRec)

EOS_PT_INQATTRS - Returns attribute list as a comma-separated string.
Result = EOS_PT_INQATTRS( pointID, attriblist [, LENGTH=value] )

EOS_PT_INQPOINT - Retrieves number and names of points defined in HDF-EOS file.
Result = EOS_PT_INQPOINT( filename, pointlist [, LENGTH=variable] )

EOS_PT_LEVELINDEX - Returns the level index for a given level.
Result = EOS_PT_LEVELINDEX( pointID, levelID )

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 [, LENGTH=bytes] )

EOS_PT_NLEVELS - Returns the number of levels in a point.
Result = EOS_PT_NLEVELS(pointID)

EOS_PT_NRECS - Returns the number of records in a given level.
Result = EOS_PT_NRECS( pointID, level )

EOS_PT_OPEN - Creates a new file or opens an existing one.

EOS_PT_PERIODINFO - Returns information about a subsetted time period for a given fieldlist.
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, level, nrec, recs )

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, databuf)

EOS_PT_READLEVEL - Reads data from the specified fields and records of a single level in a point.
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, fieldlist, size )

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, size )

EOS_PT_WRITEATTR - Writes an attribute.
Result = EOS_PT_WRITEATTR( pointID, attrname, databuf [, 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(filename, swathname)

EOS_SW_DEFBOXREGION - Defines a longitude-latitude box region for a swath.
   Result = EOS_SW_DEFBOXREGION(swathID, cornerlon, cornerlat, mode)

EOS_SW_DEFDEFGEOFIELD - Defines geolocation fields to be stored in the swath.
   Result = EOS_SW_DEFDEFGEOFIELD(swathID, fieldname, geodim, datadim, offset, increment)

EOS_SW_DEFDEFDATAFIELD - Defines dimensions that are used by the field.
   Result = EOS_SW_DEFDEFDATAFIELD(swathID, fieldname, dimlist, numbertype [/.MERGE])

EOS_SW_DEFDEFDIM - Defines dimensions that are used by the field.
   Result = EOS_SW_DEFDEFDIM(swathID, fieldname, dim)

EOS_SW_DEFDEFDIMMAP - Defines monotonic mapping between the geolocation and data dimensions.
   Result = EOS_SW_DEFDEFDIMMAP(swathID, geodim, datadim, offset, increment)

EOS_SW_DEFDEFGEOFIELD - Defines geolocation fields to be stored in the swath.
   Result = EOS_SW_DEFDEFGEOFIELD(swathID, fieldname, geodim, datadim, numbertype [/.MERGE])

EOS_SW_DEFDEFDIMMAP - Defines mapping between a geolocation.
   Result = EOS_SW_DEFDEFDIMMAP(swathID, geodim, datadim, index)

EOS_SW_DEFDEFTIMEPERIOD - Defines a time period for a swath.
   Result = EOS_SW_DEFDEFTIMEPERIOD(swathID, starttime, stoptime, mode)

EOS_SW_DEFDIMINFO - Retrieves the size of the specified dimension.
   Result = EOS_SW_DEFDIMINFO(swathID, dimname)

EOS_SW_DUPREGION - Copies information stored in a current.
   Result = EOS_SW_DUPREGION(swathID, regionID)

EOS_SW_EXTRACTPERIOD - Reads data into the data buffer.
   Result = EOS_SW_EXTRACTPERIOD(swathID, periodID, fieldname, external_mode, buffer)

EOS_SW_EXTRACTREGION - Reads data into the data buffer.
   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.
   Result = EOS_SW_IDXMAPINFO(swathID, geodim, datadim, index)

EOS_SW_INQATRERS - Retrieves information about attributes defined in swath.
   Result = EOS_SW_INQATRERS(swathID, attrlist[, LENGTH (OUT)=value])

EOS_SW_INQDIMS - Retrieves information about all of the dimensions defined in swath.
   Result = EOS_SW_INQDIMS(swathID, dimname, dim)

EOS_SW_INQGEOFIELDS - Retrieves information about all of the geolocation fields defined in swath.
   Result = EOS_SW_INQGEOFIELDS(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_INQSATH - Retrieves number and names of swaths defined in HDF-EOS file.
   Result = EOS_SW_INQSATH(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_WRITEFIELD** - Writes data to a swath field.
Result = EOS_SW_WRITEFIELD(swathID, fieldname, 
data[, STRIDE=array][, START=arry] 
[. EDGE=arry])

**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)

**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[, EDGE(array)[, START(array] 
[. STRIDE=arry]])

**EOS_SW_WRITEGEOMETAL** - Writes field metadata for an existing geolocation field.
Result = EOS_SW_WRITEGEOMETAL(swathID, 
fieldname, dimlist, numbertype)

