

Compaq Array Visualizer

Date: August, 2001

**Software
Version:** Array Visualizer Version 1.6

**Operating
Systems:** Microsoft Windows 98, Windows Me, Windows 95,
Windows 2000, or Windows NT Version 4

**Compaq Computer Corporation
Houston, Texas**

Copyright Information

Confidential computer software. Valid license from Compaq required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Compaq shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is provided "as is" without warranty of any kind and is subject to change without notice. The warranties for Compaq products are set forth in the express limited warranty statements accompanying such products. Nothing herein should be construed as constituting an additional warranty.

© 2001 Compaq Computer Corporation

Visual Fortran Home Page, Photographic images: Copyright © 1997 PhotoDisc, Inc.

Visual Fortran Home Page, Image: CERN, European Laboratory for Particle Physics: ALICE detector on CERN's future accelerator, the LHC, Large Hadron Collider.

Compaq, the COMPAQ logo, DEC, DEC Fortran, DIGITAL, OpenVMS, Tru64, VAX, VAX FORTRAN, and VMS are trademarks of Compaq Information Technologies Group, L.P. in the United States and other countries.

ActiveX, Microsoft, Microsoft Press, MS-DOS, PowerPoint, Visual Basic, Visual C++, Visual J++, Visual Studio, Win32, Win32s, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States and other countries.

Intel and Pentium are trademarks of Intel Corporation in the United States and other countries.

AMD and Athlon are trademarks of Advanced Micro Devices, Inc.

Borland and Delphi are trademarks of Borland Software Corporation.

CRAY is a registered trademark of Cray Research, Inc.

IBM is a registered trademark of International Business Machines, Inc.

IEEE is a registered trademark of the Institute of Electrical and Electronics Engineers,

Inc.

IMSL and Visual Numerics are registered trademarks of Visual Numerics, Inc.

Linux is a registered trademark of Linus Torvalds.

OpenGL is a registered trademark of Silicon Graphics, Inc.

OpenMP and the OpenMP logo are trademarks of OpenMP Architecture Review Board.

Sun Microsystems is a registered trademark and Java is a trademark of Sun Microsystems, Inc.

UNIX is a trademark of the Open Group in the United States and other countries.

All other product names mentioned herein may be trademarks of their respective companies.

New Features for Compaq Array Visualizer Version 1.6

New features added to Compaq Array Visualizer (Array Visualizer) Version 1.6 (since [Array Visualizer Version 1.5](#)) and its free viewer Array Viewer include the following:

- Three new fav routines for Fortran programmers: [favGetModifiedFlag](#), [favIsVisible](#), and [favSetModifiedFlag](#).
- Three new CAVIEWER class routines for C++ programmers: [GetModifiedFlag](#), [IsVisible](#), and [SetModifiedFlag](#).
- Two new data type codes for the following agl routines for C programmers: [aglAlloc](#) and [aglReShape](#).

The Array Viewer Help file now has a more detailed description of text file format, including descriptions of header lines that can be added to your text files.

New Features for Compaq Array Visualizer Version 1.5 and 1.5A

New features added to Compaq Array Visualizer (Array Visualizer) Version 1.5 and 1.5A (since [Array Visualizer Version 1.1](#)) and its free viewer Array Viewer include the following:

- The Array Viewer now has a File menu option to load a data file from an Internet URL.
- Changes have been made to the Avis2D and AvisGrid [ActiveX controls](#) to enable them to be used inside HTML pages.
- A resource leak in Array Viewer that occurred on Windows 95 and 98 systems has been fixed.
- The Array Viewer (and [Avis2D ActiveX control](#)) now use software based OpenGL rendering as the default. Compaq has found that many graphics adapters do not support OpenGL correctly, resulting in rendering artifacts or crashes. If your adapter does support OpenGL acceleration and is reliable, you can enable it by checking the Checkbox in the Array Viewer Options dialog and restarting Array Viewer.
- Printing from Array Viewer has been improved.
- Axis tickmark labels now always have fixed increments.
- HDF release 4.1r3 is now used by Array Viewer to display HDF files.
- The graph view in Array Viewer (and the Avis2D control) now changes the cursor to indicate when the view is being changed.
- The Array Viewer Options dialog, now has a checkbox that can be used to turn off OpenGL hardware acceleration and use the generic GDI OpenGL drivers instead.
- COMINITIALIZE is no longer required for QuickWin programs that use the AView library.
- The Palette Editor now allows spline curves to be used when editing palette colors.
- Height Plot graphs now use 1D textures by default.
- Barchart style for Height Plot graphs has been enhanced and a problem

with printing has been fixed.

- Logical and Complex arrays can now be viewed with Array Viewer from the Visual Fortran debugger and can also be used as arguments to `fagIStartWatch`, `fagIShow`, etc.
- The HDF libraries are no longer provided on the Array Visualizer CD. The latest version of the HDF libraries can be downloaded from:
<http://hdf.ncsa.uiuc.edu/>.
- A problem in the 1.1 release prevented Vector Graphs from being displayed as a series of connected line segments. This was fixed in 1.1A. Use the Vector Graph Settings dialog in Array Viewer to switch between point style and line style.
- In the Array Viewer Settings menu, the ROI Settings dialog now allows values greater than 64K.
- There are new Array Visualizer sample programs (see the `Samples.htm` file in `Program Files\ArrayVisualizer\Samples`) as well as improvements to the Array Viewer online help (Array Viewer Help menu) and online Array Visualizer HTML Help documentation.
- For additional details, see the online release notes (`relnotes.txt`), installed in `Program Files\ArrayVisualizer`.

New Features for Compaq Array Visualizer Version 1.1



New features added to Compaq New Features for Compaq Array Visualizer Version 1.1 (since Version 1.0.A) include the following:

- The AvisGrid [ActiveX control](#) was added.
- The ArrayViewer Data View window is no longer limited to 500x500 cells, and now supports editing cell values.
- Array Viewer now allows the axis legends to be customized.
- The Array Viewer has a Palette Editor that you can use to create custom palettes.
- The Array Viewer has two new toolbars that you can use to change the Region Of Interest (ROI).
- The Array Viewer has improved support for displaying small (10⁻¹⁰ or less) floating-point values.
- The Aview library routines [favUpdate](#) and [CAViewer::Update](#) now have an option that forces the ArrayViewer to update before returning.
- The Aview library has two new subroutines that can be used in conjunction with the AvisGrid and Avis2D controls to display array data: [faglGetShareName](#) (Fortran language) and [aglGetShareName](#) (C language).
- The Aview library supports new fav and CAViewer routines to allow programmatic access to new Array Viewer features (such as Axis labeling).
- The Avis2D control has two new methods designed for Visual Basic users: [CopyAxisScaleData](#) and [CopyPaletteData](#). These methods can be used to pass a Visual Basic array to the control as axis scale data and custom palette data respectively.
- The Avis2D control event, [RndrPass](#), has a parameter to inform the container of whether this is the final repaint for the last Update call.
- The Avis2D control transparently supports printing.
- There are many new Array Visualizer sample programs (see the `Samples.htm` file in `Program Files\ArrayVisualizer\Samples`) as well as improvements to the Array Viewer online help (Array Viewer Help menu) and online Array

Visualizer HTML Help documentation.

- o For additional details, see the online release notes (`relnotes.txt`), installed in `Program Files\ArrayVisualizer`.

Contents

In this document, links are denoted by a  or  when you pass your pointer over a blue-colored term. In either case, click on the link to see further information.

Compaq Array Visualizer includes the following online documentation:

- [Getting Started](#)
- [API Calls for Fortran Programmers](#)
- [API Calls for C Programmers](#)
- [API Calls for C++ Programmers](#)
- [Array Visualizer Controls](#)
- [Error Messages](#)

Online help for the user interface to the Array Viewer is available at:

`...\Array Visualizer\bin\aviewer.hlp`

Getting Started

Welcome to the Compaq Array Visualizer.

The Compaq® Array Visualizer is a software tool that lets you view and analyze array data graphically. The Array Visualizer's advanced data visualization techniques let you discover hidden patterns in large multidimensional arrays. An extensive number of customization options lets you bring out the important features of your data. The Array Visualizer uses OpenGL® (a high-speed 3D rendering library) to let you interactively move, rotate, and zoom data graphs.

The Array Visualizer includes several different software components:

- The Array Viewer application
- The Aview routines library
- The Avis2D and AvisGrid ActiveX® controls
- Enhancements to the Microsoft® Visual C++® development environment (visual development environment).

The Compaq Array Viewer is a Windows application that displays array data in two adjustable panes:

- The top pane shows the array's numeric values in a scrollable spreadsheet.
- The bottom pane displays a graphical view of the array data as a three-dimensional surface. Other viewing modes include graphical representation of the array data as plane view image map, vector graph, or two-dimensional chart.

The Aview routines library contains a set of subroutines that allow Visual Fortran or Visual C++ applications to display array data using the Array Viewer (via OLE Automation). Using this library, you can create data visualization applications with just a few lines of code! The library routines can also save array data as a file for later viewing with the Array Viewer.

The Avis2D and AvisGrid ActiveX (OCX) controls can be used by any development environment that supports ActiveX controls (Visual C++, Visual Basic®, Visual Fortran dialogs) to display array data in a variety of graphing modes. The Avis2D control provides more than 100 properties, methods, and events that the developer can use to customize its behavior. The AvisGrid control can be used to create tables of array data using about 30 properties, methods, and events.

The visual development environment provided with the Visual Fortran Professional or Enterprise Edition is enhanced to let you select an array in the debugger watch window and use the Array Viewer to inspect it (see the Debugger chapter in the *Visual Fortran Programmers Guide*). You can update

the view to observe changes in the array data as the program executes.

This section discusses the following topics:

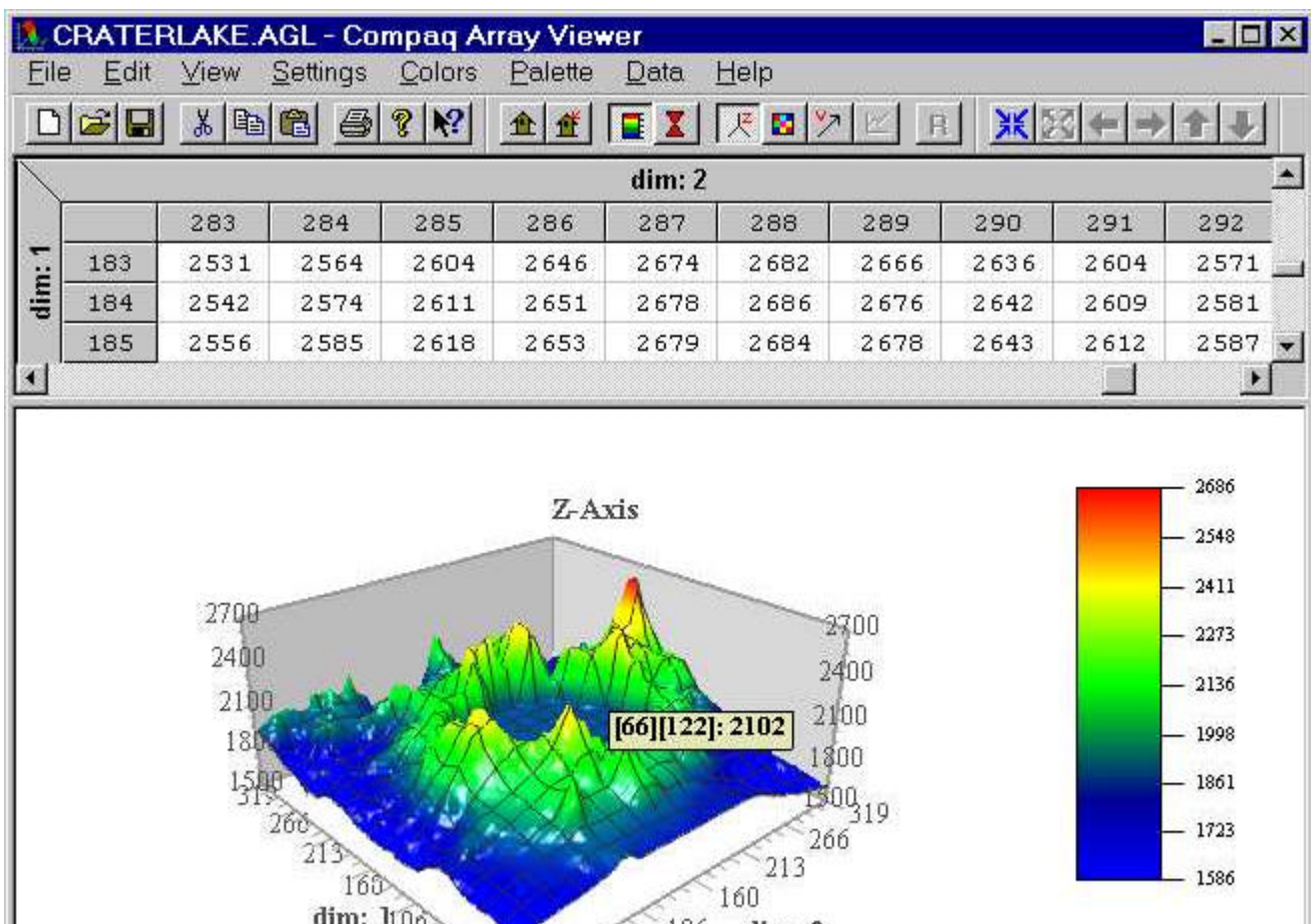
- o [Using the Array Viewer](#)
- o [Overview](#)
- o [Development Scenarios](#)
- o [Using the Fortran and C APIs](#)
- o [Using the Array Visualizer Controls](#)
- o [Redistribution of the Array Visualizer Controls and Array Viewer](#)
- o [Glossary](#)

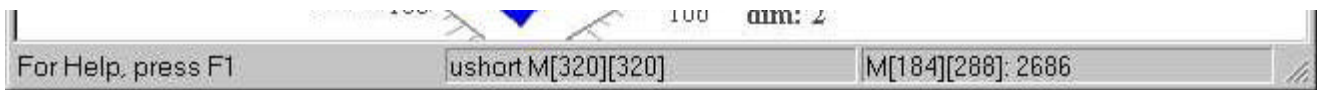
Using the Array Viewer

The Array Viewer displays array data in two adjustable panes:

- o The top pane shows the array's numeric values in a scrollable spreadsheet.
- o The bottom pane displays a graphical view of the array data as a three-dimensional surface. Other viewing modes include graphical representation of the array data as an image map, a vector graph, or a plane graph.

The following figure shows the Array Viewer, with a height plot view:





When you initially start Array Viewer from the Compaq Array Viewer program group (or by double-clicking the file `Avview.exe`), some typical actions include:

- Open a file

To open a file, in the File menu, click Open. Select the folder and the file you want to open. You can open AGL files (as shown), HDF files, and other file types. The Array Visualizer Sample graphics files are installed during a Complete installation.

[Show me how to Open a File](#) (click the Back button when done)

- Rotate the current view

You can drag the pointer to adjust the angle of view.

[Show me how to adjust the angle of the view](#) (click the Back button when done)

- Select a different type of view

From the View menu, select the type of view, such as Height plot, Image map, Vector graph, or (if applicable) Plane view or use the toolbar.

[Show me how to select a new type of view](#) (click the Back button when done)

- Display and move the marker

To display the red marker, click the appropriate button on the toolbar.

To move the marker, do one of the following:

- Double-click the appropriate area on the graph.
- Use the arrow (cursor) keys. Pressing an arrow key moves the marker one cell in the direction indicated by the key. Pressing Shift and an arrow key moves the marker by 10% of the array width or height.
- Double-click the appropriate array element cell in the top pane.

[Show me how to display and change the position of the marker](#)

- Display toolbars and move toolbars

Array Viewer provides four toolbars you can select in the View menu,

Toolbars item. You can move a toolbar by dragging its outline.