**HDF Routines**

**HDF_AN_CREATE** - Creates HDF AN annotation.
Result = HDF_AN_CREATE(swathID, attrname, 
result)

**HDF_AN_CREATEF** - Creates file annotation.
Result = HDF_AN_CREATEF(swathID, type, value)

**HDF_AN_WRITEANN** - Writes annotation text.
Result = HDF_AN_WRITEANN(swathID, attrname, 
result)

**HDF_AN_READANN** - Reads specified annotation.
Result = HDF_AN_READANN(swathID, attrname, 
result)

**HDF_AN_NUMANN** - Returns total number of annotations of a given type.
Result = HDF_AN_NUMANN(swathID, attrname, 
result)

**HDF_AN_ID2TAGREF** - Retrieves HDF tag and reference number pair of annotation.
Result = HDF_AN_ID2TAGREF(swathID, attrname, 
result)

**HDF_AN_NUMANN** - Returns total number of annotations of a given type.
Result = HDF_AN_NUMANN(swathID, attrname, 
result)

**HDF_AN_READANN** - Reads specified annotation.
Result = HDF_AN_READANN(swathID, attrname, 
result)

**HDF_AN_SELECT** - Obtains identifier of specified annotation.
Result = HDF_AN_SELECT(swathID, attrname, 
result)

**HDF_AN_START** - Initializes interface for specified file.
Result = HDF_AN_START(swathID, attrname, 
result)

**HDF_AN_TAG2ATYPE** - Returns annotation type of corresponding HDF tag.
Result = HDF_AN_TAG2ATYPE(swathID, attrname, 
result)

**HDF_AN_TAGREF2ID** - Returns ID of annotation with given tag.
Result = HDF_AN_TAGREF2ID(swathID, attrname, 
result)

**HDF_AN_WRITEANN** - Writes annotation text.
Result = HDF_AN_WRITEANN(swathID, attrname, 
result)

**HDF_BROWSER** - See "HDF_BROWSER" on page 44.

**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 ] [, /PLANES]

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 an HDF file.
HDF_DFAN_GETDESC, Filename, Tag, Ref, Description

HDF_DFAN_GETFDS - Retrieves list of reference numbers and labels for given tag.
Result = HDF_DFAN_GETFDS(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_PUTPAL - Appends palette to a HDF file.
HDF_DFP_PUTPAL, Filename, Palette [, /DELETE] 
[, /OVERWRITE]

HDF_DFP_READREF - Sets reference number of the palette.
HDF_DFP_READREF, Filename, Refno

HDF_DFP_RESTART - Causes next call to HDF_DFR8_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 
[, /FORCE_BASELINE{useful only if QUALITY<25}] 
[, /JPEG | /RLE] [, /IMCOMP] [, PALETTE=vector or array] [, QUALITY=value]

HDF_DFR8_GETIMAGE - Retrieves image, palette from HDF file.
HDF_DFR8_GETIMAGE, Filename, Image [, Palette]

HDF_DFR8_GETINFO - Retrieves information about the current 8-bit HDF image.
HDF_DFR8_GETINFO, Filename, Width, Height, 
Has_Palette

HDF_DFR8_LASTREF - Returns reference number of the most recently read or written 8-bit image in HDF file.
Result = HDF_DFR8_LASTREF( )

HDF_DFR8_NIMAGES - Returns number of 8-bit images in specified HDF file.
Result = HDF_DFR8_NIMAGES(Filename)
**HDF_DFR8_PUTIMAGE** - Writes 8-bit raster image as first image in HDF file.

HDF_DFR8_PUTIMAGE, Filename, Image

\[/, /FORCE_BASELINE\{useful only if QUALITY<25\}\]
\[/, /IMCOMP\] \{PALETTE=vector or array\} \[/, JPEG\] \{, /RLLE\} \{, QUALITY=value\}

**HDF_DFR8_RESTART** - Sets reference number of image to be read from a HDF file by the next call to HDF_DFR8_GETIMAGE.

HDF_DFR8_READREF, Filename, Refno

**HDF_DFR8_ERASE** - Causes next call to HDF_DFR8_GETIMAGE to read from first image in HDF file.

HDF_DFR8_RESTART

**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_ATTRINFO** - Retrieves information about specified HDF data object.