[Show me how to display and move toolbars](#)

- Adjust the size of the image in the bottom pane (graph window)

While holding down the `shift` key, move the pointer up to reduce the size of the image or move the pointer down to enlarge the image.

You can also use the Zoom In and Zoom Out buttons on the ROI2 toolbar.

[Show me how to enlarge or reduce the size of the image](#)

- Adjust the sizes of the top pane (array values) and bottom pane (graph view)

Move the pointer to the boundary between the two panes and drag the boundary to the desired size. Alternatively, in the View menu, click Split. Adjust the top and bottom pane sizes as needed by dragging the pointer.

For more information about using the Array Viewer, in the Array Viewer Help menu, click Help Topics.

Overview

Developing interactive data analysis applications is a difficult task. In addition to the complexities inherent in writing code to collect and manipulate data, developing an intuitive user interface for an interactive 3D graphing tool is far from being an easy task. Furthermore, using API functions for 3D graphics, such as OpenGL, is quite complex and requires a large amount of time-consuming programming.

Array Visualizer (AVIS) makes the job of writing a high quality array viewer for your customers easier by providing API functions in the Aview library, as well as two ActiveX controls. Using Array Visualizer lets you focus on the simulation or modeling aspects of your application without spending excessive time writing low-level graphics code.

There are several different approaches to using the Array Visualizer. The best approach for you will depend on the particular problem you are trying to solve and the amount of time you have to spend to add visualization features to your program.

Here are the four basic options to using the Array Visualizer:

Option	Advantages:	Disadvantages:
<p>Use the Visual Fortran debugger to view array data with the Array Viewer program (see the Visual Fortran Programmer's Guide, Debugger chapter).</p>	<ul style="list-style-type: none"> • It does not require any code modifications • It works with any project type 	<ul style="list-style-type: none"> • Requires manual invocation and customization of array properties • Cannot be used from Visual C++ or Visual Basic • Cannot be used in Release mode
<p>Use the fagl subroutines (or agl functions for C programmers) to view array data with the Array Viewer program</p>	<ul style="list-style-type: none"> • Small number of routines (about 10) are easy to learn and use • Minimal changes are required to existing code • Works with all project types and with both Debug and Release builds 	<ul style="list-style-type: none"> • Requires manual customization of Array Viewer properties (if desired)
<p>Use the fagl (agl) subroutines in conjunction with the fav routines (or CAViewer class routines for C++ programmers).</p>	<ul style="list-style-type: none"> • Allows the Array Viewer settings to be customized to view your array data in the most appropriate fashion. • Works with any project type • Multiple arrays can be shown sequentially in one Array Viewer instance 	<ul style="list-style-type: none"> • More routines to learn! For example, there are about 100 fav routines (though typically you'll only need to use a small subset of these) • Can't use CAViewer class from C (requires C++).

<p>Use the Avis2D and/or AvisGrid ActiveX controls</p>	<ul style="list-style-type: none"> • Allows you to incorporate Graph and Data views directly into your program • Startup time will be faster, because there is no external process to initiate • Allows more possibilities for customization 	<ul style="list-style-type: none"> • Only works with Fortran Windows project type (for Visual Fortran) or MFC (for Visual C++). Note: In Visual Basic, most EXE project types can use the Avis2D and AvisGrid controls. • There are lot of Avis2D/AvisGrid properties, methods, and events to become familiar with! • Avis2D and AvisGrid can't be used to display HDF or text data files
--	---	--

Development Scenarios

Using the visual development environment interface to launch Array Viewer requires no modifications of a source program. It lets you inspect the array data and optionally save the entire array to an .AGL file for later viewing.

Array Viewer includes many user interface controls that can adjust the way in which data is presented. For example, you can select a particular sub-region of the array to be displayed. See the Array Viewer help (Aviewer.hlp) for more information on how to use Array Viewer.

Using the Fortran or C routines (Aview library) lets the application launch Array Viewer independently of the visual development environment to display array data. The number of function calls needed is quite small and is available to any type of project type (Fortran Console, Fortran QuickWin, Fortran Windows, and so on). The Aview library provides a way to save array data directly to an .AGL file for later viewing with Array Viewer.

Using the Avis2D or AvisGrid controls is slightly more complicated than using the API, and they require a development environment that supports ActiveX controls. You can use these controls to do the following:

- Fine-tune the way in which the data will be displayed
- Include the graph or grid view within the top-level window of the application

Using the Fortran and C APIs

The primary purpose of the Array Visualizer routines is to enable viewing array data using the Array Viewer. Another set of routines allow the program to control how the array data is viewed, display a different array, and other functions. The routines can also save array data to an .AGL file.

Two sets of routines are provided for Fortran programmers:

- [APIs for Fortran Programmers: fagl Routines](#)
This small set of routines start the Array Viewer, display an array, and perform other major functions. The user of the Array Viewer interacts with Array Viewer to perform certain tasks. The routine names have a prefix of **fagl**, such as faglStartWatch.
- [APIs for Fortran Programmers: fav Routines](#)
This comprehensive set of routines allows a program to control the appearance of the Array Viewer window and perform actions that would otherwise require user interaction. The routine names have a prefix of **fav**, such as favStartViewer.

If you start the Array Viewer with favStartViewer, use of certain fagl routines like faglHide, faglName, faglShow, and faglUpdate are ignored. Instead, use the corresponding fav routines. For example, instead of using faglShow, use favSetArray.

Two sets of routines are provided for C/C++ programmers:

- [APIs for C Programmers: agl Routines](#)
This small set of routines start the Array Viewer, display an array, and perform other major functions. The user of the Array Viewer interacts with Array Viewer to perform certain tasks. The routine names have a prefix of **agl**, such as aglStartWatch, and can be called from C or C++ programs.
- [APIs for C++ Programmers: CAVIEWER Routines](#)
This comprehensive set of routines allows a program to control the appearance of the Array Viewer window and perform actions that would otherwise require user interaction. The routine names use the **CAVIEWER** class and are usually called from C++ programs.

To find out more about the routines and how to use them, see:

- [APIs for Fortran Programmers: fagl Routines](#)
- [APIs for Fortran Programmers: fav Routines](#)
- [APIs for C Programmers: agl Routines](#)
- [APIs for C++ Programmers: CAVIEWER Class Routines](#)

Using the Array Visualizer Controls

For developers, there are several advantages in using the AVis2D and Grid controls:

- You can choose which features to expose to users.
- By using other controls to give your users influence over property values, you can build into your application a set of graphing facilities that perfectly suits your application.

To find out more about the Avis2D and AvisGrid controls, see [Array Visualizer Controls](#).

Redistribution of Array Visualizer Components

The Aviewxxx.DLL (Aview110.DLL for Array Visualizer Version 1.1) should be redistributed with the application if the application uses the Fortran or C library routines. Copy the DLL to the system directory, the same directory as the application, or a directory in the PATH of the user.

For applications that use the Avis2D ActiveX control, copy the Avis2D.ocx file to the target system and register the control with the following command:

```
regsvr32 Avis2D.ocx
```

For applications that use the AvisGrid ActiveX control, copy the AvisGrid.ocx file to the target system and register the control with the following command:

```
regsvr32 AvisGrid.ocx
```

If the target system has not installed Array Viewer or Array Visualizer from a Visual Fortran kit, download the file containing the Array Viewer executable (Aviewer.exe) from the Visual Fortran Web site (<http://www.compaq.com/fortran/>) and install it on the target system.

Glossary

axis scales

Data that allows custom numeric values to be displayed along the dimension axis, as opposed to array index values.

camera

The virtual viewpoint. Can be set in Cartesian coordinates with properties

[CamXPos](#), [CamYPos](#), and [CamZPos](#), or in spherical coordinates with [CamAzimuth](#), [CamElevation](#), and [CamDistance](#).

COI (center of interest)

The point at which the camera is “focused” in the view volume. Properties [CamXCoi](#), [CamYCoi](#), and [CamZCoi](#) can be used to set the COI.

data tip

A rectangular window that displays information about a data point directly under the cursor. For example, row and column indexes, and value.

face

Any of the six sides of the [View Volume](#).

grid lines

Lines that run along the surface of the graph in the X and Y directions. Use of grid lines provides enhanced perspective for the graph. You can enable or disable grid lines using the [ShowGrid](#) property.

palette

The colors available for graphing. More specifically, a sequence of colors that map data values to colors in the graph, so that each value is associated with a color in the palette.

rank

The number of dimensions in the array.

ROI (region of interest)

A rectangular subset of array data to which the graph is limited. The [SetRoiLB](#) and [SetRoiUB](#) methods, and the [ColumnStart](#), [RowStart](#), [NumColumns](#), and [NumRows](#) properties, are all available for defining an ROI.

shape

The rank, dimensions, and type of the array.

view volume

The box-shaped 3D region where the graph is displayed. Opposite corners (0, 0, 0) and ([XScale](#), [YScale](#), [ZScale](#)) define the view volume.

API Calls for Fortran Programmers: fagl Routines

Fortran programmers can use the following API calls (faglxxx routines) to incorporate the Array Viewer in their applications:

API Calls for Fortran Programmers

[faglClose](#)

[faglEndWatch](#)

[faglGetShareName](#)

[faglHide](#)

[faglLBound](#)

[faglName](#)

[faglSaveAsFile](#)

[faglShow](#)

[faglStartWatch](#)

[faglUpdate](#)

These routines are typically used in the following sequence:

- Before an array can be displayed or saved to a file, call **faglStartWatch** with the array as an argument.
- If you want the array to be displayed in Array Viewer with a starting array index other than 1, call **faglLBound** with the array as an argument and another array specifying the lower bound values.
- If you want the Array Viewer to display a descriptive name on its title bar, call **faglName** with the array and descriptive name as arguments.
- To start the Array Viewer, call **faglShow** with the array as an argument. The Viewer is displayed until the user closes the application, or the **faglClose** call is invoked.
- If later in your program the array data has changed, and you want the viewer to display the modified data, call **faglUpdate** with the array as an argument.
- If you want to save the array as an Array Graphing Language (.AGL) file, call **faglSaveAsFile** with the array as an argument. (You do not need to have the Viewer active to use this call.)

- When you are finished viewing the array, call **faglEndWatch** with the array as an argument.

You can use **faglGetShareName** to get a string identifier that identifies the shared memory region associated with an array.

You can also use the [fav routines](#) to perform tasks usually done interactively by the user.

The USE AVDEF statement includes the AVDEF module file that defines the routine interfaces (see the file `...ArrayVisualizer\INCLUDE\AVDEF.F90`).

Samples of these routines are provided online in folders in `...ArrayVisualizer\Samples\Fortran\`. For a description of the Array Visualizer Samples, view the file `...ArrayVisualizer\Samples\Samples.htm` in a Web browser. Most Samples include a project workspace file, which allows you to open and build the project in the visual development environment.

Samples are installed on the hard disk when a Complete installation is performed. You can copy Samples folders from the Array Visualizer CD-ROM to your hard disk (remove the read-only file property).

faglClose

AVDEF Subroutine: Closes an instance of the Array Visualizer.

Module: USE AVDEF

Syntax

faglClose (*array*, *status*)

array

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

faglClose closes the instance of the Array Visualizer associated with the specified array. If, later in the execution of the program, you want to display the array again, you need only call **faglShow**. Another call to **faglStartWatch** is

not necessary.

See Also

[faglEndWatch](#), [faglShow](#), [faglStartWatch](#)

faglEndWatch

AVDEF Subroutine: Removes the array from the list of viewable arrays.

Module: USE AVDEF

Syntax

faglEndWatch (*array*, *status*)

array

A name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Removes the specified array from the list of viewable arrays and frees any associated resources used by this library of subroutines. To display the array at a later time, your application *must* again call both **faglStartWatch** and **faglShow**.

Before calling **faglEndWatch**, you might want to save the array as a .AGL file using **faglSaveAsFile**.

See Also

[faglSaveAsFile](#), [faglStartWatch](#)

faglGetShareName

AVDEF Subroutine: Gets a string identifier that identifies the shared memory region associated with an array.

Module: USE AVDEF

Syntax

faglGetShareName (*array, filename, status*)*array*

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

filename

An output argument of type CHARACTER string. The length of the string should be AV_SHARENAME_LEN.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The string returned by **faglGetShareName** can be passed to the Avis2D or AvisGrid controls as the **FileName** property. This allows array data to be displayed by applications that host these controls without saving the array to a disk file.

See Also

[faglStartWatch](#), Avis2D [FileName property](#), AvisGrid [FileName property](#)

faglHide

AVDEF Subroutine: Causes an instance of Array Visualizer to become invisible.

Module: USE AVDEF

Syntax**faglHide** (*array, status*)*array*

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

faglHide causes the instance of Array Visualizer associated with *array* to become invisible but does not close that instance. To close the instance but retain its associated resources, call **faglClose**. To close the instance and release its associated resources, call **faglEndWatch**. To make the Array Visualizer instance visible again, call **faglShow**.

If the Array Viewer instance associated with *array* was created by using [favStartViewer](#), rather than **faglShow**, this routine will have no effect. Use the [favShowWindow](#) routine instead.

See Also

[faglClose](#), [faglEndWatch](#), [faglShow](#), [faglStartWatch](#)

faglBound

AVDEF Subroutine: Sets the lower bound indices for the array that will be displayed in the Data and Graph Views of the Array Viewer.

Module: USE AVDEF

Syntax

faglBound (*array*, *lbnd*, *status*)

array

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

lbnd

An input Integer array of rank 1; its extent should be equal to the rank of array.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If the **faglBound** routine is not used, the Array Viewer will display the array using a lower bound index of 1.

faglName

AVDEF Subroutine: Places a specified string onto the title bar of an Array

Viewer instance.

Module: USE AVDEF

Syntax

faglName (*array, title, status*)

array

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

title

The string to be displayed on the title bar of the specified instance of Array Viewer.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If the Array Viewer instance associated with *array* was created by using [favStartViewer](#), rather than **faglShow**, this routine will have no effect. Use the [favSetArrayName](#) routine instead.

See Also

[faglShow](#), [faglStartWatch](#)

faglSaveAsFile

AVDEF Subroutine: Saves the current array with a .AGL extension.

Module: USE AVDEF

Syntax

faglSaveAsFile (*array, filename, status*)

array

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

filename

Either just the name of the file to be saved (not including the

path) in the calling application's directory or a complete path/filename combination to save the file in any other directory.

status

An argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If your end users are likely to use the array again with Array Visualizer, use this subroutine to save it. The users can run Array Viewer as a stand-alone program to display the array.

faglShow

AVDEF Subroutine: Creates an instance of Array Visualizer displaying data for a specified array.

Module: USE AVDEF

Syntax

faglShow (*array*, *status*)

array

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

faglShow creates an instance of Array Visualizer and displays it. **faglShow** can also make an existing instance of Array Visualizer visible if that instance has been rendered invisible by a call to **faglHide**. Before showing the instance of Array Visualizer, you might want to call **faglName** to put a suitable heading on its title bar.

If the Array Viewer instance associated with *array* was created by using [favStartViewer](#), rather than **faglShow**, this routine will have no effect. Use the [favShowWindow](#) routine instead.

See Also

[faglHide](#), [faglName](#), [faglStartWatch](#)

faglStartWatch

AVDEF Subroutine: Adds the specified array to the list of viewable arrays and returns a handle to be used by other subroutines in this library.

Module: USE AVDEF

Syntax

faglStartWatch (*array*, *status*)

array

A name of an array.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

faglStartWatch uses system resources to prepare the specified array for a call to **faglShow**. To return these resources, call **faglEndWatch**.

See Also

[faglEndWatch](#), [faglShow](#)

faglUpdate

AVDEF Subroutine: Re-synchronizes Array Visualizer's display of the array data with the actual data values.