Result = HDF_GR_ATTRINFO(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)

**HDF_GR_END** - Terminates specified HDF GR interface session.

HDF_GR_END, gr_id

**HDF_GR_ENDACCESS** - Terminates access to specified raster image.

HDF_GR_ENDACCESS, ri_id

**HDF_GR_FILEINFO** - Retrieves number of raster images and global attributes for the specified HDF GR interface.

Result = HDF_GR_FILEINFO(gr_id, n_images, n_file_atts)

**HDF_GR_FINDATTR** - Finds index of HDF data object's attribute given its attribute name.

Result = HDF_GR_FINDATTR(obj_id, attr_name)

**HDF_GR_GETATTR** - Obtains all values of HDF GR attribute.

Result = HDF_GR_GETATTR(obj_id, attr_index, values)

**HDF_GR_GETCHUNKINFO** - Retrieves chunking information about HDF GR raster image.

Result = HDF_GR_GETCHUNKINFO(ri_id, dim_length, flag)

**HDF_GR_GETIMINFO** - Retrieves general information about HDF GR raster image.

Result = HDF_GR_GETIMINFO(ri_id, gr_name, ncomp, data_type, interlace_mode, dim_sizes, num_atts)

**HDF_GR_GETLUTID** - Gets identifier of HDF GR palette.

Result = HDF_GR_GETLUTID(ri_id, pal_index)

**HDF_GR_GETLUTINFO** - Retrieves information about a palette.

Result = HDF_GR_GETLUTINFO(pal_id, ncomp, data_type, interlace_mode, num_entries)

**HDF_GR_IDTOREF** - Returns HDF reference number of specified raster image.

Result = HDF_GR_IDTOREF(ri_id)

**HDF_GR_LUTTORE** - Returns HDF reference number of the specified palette.

Result = HDF_GR_LUTTORE(pal_id)

**HDF_GR_NAMETOINDEX** - Returns index of HDF data object's attribute given its name.

Result = HDF_GR_NAMETOINDEX(gr_id, gr_name)

**HDF_GR_READIMAGE** - Reads subsample of raster image.

Result = HDF_GR_READIMAGE(ri_id, data

\[, EDGE=\{array\}\] \[/, INTERLACE\] \[, START=\{array\}\]
\[, STRIDE=\{array\}\]

**HDF_GR_READLUT** - Reads specified palette.

Result = HDF_GR_READLUT(pal_id, pal_data

\[, /INTERLACE\])

**HDF_GR_REFTOINDEX** - Returns index of specified raster image.

Result = HDF_GR_REFTOINDEX(gr_id, gr_ref)

**HDF_GR_SELECT** - Obtains identifier of specified raster image.

Result = HDF_GR_SELECT(gr_id, index)

**HDF_GR_SETATTR** - Attaches attribute to specified object.

Result = HDF_GR_SETATTR(obj_id, attr_name, data_type, count, values)

**HDF_GR_SETCOMPRESS** - Specifies whether specified raster image will be stored in compressed format.

Result = HDF_GR_SETCOMPRESS(ri_id, comp_type, comp_prm)

**HDF_GR_SETCHUNK** - Makes specified raster image a chunked raster image.

Result = HDF_GR_SETCHUNK(ri_id, dim_length, comp_type, comp_prm)

**HDF_GR_SETCHUNKCACHE** - Sets maximum number of chunks to be cached.

Result = HDF_GR_SETCHUNKCACHE(ri_id, maxcache, flags)

**HDF_GR_SETEXTERNALFILE** - Specifies that raster image will be written to external file.

Result = HDF_GR_SETEXTERNALFILE(ri_id, filename, offset)

**HDF_GR_START** - Initializes interface for the specified file.

Result = HDF_GR_START(file_id)
HDF SD_ATTRIB - Writes attributes to an open HDF SD dataset.

HDF SD_ATTRIBSET - Writes attributes to an open HDF SD dataset.

HDF SD_ATTRIBSET, S_ID, Attr_Name, Values [, Count]
[ , /BYTE] [ , /DFNT_CHAR] [ , /DFNT_FLOAT32]
[ , /DFNT_FLOAT64] [ , /DFNT_INT8] [ , /DFNT_INT16]
[ , /DFNT_INT32] [ , /DFNT_UINT8] [ , /DFNT_UINT16]
[ , /DFNT_UINT32] [ , /DOUBLE] [ , /FLOAT] [ , /INT]
[ , /LONG] [ , /SHORT] [ , /STRING]

HDF SD_CREATE - Creates and defines a Scientific Dataset for an
HDF file.

HDF SD_CREATE, SD_ID, Name, Dims
[ , /BYTE] [ , /DFNT_CHAR] [ , /DFNT_FLOAT32]
[ , /DFNT_FLOAT64] [ , /DFNT_INT8] [ , /DFNT_INT16]
[ , /DFNT_INT32] [ , /DFNT_UINT8] [ , /DFNT_UINT16]
[ , /DFNT_UINT32] [ , /DOUBLE] [ , /FLOAT]
[ , /STRING]

HDF SD_DIMGET - Retrieves info. about SD dataset dimension.

HDF SD_DIMGET, Dim_ID [, COUNT]
[ , COMPATIBILITY=variable] [ , /FORMAT]
[ , /LABEL] [ , /NAME] [ , /NATTR] [ , /SCALE]
[ , /TYPE] [ , /UNIT]

HDF SD_DIMGETID - Returns dimension ID given a dataset
“SDS_ID” and dimension number.