Module: USE AVDEF

Syntax

faglUpdate (*array*, *status*)

array

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If the application has modified the values in the array associated with *array* since the last **faglUpdate** or **faglShow** call, and you want Array Visualizer to reflect the new data values, you can do so by calling **faglUpdate**.

If the Array Viewer instance associated with *array* was created by using [favStartViewer](#), rather than **faglShow**, this routine will have no effect. Use the [favUpdate](#) routine instead.

See Also

[faglShow](#), [faglStartWatch](#)

Examples

The following Visual Fortran example program illustrates the use of the Fortran routine APIs (faglxxx routines). This is a Sample program that can be found in the folder ...\\ArrayVisualizer\\Samples\\Fortran\\Simple2\\:

```

program MAIN

    ! AVDef is the Array Visualizer module file
    use AVDef
    use DFLib

    IMPLICIT NONE
    ! Define a 2D array of reals
    integer, parameter :: lbc=1, ubc=40, lbr=1, ubr=50
    real, parameter :: pi=3.14159
    ! Using allocatable array M for array viewing
    ! The array_visualizer attribute will result in better performance
    ! when using the aview lib with allocatable arrays.
    real(4), allocatable :: M(:, :)
    !DEC$ATTRIBUTES array_visualizer :: M
    integer :: lbnd(2) = 0
    real x, y, z, rval
    integer status, i, j, arrayData
    character(1) :: key

    ! allocate memory for the array
    allocate(M(lbc:ubc, lbr:ubr))

    print *, "Initializing array data"
    do i=lbr,ubr
        x = i/(ubr-lbr+1.0)
        do j=lbc,ubc
            y = j/(ubc-lbc+1.0)
            z = sin(x*pi) + cos(y*pi);
            M(j, i) = z
        end do
    end do

```

```

end do

! Call StartWatch since we are interested in viewing M
call faglStartWatch(M, status)

! Set lbnd to the lower bound values of M
! (This is not really needed if we are using the default Fortran array
! indexing value of 1)
lbnd(1:size(shape(M))) = lbound(M)
call faglLBound(M, lbnd, status)

print *, "Starting Array Viewer"
call faglShow(M, status)

! Set the title bar on ArrayViewer
call faglName(M, "sin(x) + cos(y)", status)

print *, "press any key to continue"
key = GETCHARQQ()

print *, "Adding some random fluctuations to the array data"
do i=lbr,ubr
    do j=lbc,ubc
        call RANDOM(rval)
        rval = rval * 0.2 - 0.1
        M(j, i) = M(j, i) + rval
    end do
end do

print *, "Informing the viewer that the array data has been changed."
call faglUpdate(M, status)

! Change the title to reflect the changes in the data set
call faglName(M, "sin(x) + cos(y) + noise", status)

print *, "press any key to continue"
key = GETCHARQQ()

! Uncomment the following line to have the ArrayViewer
! closed automatically.
! call faglClose(M, status)

! Remove M from the watch list
call faglEndWatch(M, status)

! Free the memory allocation
deallocate(M)

print *, "Done!"

end program MAIN

```

This example illustrates most of the Fortran API calls. The program displays the initial values of allocatable array *M*, and then, after calling [faglUpdate](#), displays the new values.

This example also shows the use of the `cDEC$ ATTRIBUTES` option `ARRAY_VISUALIZER`:

```
real(4), allocatable :: MyArray(:, :)  
!DEC$ ATTRIBUTES array_visualizer :: MyArray
```

When this option is used, array memory can be shared between the Array Viewer and your application. Otherwise, the array data will need to be copied during each faglUpdate call.

Array Visualizer Sample programs are installed on your hard disk if you selected a Complete installation. You can also copy Samples folders from the Array Visualizer CD-ROM to your hard disk.

API Calls for Fortran Programmers: fav Routines

Fortran programmers can use the following API calls (favxxx routines) to control the appearance of the Array Viewer window and the user-interaction of the Array Viewer in their applications. These routines are typically used in conjunction with one or more of the [fagl routines](#).

The fav routines include the following groups:

- [Array Viewer Instance](#)
- [Data Access](#)
- [Region of Interest \(ROI\)](#)
- [Data Filter](#)
- [Palette](#)
- [Graph Axis](#)
- [Selection](#)
- [Graph View \(common to all graph types\)](#)
- [Height Plot](#)
- [Image Map](#)
- [Vector Graph](#)
- [Data View](#)
- [Camera](#)
- [Marker](#)
- [Miscellaneous](#)

The USE AVVIEWER statement includes the AVVIEWER module file that defines the routine interfaces and some parameter constants. For definitions of the parameter constants GraphType, GraphStyle, ImageOrientation, StandardPalettes, AV_TRUE, and AV_FALSE, view the file `...ArrayVisualizer\INCLUDE\AVVIEWER.F90`.

Samples of these routines are provided online in folders in `...ArrayVisualizer\Samples\Fortran\`, such as AXIScale2d, Mandel, and Palette2D. Most Samples include a project workspace file, which allows you to open and build the project in the visual development environment.

Samples are installed on the hard disk when a Complete installation of Array Visualizer is performed. You can copy Samples folders from the Array Visualizer CD-ROM to your hard disk.

Array Viewer Instance	
Create an Array Viewer instance.	favStartViewer
Terminate an Array Viewer instance.	favEndViewer

Data Access	
Loads and displays a specified file in Array Viewer.	favSetFileName
Displays a specified array in Array Viewer.	favSetArray

Region of Interest (ROI)	
Sets the position number of the first and last elements in the region of interest (ROI) of the specified array dimension.	favSetRoi
Gets the position number of the first element in the ROI of the specified dimension.	favGetRoiLb
Gets the position number of the last element in the ROI of the specified dimension.	favGetRoiUb
Sets the Row and Column dimensions.	favSetRowColDim
Gets the Row and Column dimensions.	favGetRowColDim
Sets the ROI within the current 2D array slice.	favSetRoi2D

Data Filter	
Determines whether the Graph View clamps data values that are outside the clamp range.	favSetDataClamp
Determines whether the Graph View clamps data values that are outside the clamp range.	favGetDataClamp
Sets the upper and lower bounds for X-Coordinate values in the Graph View.	favSetXClamp
Gets the upper and lower bounds for X-Coordinate values in the Graph View.	favGetXClamp
Sets the upper and lower bounds for Y-Coordinate values in the Graph View.	favSetYClamp
Gets the upper and lower bounds for Y-Coordinate values in the Graph View.	favGetYClamp
Sets the upper and lower bounds for Z-Coordinate values in the Graph View.	favSetZClamp

Gets the upper and lower bounds for Z-Coordinate values in the Graph View.	favGetZClamp
Enables or disables the Data Refresh menu item.	favSetDataRefreshEnable
Enables or disables the Data Refresh menu item.	favGetDataRefreshEnable
Forces Array Viewer to redraw its view to reflect any data changes since the last Update (or initial load).	favUpdate

Palette	
Creates a custom palette.	favSetCustomPalette
Sets the color palette identifier.	favSetPaletteId
Gets the color palette identifier.	favGetPaletteId
Sets the range of data values with which the color palette will be associated with.	favSetPaletteRange
Gets the range of data values with which the color palette will be associated with.	favGetPaletteRange
Enables/Disables automatic adjustment of the palette range to the data range.	favSetPaletteAutoAdjust
Enables/Disables use of color palette.	favSetUseColorPalette
Gets the current setting of use color palette.	favGetUseColorPalette
Enables or disables display of the color palette.	favSetShowPalette
Gets the current setting of show color palette.	favGetShowPalette

Graph Axis	
Enables or disables Axis Auto Scaling.	favSetAxisAutoScale
Gets the AxisAutoScale setting.	favGetAxisAutoScale
Associates an axis dimension scale with a dimension.	favSetDimScale
Displays or hides the graph axis.	favSetShowAxis
Gets the show axis setting.	favGetShowAxis
Sets the label to be displayed along the Graph View's axis.	favSetAxisLabel

Gets the axis label of the given axis.	favGetAxisLabel
Enables or disables display of Axis labels.	favSetUseAxisLabel
Gets the current use axis label state for the given axis.	favGetUseAxisLabel
Sets a value that determines whether the font size used in axis labels changes as the size of the graph view window changes, or is held fixed.	favSetFontAutoScale
Gets the current FontAutoScale state.	favGetFontAutoScale
Sets the current axis style.	favSetAxisStyle
Gets the current axis style.	favGetAxisStyle
Determines whether the number of both large and small tick marks on the axes is to be handled automatically by Array Visualizer or explicitly.	favSetAxisAutoDetail
Gets the current AxisAutoDetail state.	favGetAxisAutoDetail
Sets the number of large tickmarks displayed along the indicated axis.	favSetNumMajorTickmarks
Gets the number of large tickmarks for the desired axis.	favGetNumMajorTickmarks
Sets the number of small tickmarks for the desired axis.	favSetNumMinorTickmarks
Gets the number of small tickmarks for the desired axis.	favGetNumMinorTickmarks

Selection

Enables or disables data selection.	favSetDataSelectEnable
Gets the data selection enable setting.	favGetDataSelectEnable

Graph View

Sets the graph type (Height plot, Image map, Vector plot, or PlaneView).	favSetGraphType
Gets the current graph type.	favGetGraphType
Sets the graph style (mesh, surface, barchart, lines, or points).	favSetGraphStyle
Gets the current graph style.	favGetGraphStyle

Sets a value that determines how finely the grid is drawn on the graph surface.	favSetGridDensity
Gets the grid density value.	favGetGridDensity
Sets the depth cueing state.	favSetDepthcue
Gets the depth cueing setting.	favGetDepthcue
Sets a value that determines whether grid lines are displayed in the graph.	favSetShowGrid
Gets the ShowGrid setting.	favGetShowGrid
Enables or disables line smoothing.	favSetLineSmooth
Gets the line smooth setting.	favGetLineSmooth

Height Plot	
Sets the height of the Z-axis relative to the X and Y axis.	favSetZScale
Gets the ZScale value.	favGetZScale
Enables or disables one-dimensional (1D) textured color mapping.	favSetTextureMode
Gets the current texture mode state.	favGetTextureMode
Sets the current shading state.	favSetShading
Gets the current shading state.	favGetShading
Sets the HighLighting state.	favSetHighLight
Gets the current HighLighting state.	favGetHighLight
Sets the HiddenLine state.	favSetHiddenLine
Gets the current HiddenLine state.	favGetHiddenLine

Image Map	
Sets the Image Orientation setting (IDENTITY, XFLIP, YFLIP, XYFLIP).	favSetImageOrientation
Gets the current Image Orientation setting.	favGetImageOrientation
Sets the Fixed Aspect ration setting.	favSetFixedAspect

Gets the current Fixed Aspect Ratio setting.	favGetFixedAspect
Sets the Image Linear Filter setting.	favSetImageFilter
Gets the current Image Linear Filter settings.	favGetImageFilter

Vector Graph	
Sets the component index values.	favSetCompIndex
Gets the current component index values.	favGetCompIndex

Data View	
Sets the precision with which floating-point numbers should be displayed in the data view.	favSetPrecision
Gets the current data view precision value.	favGetPrecision
Enables or disables Cell Editing in the Data View window.	favSetCellEditEnabled
Gets the current Cell Edit state.	favGetCellEditEnabled
Gets the default field width and precision used for displaying data in the Data View window.	favGetDefaultFormat
Sets a value that determines whether to display data in the Data View window using the default format.	favSetUseDefaultFormat
Gets the current UseDefaultFormat state.	favGetUseDefaultFormat
Associates a name with the given dimension.	favSetDimName
Gets the name that's associated with the given dimension.	favGetDimName
Sets a value that determines whether to display the column and row dimension labels in the Data View window.	favSetShowDimLabels
Gets the current ShowDimLabels state.	favGetShowDimLabels
Sets a value that determines whether to display integer data in hexadecimal or decimal format in the Data View window.	favSetUseHex
Gets the current UseHex state.	favGetUseHex
Sets a value that determines whether to display floating-point data in scientific notation in the Data View window.	favSetUseExp
Gets the current UseExp state.	favGetUseExp

Sets a value that determines the width allocated to display each data value in the Data View window.	favSetFieldWidth
Gets the current field width in the Data View window.	favGetFieldWidth

Camera	
Sets the camera position.	favSetCameraPosition
Gets the current camera position.	favGetCameraPosition
Sets the camera center of interest (COI).	favSetCameraCoi
Gets the current camera center of interest.	favGetCameraCoi
Sets the current camera position as the home position.	favSetHomePosition
Sets the camera position to be the home position.	favToHomePosition

Marker	
Sets the row/column position.	favSetRowCol
Gets the current row/column position.	favGetRowCol
Shows or hides the marker.	favSetShowMarker
Gets the current show marker state.	favGetShowMarker

Miscellaneous	
Gets the error number.	favGetErrorNo
Gets the modified flag.	favGetModifiedFlag
Returns the Array Viewer visibility state.	favIsVisible
Sets the annotation string.	favSetAnnotation
Sets the string to be displayed on the Array Viewer title bar.	favSetArrayName
Sets the modified flag.	favSetModifiedFlag
Displays or hides the Array Viewer window.	favShowWindow

The following table alphabetically lists the fav routines:

favEndViewer	favGetShowDimLabels	favSetGridDensity
favGetAxisAutoDetail	favGetShowMarker	favSetHiddenLine
favGetAxisAutoScale	favGetShowGrid	favSetHighLight
favGetAxisLabel	favGetShowPalette	favSetHomePosition
favGetAxisStyle	favGetTextureMode	favSetImageFilter
favGetCameraCoi	favGetUseAxisLabel	favSetImageOrientation
favGetCameraPosition	favGetUseColorPalette	favSetLineSmooth
favGetCellEditEnabled	favGetUseDefaultFormat	favSetModifiedFlag
favGetCompIndex	favGetUseExp	favSetNumMajorTickmarks
favGetDataClamp	favGetUseHex	favSetNumMinorTickmarks
favGetDataRefreshEnable	favGetXClamp	favSetPaletteAutoAdjust
favGetDataSelectEnable	favGetYClamp	favSetPaletteId
favGetDefaultFormat	favGetZClamp	favSetPaletteRange
favGetDepthcue	favGetZScale	favSetPrecision
favGetDimName	favIsVisible	favSetRoi
favGetErrorNo	favSetAnnotation	favSetRoi2D
favGetFieldWidth	favSetArray	favSetRowCol
favGetFixedAspect	favSetArrayName	favSetRowColDim
favGetFontAutoScale	favSetAxisAutoDetail	favSetShading
favGetGraphStyle	favSetAxisAutoScale	favSetShowAxis
favGetGraphType	favSetAxisLabel	favSetShowDimLabels
favGetGridDensity	favSetAxisStyle	favSetShowGrid
favGetHiddenLine	favSetCameraCoi	favSetShowMarker
favGetHighLight	favSetCameraPosition	favSetShowPalette
favGetImageFilter	favSetCellEditEnabled	favSetTextureMode

favGetImageOrientation	favSetCompIndex	favSetUseAxisLabel
favGetLineSmooth	favSetCustomPalette	favSetUseColorPalette
favGetModifiedFlag	favSetDataClamp	favSetUseDefaultFormat
favGetNumMajorTickmarks	favSetDataRefreshEnable	favSetUseExp
favGetNumMinorTickmarks	favSetDataSelectEnable	favSetUseHex
favGetPaletteId	favSetDepthcue	favSetXClamp
favGetPaletteRange	favSetDimName	favSetYClamp
favGetPrecision	favSetDimScale	favSetZClamp
favGetRoiLb	favSetFieldWidth	favSetZScale
favGetRoiUb	favSetFileName	favShowWindow
favGetRowCol	favSetFixedAspect	favStartViewer
favGetRowColDim	favSetFontAutoScale	favToHomePosition
favGetShading	favSetGraphStyle	favUpdate
favGetShowAxis	favSetGraphType	

Samples are provided online in folders in `...\ArrayVisualizer\Samples\Fortran\`. For a description of the Array Visualizer Samples, view the file `...\ArrayVisualizer\Samples\Samples.htm` in a Web browser. Most Samples include a project workspace file, which allows you to open and build the project in the visual development environment.