HDF SD_DIMGETID(SDS_ID, Dimension_Number)

HDF SD_DIMSET - Sets scale and data strings for SD dimension.

HDF SD_DIMSET, Dim_ID [, /BW_INCOMP]
[ , /FORMAT=string] [ , /LABEL=string] [ , /NAME=string]
[ , /SCALE=vector] [ , /UNIT=string]

HDF SD_END - Closes SD interface to an HDF file.

HDF SD_END, SD_ID

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_GETINFO, SDS_ID, CALDATA=variable
[ , COORDSYS=variable] [ , DIMS=variable]
[ , FILL=variable] [ , FORMAT=variable]
[ , /DFNT_UINT8] [ , /LABEL=variable]
[ , /NAME=variable] [ , /NATTR=variable]
[ , /NDIMS=variable] [ , /NOREVERSE]
[ , RANGE=variable] [ , /TYPE=variable]
[ , /UNIT=variable]
HDF_SD_IDTOREF - Converts SD data set ID into SD data set reference number.
Result = HDF_SD_IDTOREF(SDS_ID)

HDF_SD_ISOORDVAR - Determines whether supplied dataset ID represents NetCDF "coordinate" variable.
Result = HDF_SD_ISOORDVAR(SDS_ID)

HDF_SD_NAMETOINDEX - Returns SD dataset index given its name and SD interface ID.
Result = HDF_SD_NAMETOINDEX(SD_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_SETINFO - Sets information about SD dataset.
HDF_SD_SETINFO, SDS_ID, [ . FILL=value ]
[. FORMAT=string ]
[. LABEL=string ]
[. RANGE=[min, max] ]
[. UNIT=string ]
[. COORDSYS=string ]
[. CALCDATA=structure ]

HDF_SD_START - Opens or creates HDF file and initializes SD interface.
Result = HDF_SD_START(Filename [, .READ [, .RDWR [, .CREATE ] ] ] )

HDF_UNPACKDATA - Unpacks array of byte data into IDL variables.
HDF_UNPACKDATA, packeddata, data1 [, data2
[. data3 ]
[. data4 [, data5 [, data6 [, data7 [, data8]]]]]]]
[. HDF_ORDER=arrary ]
[. HDF_TYPE=arrary ]
[. NREC=records ]

HDF_VD_ATTACH - Accesses a VData with the given ID.
Result = HDF_VD_ATTACH( FileHandle, VDataId
[. /READ ] [. /WRITE ] )

HDF_VD_ATTRFIND - Returns an attribute's index number given the name of an attribute.
Result = HDF_VD_ATTRFIND(VData, FieldID, Name)

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_ATTRSET, VData, FieldID, Attr_Name, Values
[. Count=variable ]
[. /BYTE ]
[. /DFNT_CHAR8 ]
[. /DFNT_FLOAT32 ]
[. /DFNT_FLOAT64 ]
[. /DFNT_INT8 ]
[. /DFNT_INT16 ]
[. /DFNT_INT32 ]
[. /DFNT_UINT8 ]
[. /DFNT_UINT16 ]
[. /DFNT_UINT32 ]
[. /DOUBLE ]
[. /FLOAT ]
[. /INT ]
[. /LONG ]
[. /SHORT ]
[. /STRING ]
[. /UINT ]
[. /ULONG ]

HDF_VD_DETACH - Called when done accessing a VData.
HDF_VD_DETACH, VData

HDF_VD_FDEFINE - Adds new field specification for VData.

HDF_VD_FEXIST - Returns true if specified fields exist in HDF file.
Result = HDF_VD_FEXIST(VData, Fieldnames)

HDF_VD_FIND - Returns reference number of specified VData.
Result = 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.
Result = HDF_VD_GETID(FileHandle, VDataId)

HDF_VD_GETINFO - Returns information about each VData field.
HDF_VD_GETINFO, VData, Index [, .NAME=variable ]
[. ORDER=value ]
[. 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.
Result = HDF_VD_ISATTR(VData)

HDF_VD_ISVD - Returns True (1) if an object is a VData.
Result = HDF_VD_ISVD(VGroup, Id)

HDF_VD_ISVG - Returns True (1) if an object is a VGroup.
Result = HDF_VD_ISVG(VGroup, Id)

HDF_VD_LONE - Returns array containing all VDatas that are not contained in another VData.
Result = HDF_VD_LONE( FileHandle [, .MAXSIZE=value ] )

HDF_VD_NATTRS - Returns the number of attributes associated with the specified VData.
Result = HDF_VD_NATTRS( VData, FieldId )

HDF_SD_IDTOREF - Converts SD data set ID into SD data set reference number.
Result = HDF_SD_IDTOREF(SDS_ID)