Samples are installed on the hard disk when a Complete installation is performed. You can copy Samples folders from the Array Visualizer CD-ROM to your hard disk (remove the read-only file property).

favEndViewer

AVVIEWER Subroutine: Destroys an instance of the Array Viewer.

Module: USE AVVIEWER

Syntax

favEndViewer (*hv*, *status*)

hv

A handle that references an instance of Array Viewer created by **favStartViewer**.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favStartViewer](#)

favGetAxisAutoDetail

AVVIEWER Subroutine: Gets the current AxisAutoDetail state.

Module: USE AVVIEWER

Syntax

favGetAxisAutoDetail (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). It returns 1 if AxisAutoDetail is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetAxisAutoDetail](#)

favGetAxisAutoScale

AVVIEWER Subroutine: Gets the axis auto scale setting.

Module: USE AVVIEWER

Syntax

favGetAxisAutoScale (*hv, autoScale, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

autoScale

An output argument of type INTEGER(4). It returns TRUE if AxisAutoScale is enabled; otherwise, FALSE.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetAxisAutoScale](#), [favSetXClamp](#), [favSetYClamp](#), [favSetZClamp](#)

favGetAxisLabel

AVVIEWER Subroutine: Gets the axis label of the given axis.

Module: USE AVVIEWER

Syntax

favGetAxisLabel (*hv*, *axis*, *label*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the label. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

label

An input argument of type CHARACTER string. It returns the label associated with the given axis. The returned string is blank-filled at the end.

If the character length of this argument is less than the length of the stored label, the action fails and a status code of -1 is returned. If you do not know the length of the label, use a CHARACTER string of length AV_MAX_LABEL_LEN.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetAxisLabel](#)

favGetAxisStyle

AVVIEWER Subroutine: Gets the current axis style.

Module: USE AVVIEWER**Syntax**

favGetAxisStyle (*hv*, *nstyle*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

nstyle

An input argument of type INTEGER(2). It returns the current axis style.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetAxisStyle](#)

favGetCameraCoi

AVVIEWER Subroutine: Gets the current camera Center of Interest (COI).

Module: USE AVVIEWER**Syntax**

favGetCameraCoi (*hv*, *xpos*, *ypos*, *zpos*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

xpos

An output argument of type REAL(4). It returns the X component of the camera COI.

ypos

An output argument of type REAL(4). It returns the Y component of the camera COI.

zpos

An output argument of type REAL(4). It returns the Z component of the camera COI.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetCameraCoi](#)

favGetCameraPosition

AVVIEWER Subroutine: Gets the current camera position.

Module: USE AVVIEWER

Syntax

favGetCameraPosition (*hv, xpos, ypos, zpos, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

xpos

An output argument of type REAL(4). It returns the X component of the camera Center of Interest (COI).

ypos

An output argument of type REAL(4). It returns the Y component of the camera COI.

zpos

An output argument of type REAL(4). It returns the Z component of the camera COI.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetCameraPosition](#)

favGetCellEditEnabled

AVVIEWER Subroutine: Gets the current Cell Edit state.

Module: USE AVVIEWER

Syntax

favGetCellEditEnabled (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). It returns 1 if Cell Editing is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetCellEditEnabled](#)

favGetCompIndex

AVVIEWER Subroutine: Gets the current component index values.

Module: USE AVVIEWER

Syntax

favGetCompIndex (*hv*, *XComp*, *YComp*, *ZComp*, *WComp*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

XComp

An output argument of type INTEGER(4). It returns the X-component index value.

YComp

An output argument of type INTEGER(4). It returns the Y-component index value.

ZComp

An output argument of type INTEGER(4). It returns the Z-component index value.

WComp

An output argument of type INTEGER(4). It returns the W-component index value.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetCompIndex](#)

favGetDataClamp

AVVIEWER Subroutine: Gets the data clamp setting.

Module: USE AVVIEWER

Syntax

favGetDataClamp (*hv*, *dataClamp*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dataClamp

An output argument of type INTEGER(4). It returns TRUE if data clamp is enabled; otherwise, FALSE.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

See Remarks under **favSetDataClamp**.

See Also

[favSetDataClamp](#)

favGetDataRefreshEnable

AVVIEWER Subroutine: Gets the setting of the Data...Refresh menu item.

Module: USE AVVIEWER

Syntax

favGetDataRefreshEnable (*hv*, *dataRefreshEnable*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dataRefreshEnable

An output argument of type INTEGER(4). It returns the Data...Refresh Enable setting.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetDataRefreshEnable](#)

favGetDataSelectEnable

AVVIEWER Subroutine: Gets the data selection setting.

Module: USE AVVIEWER

Syntax

favGetDataSelectEnable (*hv, dataSelectEnable, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dataSelectEnable

An output argument of type INTEGER(4). It returns 1 if Data Select is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetDataSelectEnable](#)

favGetDefaultFormat

AVVIEWER Subroutine: Gets the default field width and precision used for displaying data in the Data View window.

Module: USE AVVIEWER

Syntax

favGetDefaultFormat (*hv, fieldwidth, precision, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

fieldwidth

An input argument of type INTEGER(4). It returns the field width of the default format.

precision

An input argument of type INTEGER(4). It returns the precision of the default format.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetUseDefaultFormat](#)

favGetDepthcue

AVVIEWER Subroutine: Gets the depth cueing setting.

Module: USE AVVIEWER

Syntax

favGetDepthcue (*hv*, *depthCue*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

depthCue

An output argument of type INTEGER(4). It returns 1 if depth cueing is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetDepthcue](#)

favGetDimName

AVVIEWER Subroutine: Gets the name that's associated with the given dimension.

Module: USE AVVIEWER

Syntax

favGetDimName (*hv*, *dim*, *name*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dim

An output argument of type INTEGER(2). It specifies the dimension of the array. The dimension must be in the range 0 to rank, where *rank* is the rank of the currently loaded array.

name

An output argument of type CHARACTER string. It returns the label associated with the given

dimension. The returned string is blank-filled at the end.

If the character length of this argument is less than the length of the stored label, the action fails and a status code of -1 is returned. If you do not know the length of the label, use a CHARACTER string of length AV_MAX_LABEL_LEN.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetDimName](#)

favGetErrorNo

AVVIEWER Subroutine: Gets an error number.

Module: USE AVVIEWER

Syntax

favGetErrorNo (*hv*, *errorNo*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

errorNo

An output argument of type INTEGER(4). It returns the error number.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When a subroutine returns failure, **favGetErrorNo** can be used to get the error code. Calling this subroutine clears the error code.

favGetFieldWidth

AVVIEWER Subroutine: Gets the current field width in the Data View window.

Module: USE AVVIEWER

Syntax

favGetFieldWidth (*hv*, *width*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

width

An input argument of type INTEGER(4). It returns the current field width.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetFieldWidth](#)

favGetFixedAspect

AVVIEWER Subroutine: Gets the current fixed aspect ratio setting.

Module: USE AVVIEWER

Syntax

favGetFixedAspect (*hv*, *fixedAspect*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

fixedAspect

An output argument of type INTEGER(4). It returns 1 if Fixed Aspect Ratio is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetFixedAspect](#)

favGetFontAutoScale

AVVIEWER Subroutine: Gets the current FontAutoScale state.

Module: USE AVVIEWER

Syntax

favGetFontAutoScale (*hv, onoroff, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). It returns 1 if FontAutoScale is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetFontAutoScale](#)

favGetGraphStyle

AVVIEWER Subroutine: Gets the current graph style.

Module: USE AVVIEWER

Syntax**favGetGraphStyle** (*hv, graphStyle, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

graphStyle

An output argument of type INTEGER(4). It returns the current graph style.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetGraphStyle](#)

favGetGraphType

AVVIEWER Subroutine: Gets the current graph type.