HDF_SD_ISOORDVAR - Determines whether supplied dataset ID represents NetCDF "coordinate" variable.
Result = HDF_SD_ISOORDVAR(SDS_ID)

HDF_SD_NAMETOINDEX - Returns SD dataset index given its name and SD interface ID.
Result = HDF_SD_NAMETOINDEX(SD_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_SETINFO - Sets information about SD dataset.
HDF_SD_SETINFO, SDS_ID, [ . FILL=value ]
[. FORMAT=string ]
[. LABEL=string ]
[. RANGE=[min, max] ]
[. UNIT=string ]
[. COORDSYS=string ]
[. CALCDATA=structure ]

HDF_SD_START - Opens or creates HDF file and initializes SD interface.
Result = HDF_SD_START(Filename [, .READ [, .RDWR [, .CREATE ] ] ] )

HDF_UNPACKDATA - Unpacks array of byte data into IDL variables.
HDF_UNPACKDATA, packeddata, data1 [, data2
[. data3 ]
[. data4 [, data5 [, data6 [, data7 [, data8]]]]]]]
[. HDF_ORDER=arrary ]
[. HDF_TYPE=arrary ]
[. NREC=records ]

HDF_VD_ATTACH - Accesses a VData with the given ID.
Result = HDF_VD_ATTACH( FileHandle, VDataId
[. /READ ] [. /WRITE ] )

HDF_VD_ATTRFIND - Returns an attribute's index number given the name of an attribute.
Result = HDF_VD_ATTRFIND(VData, FieldID, Name)

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_ATTRSET, VData, FieldID, Attr_Name, Values
[. Count=variable ]
[. /BYTE ]
[. /DFNT_CHAR8 ]
[. /DFNT_FLOAT32 ]
[. /DFNT_FLOAT64 ]
[. /DFNT_INT8 ]
[. /DFNT_INT16 ]
[. /DFNT_INT32 ]
[. /DFNT_UINT8 ]
[. /DFNT_UINT16 ]
[. /DFNT_UINT32 ]
[. /DOUBLE ]
[. /FLOAT ]
[. /INT ]
[. /LONG ]
[. /SHORT ]
[. /STRING ]
[. /UINT ]
[. /ULONG ]

HDF_VD_DETACH - Called when done accessing a VData.
HDF_VD_DETACH, VData

HDF_VD_FDEFINE - Adds new field specification for VData.

HDF_VD_FEXIST - Returns true if specified fields exist in HDF file.
Result = HDF_VD_FEXIST(VData, Fieldnames)

HDF_VD_FIND - Returns reference number of specified VData.
Result = 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.
Result = HDF_VD_GETID(FileHandle, VDataId)

HDF_VD_GETINFO - Returns information about each VData field.
HDF_VD_GETINFO, VData, Index [, .NAME=variable ]
[. ORDER=value ]
[. 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.
Result = HDF_VD_ISATTR(VData)

HDF_VD_ISVD - Returns True (1) if an object is a VData.
Result = HDF_VD_ISVD(VGroup, Id)

HDF_VD_ISVG - Returns True (1) if an object is a VGroup.
Result = HDF_VD_ISVG(VGroup, Id)

HDF_VD_LONE - Returns array containing all VDatas that are not contained in another VData.
Result = HDF_VD_LONE( FileHandle [, .MAXSIZE=value ] )

HDF_VD_NATTRS - Returns the number of attributes associated with the specified VData.
Result = HDF_VD_NATTRS( VData, FieldId )
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.
HDF_VG_ISVD(VGroup, Id)

HDF_VG_ISVG - Returns true if object is a VGroup.
HDF_VG_ISVG(VGroup, Id)

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=string ]

HDF5 Routines

H5_BROWSE - Presents a graphical user interface for viewing and reading HDF5 files.
Result = H5_BROWSE([Files] [, /DIALOG_READ] )

H5_CLOSE - Flushes all data to disk, closes file identifiers, and cleans up memory.
H5_CLOSE

H5_CREATE - Creates and closes a new HDF5 file.
H5_CREATE, Filename, Structure

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_CREATE - Creates a dataset as an attribute of another group or dataset.
Result = H5A_CREATE( Loc_id, Name, Datatype_id, Dataspace_id)

H5A_DELETE - Removes the attribute specified by its name from a dataset, group, or named datatype.
H5A_DELETE, Loc_id, Name

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_OPENIDX - Opens an existing attribute by the index of that attribute.
Result = H5A_OPENIDX(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 [, Datatype_id])

H5A_WRITE - Writes data to an attribute.
H5A_WRITE, Attribute_id, Data

H5D_CLOSE - Closes the specified dataset and releases its used resources.
H5D_CLOSE, Dataset_id

H5D_CREATE - Creates a dataset at the specified location.
Result = H5D_CREATE(Loc_id, Name, Datatype_id, Dataspace_id [, CHUNK_DIMENSIONS=vector [, GZIP=value [, /SHUFFLE]]])

H5D_EXTEND - Changes the current dimensions of the Dataset within the limits of the Dataspace.
H5D_EXTEND, Dataset_id, Size

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 [, Datatype_id] [, FILE_SPACE=setId] [, MEMORY_SPACE=setId])

H5D_WRITE - Writes data to a dataset.
H5D_WRITE, Dataset_id, Data
[ , MEMORY_SPACE_ID=value]
[ , FILE_SPACE_ID=value]

H5F_CLOSE - Closes the specified file and releases resources used by it.
H5F_CLOSE, File_id

H5F_CREATE - The primary function for creating HDF5 files.
Result = H5F_CREATE(Filename)

H5F_IS_HDF5 - Determines if a file is in the HDF5 format.
Result = H5F_IS_HDF5(Filename)

H5F_OPEN - Opens an existing HDF5 file.
Result = H5F_OPEN(Filename)([, /WRITE])

H5G_CLOSE - Closes the specified group and releases resources used by it.
H5G_CLOSE, Group_id

H5G_CREATE - Creates a new empty group and gives it a name.
Result = H5G_CREATE(Loc_id, Name)

H5G_GET_COMMENT - Retrieves a comment string from a specified object.
Result = H5G_GET_COMMENT(Loc_id, Name)

H5G_GET_LINKVAL - Returns the name of the object pointed to by a symbolic link.
Result = H5G_GET_LINKVAL(Loc_id, Name)

H5G_GET_MEMBER_NAME - Retrieves the name of an object within a group, by its zero-based index.
Result = H5G_GET_MEMBER_NAME(Loc_id, Name, Index)

H5G_GET_NMEMBERS - Returns the number of objects within a group.
Result = H5G_GET_NMEMBERS(Loc_id, Name)

H5G_GET_NUMOBJ - Returns number of objects in the group specified by its identifier.
Result = H5G_GET_NUMOBJ(Loc_id)

H5G_GET_OBJ_NAME_BYIDX - Returns a name of an object specified by an index.
Result = H5G_GET_OBJ_NAME_BYIDX(Loc_id, Index)

H5G_GET_OBJINFO - Retrieves information from a specified object.
Result = H5G_GET_OBJINFO(Loc_id, Name)
[, /FOLLOW_LINK]

H5G_LINK - Creates a link of the specified type. A link can only point to one of the three classes of named objects: group, dataset, and named datatype.
H5G_LINK, Loc_id, Current_Name, New_Name
[, SOFTLINK] [, NEW_LOC_ID=value]

H5G_MOVE - Renames/moves an object within an HDF5 group or file.
H5G_MOVE, Loc_id, Src_Name, Dst_Name
[, NEW_LOC_ID=value]

H5G_OPEN - Opens an existing group within an HDF5 file.
Result = H5G_OPEN(Loc_id, Name)
H5S_SET_COMMENT - Sets a comment for a specified object.
Result = H5S_SET_COMMENT(Dataspace_id, Comment)

H5G_UNLINK - Removes the link to an object from a group.
Result = H5G_UNLINK(Dataspace_id, Name)

H5S_GET_FILE_ID - Retrieves an identifier for the file containing the specified object.
Result = H5S_GET_FILE_ID(Dataspace_id)

H5S_GET_TYPE - Returns the object’s type.
Result = H5S_GET_TYPE(Dataspace_id)

H5S_CREATE_SCALAR - Creates a scalar dataspace.
Result = H5S_CREATE_SCALAR()

H5S_SET_EXTENT_NONE - Removes the extent of a dataspace.

H5R_CREATE - Creates a reference to either an object or a dataspace region of a dataset.
Result = H5R_CREATE(Dataspace_id, Name)

H5R_GET_REGION - Retrieves a dataspace associated with a region reference.
Result = H5R_GET_REGION(Dataspace_id, Reference)

H5S_CLOSE - Releases and terminates access to a dataspace.
H5S_CLOSE(Dataspace_id)

H5S_COPY - Copies an existing dataspace.
Result = H5S_COPY(Dataspace_id)

H5S_CREATE_SIMPLE - Creates a simple dataspace.
Result = H5S_CREATE_SIMPLE(Dimension, DATASPACE_ID=value)

H5R_DEREFERENCE - Opens a reference and returns the object identifier.
Result = H5R_DEREFERENCE(Dataspace_id, Reference)

H5R_GET_OBJECT_TYPE - Returns the type of object that an object reference points to.
Result = H5R_GET_OBJECT_TYPE(Dataspace_id, Reference)

H5R_GET_REGION - Returns the current class of a dataspace.
Result = H5R_GET_REGION(Dataspace_id)

H5S_SELECT_HYPER_NBLOCKS - Determines the number of hyperslab blocks in the current dataspace selection.
Result = H5S_SELECT_HYPER_NBLOCKS(Data)
H5T_COMMITTED - Sets or resets the committed state of a datatype.
H5T_COMMITTED, Datatype_id

H5T_ENUM_VALUES_TO_NAMES - Converts an enumeration datatype to corresponding names.
Result = H5T_ENUM_VALUES_TO_NAMES(Dataatype_id, Values)

H5T_EQUAL - Determines whether two type identifiers refer to the same datatype.
Result = H5T_EQUAL(Dataatype_id1, Dataatype_id2)

H5T_GET_ARRAY_DIMS - Returns the dimension sizes for an array datatype.
Result = H5T_GET_ARRAY_DIMS(Dataatype_id, [PERMUTATIONS=variable])

H5T_GET_ARRAY_NDIMS - Determines the number of dimensions (or rank) of an array datatype.
Result = H5T_GET_ARRAY_NDIMS(Dataatype_id)

H5T_GET_CLASS - Returns the datatype’s class.
Result = H5T_GET_CLASS(Dataatype_id)

H5T_GET_CSET - Returns the character set type of a string datatype.
Result = H5T_GET_CSET(Dataatype_id)

H5T_GET_EBIAS - Returns the exponent bias of a floating-point type.
Result = H5T_GET_EBIAS(Dataatype_id)

H5T_GET_FIELDS - Retrieves information about the positions and sizes of bit fields within a floating-point datatype.
Result = H5T_GET_FIELDS(Dataatype_id)

H5T_GET_INPAD - Returns the padding method for unused internal bits within a floating-point datatype.
Result = H5T_GET_INPAD(Dataatype_id)

H5T_GET_MEMBER_CLASS - Returns the datatype class of a compound datatype member.
Result = H5T_GET_MEMBER_CLASS(Dataatype_id, Member)

H5T_GET_MEMBER_INDEX - Retrieves the index of a specified member within a compound or enumeration datatype.
Result = H5T_GET_MEMBER_INDEX(Dataatype_id, Field_name)

H5T_GET_MEMBER_NAME - Returns the datatype name of a compound datatype member.
Result = H5T_GET_MEMBER_NAME(Dataatype_id, Member)

H5T_GET_MEMBER_OFFSET - Returns the byte offset of a field within a compound datatype.
Result = H5T_GET_MEMBER_OFFSET(Dataatype_id, Member)

H5T_GET_MEMBER_TYPE - Returns the datatype identifier for a specified member within a compound datatype.
Result = H5T_GET_MEMBER_TYPE(Dataatype_id, Member)

H5T_GET_MEMBER_VALUE - Returns the value of an enumeration datatype member.
Result = H5T_GET_MEMBER_VALUE(Dataatype_id, Member)

H5T_GET_NMEMBERS - Returns the number of fields in a compound datatype.
Result = H5T_GET_NMEMBERS(Dataatype_id)

H5T_GET_NORM - Returns the mantissa normalization of a floating-point datatype.
Result = H5T_GET_NORM(Dataatype_id)
IDL Quick Reference

Scientific Data Formats

**H5T_INSERT** - Adds a new member to the end of a compound datatype.

Result = H5T_INSERT(Datatype_id, Name, Field_id)

**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)

**H5T_REFERENCE_CREATE** - Creates a reference datatype object.

Result = H5T_REFERENCE_CREATE([REGION])

**H5T_SET_TAG** - Sets a tag string on an opaque data type.

Result = H5T_SET_TAG(Datatype_id, Tag)

**H5T_STR_TO_VLEN** - Converts an IDL H5_VLEN array of strings to an IDL string array.

Result = H5T_STR_TO_VLEN(Array [, /PTR_FREE])

**H5T_VLEN_CREATE** - Creates a variable length array datatype object.

Result = H5T_VLEN_CREATE(Datatype_id)

**H5T_VLEN_TO_STR** - Converts an IDL H5_VLEN array of strings to a string array.

Result = H5T_VLEN_TO_STR(Array [, /PTR_FREE])

NetCDF Routines

**NCDF_ATTNAME** - Returns the name of an attribute given its ID.

Result = NCDF_ATTNAME(Cdfid [, Variid] [, /GLOBAL], Outname)

**NCDF_ATTINQ** - Returns information about a netCDF attribute.

Result = NCDF_ATTINQ(Cdfid [, Variid] [, /GLOBAL], Name, Value)

**NCDF_ATTDEL** - Deletes an attribute from a netCDF file.

NCDF_ATTDEL[Cdfid [, Variid] [, /GLOBAL], Name]

**NCDF_ATTACH** - Creates an attribute in a netCDF file.

NCDF_ATTACH(Cdfid [, Variid] [, /GLOBAL], Name, Value)

**NCDF_ATTGET** - Retrieves value of an attribute from a netCDF file.

NCDF_ATTGET(Cdfid [, Variid] [, /GLOBAL], Name, Outname)

**NCDF_ATTNAME** - Renames an attribute in a netCDF file.

NCDF_ATTNAME(Cdfid [, Variid] [, /GLOBAL], Oldname, Newname)

**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_AttCOPY** - Copies an attribute from one netCDF file to another.

Result = NCDF_AttCOPY(Incdf [, Invarid] [, /GLOBAL], Outcdf [, Outvarid] [, /OUTGLOBAL])

**NCDF_DIMID** - Returns the ID of a netCDF dimension, given the name of the dimension.

Result = NCDF_DIMID(Cdfid, DimName)

**NCDF_GTORDER** - Returns the byte order of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the bit offset of the first significant bit in an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the precision in bits of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the padding method of the least significant bit (lsb) of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the byte order of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the sign type for an integer datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the base datatype from which a datatype is derived.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the bit offset of the first significant bit in an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the precision in bits of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the padding method of the least significant bit (lsb) of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the byte order of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the sign type for an integer datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the base datatype from which a datatype is derived.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the bit offset of the first significant bit in an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the precision in bits of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the padding method of the least significant bit (lsb) of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the byte order of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the sign type for an integer datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the base datatype from which a datatype is derived.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the bit offset of the first significant bit in an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the precision in bits of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the padding method of the least significant bit (lsb) of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the byte order of an atomic datatype.

Result = NCDF_GTORDER(Datatype_id)

**NCDF_GTORDER** - Returns the sign type for an integer datatype.
**NCDF_DIMINQ** - Retrieves the name and size of a dimension in a netCDF file, given its ID.

\[
\text{Result} = \text{NCDF\_DIMINQ}(\text{Cdfid}, \text{Dimid}, \text{Name}, \text{Size})
\]

**NCDF_DIMRENAME** - Renames an existing dimension in a netCDF file that has been opened for writing.

\[
\text{Result} = \text{NCDF\_DIMRENAME}(\text{Cdfid}, \text{Dimid}, \text{Name})
\]

**NCDF_EXISTS** - Returns True if the netCDF format library is supported on the current IDL platform.

\[
\text{Result} = \text{NCDF\_EXISTS}()
\]

**NCDF_INQUIRE** - Returns information about an open netCDF file.

\[
\text{Result} = \text{NCDF\_INQUIRE}(\text{Cdfid})
\]

**NCDF_OPEN** - Opens an existing netCDF file.

\[
\text{Result} = \text{NCDF\_OPEN}(\text{Filename}[, \text{NOWRITE}[, \text{WRITE}])}
\]

**NCDF_VARDEF** - Adds a new variable to an open netCDF file in define mode.

\[
\text{Result} = \text{NCDF\_VARDEF}(\text{Cdfid}, \text{Name}, \text{Dim}[, \text{BYTE}[, \text{CHAR}[, \text{DOUBLE}[, \text{FLOAT}[, \text{LONG}[, \text{SHORT}]])}
\]

**NCDF_VARGET** - Retrieves a hyperslab of values from a netCDF variable.

\[
\text{Result} = \text{NCDF\_VARGET}(\text{Cdfid}, \text{Varid}, \text{Value}[, \text{COUNT}]=\text{vector}[, \text{OFFSET}]=\text{vector}[, \text{STRIDE}]=\text{vector})
\]

**NCDF_VARGET1** - Retrieves one element from a netCDF variable.

\[
\text{Result} = \text{NCDF\_VARGET1}(\text{Cdfid}, \text{Varid}, \text{Value}[, \text{OFFSET}]=\text{vector})
\]

**NCDF_VARID** - Returns the ID of a netCDF variable.

\[
\text{Result} = \text{NCDF\_VARID}(\text{Cdfid})
\]

**NCDF_VARINQ** - Returns information about a netCDF variable, given its ID.

\[
\text{Result} = \text{NCDF\_VARINQ}(\text{Cdfid}, \text{Varid})
\]

**NCDF_VARPUT** - Writes a hyperslab of values to a netCDF variable.

\[
\text{Result} = \text{NCDF\_VARPUT}(\text{Cdfid}, \text{Varid}, \text{Value}[, \text{COUNT}]=\text{vector}[, \text{OFFSET}]=\text{vector}[, \text{STRIDE}]=\text{vector})
\]

**NCDF_VARRENAME** - Renames a netCDF variable.

\[
\text{Result} = \text{NCDF\_VARRENAME}(\text{Cdfid}, \text{Varid}, \text{Name})
\]