Module: USE AVVIEWER

Syntax

favGetGraphType (*hv, graphType, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

graphType

An output argument of type INTEGER(4). It returns the current graph type.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetGraphType](#)

favGetGridDensity

AVVIEWER Subroutine: Gets the grid density value.

Module: USE AVVIEWER

Syntax**favGetGridDensity** (*hv, gridDensity, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

gridDensity

An output argument of type INTEGER(4). It returns the current grid density value.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetGridDensity](#)

favGetHiddenLine

AVVIEWER Subroutine: Gets the current hidden line state.

Module: USE AVVIEWER

Syntax

favGetHiddenLine (*hv*, *hiddenLine*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

hiddenLine

An output argument of type INTEGER(4). It returns 1 if Hidden Line removal is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetHiddenLine](#)

favGetHighLight

AVVIEWER Subroutine: Gets the current highlighting state.

Module: USE AVVIEWER

Syntax

favGetHighLight (*hv*, *highLight*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

highLight

An output argument of type INTEGER(4). It returns 1 if HighLighting is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetHighLight](#)

favGetImageFilter

AVVIEWER Subroutine: Gets the current image filtering settings.

Module: USE AVVIEWER

Syntax

favGetImageFilter (*hv, imageFilter, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

imageFilter

An output argument of type INTEGER(4). It returns 1 if Image Linear Filtering is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetImageFilter](#)

favGetImageOrientation

AVVIEWER Subroutine: Gets the current image orientation setting.

Module: USE AVVIEWER

Syntax

favGetImageOrientation (*hv, orientation, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

orientation

An output argument of type INTEGER(4). It returns the current Image Orientation setting.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetImageOrientation](#)

favGetLineSmooth

AVVIEWER Subroutine: Gets the line smoothing setting.

Module: USE AVVIEWER**Syntax**

favGetLineSmooth (*hv*, *lineSmooth*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

lineSmooth

An output argument of type INTEGER(4). It returns 1 if line smoothing is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetLineSmooth](#)

favGetModifiedFlag

AVVIEWER Subroutine: Gets the modified flag.

Module: USE AVVIEWER**Syntax**

favGetModifiedFlag (*hv*, *bModified*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

bModified

An output argument of type INTEGER(4). It returns 1 if the Array Viewer document is "dirty"; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

See Remarks under **favSetModifiedFlag**.

See Also

[favSetModifiedFlag](#)

favGetNumMajorTickmarks

AVVIEWER Subroutine: Gets the number of large tickmarks for the desired axis.

Module: USE AVVIEWER

Syntax

favGetNumMajorTickmarks (*hv, axis, num, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the tickmarks. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

num

An input argument of type INTEGER(2). It returns the number of large tickmarks.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetNumMajorTickmarks](#)

favGetNumMinorTickmarks

AVVIEWER Subroutine: Gets the number of small tickmarks for the desired axis.

Module: USE AVVIEWER

Syntax

favGetNumMinorTickmarks (*hv, axis, num, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the tickmarks. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

num

An input argument of type INTEGER(2). It returns the number of small tickmarks that are displayed between each set of large tickmarks.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetNumMinorTickmarks](#), [favGetNumMajorTickmarks](#)

favGetPaletteId

AVVIEWER Subroutine: Gets the color palette identifier.

Module: USE AVVIEWER

Syntax

favGetPaletteId (*hv*, *paletteId*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

paletteId

An output argument of type INTEGER(4). It returns the palette ID value.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetPaletteId](#), [favSetCustomPalette](#)

favGetPaletteRange

AVVIEWER Subroutine: Gets the range of data values that will be associated with the color palette.

Module: USE AVVIEWER

Syntax

favGetPaletteRange (*hv*, *minval*, *maxval*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

minval

An output argument of type REAL(8). It returns the lower bound palette range.

maxval

An output argument of type REAL(8). It returns the upper bound palette range.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetPaletteRange](#)

favGetPrecision

AVVIEWER Subroutine: Gets the current Data View precision value.

Module: USE AVVIEWER

Syntax

favGetPrecision (*hv*, *precision*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

precision

An output argument of type INTEGER(4). It returns the current precision setting.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetPrecision](#)

favGetRoiLb

AVVIEWER Subroutine: Gets the position number of the first element in the Region of Interest (ROI) of the specified dimension.

Module: USE AVVIEWER

Syntax

favGetRoiLb (*hv, dim, roiLb, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dim

An input argument of type INTEGER(2). It specifies the dimension of the array.

roiLb

An output argument of type INTEGER(4). It returns the first element in the ROI.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetRoiUb](#), [favSetRoi](#)

favGetRoiUb

AVVIEWER Subroutine: Gets the position number of the last element in the Region of Interest (ROI) of the specified dimension.

Module: USE AVVIEWER

Syntax

favGetRoiUb (*hv, dim, roiUb, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dim

An input argument of type INTEGER(2). It specifies the dimension of the array.

roiUb

An output argument of type INTEGER(4). It returns the last element in the ROI.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetRoiLb](#), [favSetRoi](#)

favGetRowCol

AVVIEWER Subroutine: Gets the current row and column position.

Module: USE AVVIEWER

Syntax

favGetRowCol (*hv*, *row*, *col*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

row

An output argument of type INTEGER(4). It returns the current row.

col

An output argument of type INTEGER(4). It returns the current column.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetRowCol](#)

favGetRowColDim

AVVIEWER Subroutine: Gets the row and column dimensions.

Module: USE AVVIEWER

Syntax

favGetRowColDim (*hv*, *rowdim*, *coldim*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

rowdim

An output argument of type INTEGER(2). It returns the row dimension of the current array.

coldim

An output argument of type INTEGER(2). It returns the column dimension of the current array.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call.

The value 0 indicates success.

Remarks

For an explanation of how row and column dimensions are used in Array Viewer, see **favSetRowColDim**.

See Also

[favSetRowColDim](#)

favGetShading

AVVIEWER Subroutine: Gets the current shading state.

Module: USE AVVIEWER

Syntax

favGetShading (*hv, shading, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

shading

An output argument of type INTEGER(4). It returns 1 if shading is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetShading](#)

favGetShowAxis

AVVIEWER Subroutine: Gets the show axis setting.

Module: USE AVVIEWER

Syntax

favGetShowAxis (*hv, axis, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An output argument of type INTEGER(4). It returns 1 if the graph axis is displayed; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetShowAxis](#)

favGetShowDimLabels

AVVIEWER Subroutine: Gets the current ShowDimLabels state.

Module: USE AVVIEWER

Syntax

favGetShowDimLabels (*hv, show, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

show

An input argument of type INTEGER(4). It returns 1 if ShowDimLabels is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetShowDimLabels](#)

favGetShowGrid

AVVIEWER Subroutine: Gets the show grid setting.

Module: USE AVVIEWER

Syntax

favGetShowGrid (*hv, showGrid, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

showGrid

An output argument of type INTEGER(4). It returns 1 if grid lines are displayed; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetShowGrid](#)

favGetShowMarker

AVVIEWER Subroutine: Gets the current show marker state.

Module: USE AVVIEWER

Syntax

favGetShowMarker (*hv, marker, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

marker

An output argument of type INTEGER(4). It returns 1 if show marker is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetShowMarker](#)

favGetShowPalette

AVVIEWER Subroutine: Gets the current setting of show color palette.

Module: USE AVVIEWER

Syntax

favGetShowPalette (*hv, showPalette, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

showPalette

An output argument of type INTEGER(4). It returns TRUE if ShowColorPalette is enabled; otherwise, FALSE.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetShowPalette](#)

favGetTextureMode

AVVIEWER Subroutine: Gets the current texture mode state.

Module: USE AVVIEWER

Syntax

favGetTextureMode (*hv, mode, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

mode

An output argument of type INTEGER(2). It returns 1 if texture mode is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetTextureMode](#)

favGetUseAxisLabel

AVVIEWER Subroutine: Gets the current use axis label state for the given axis.

Module: USE AVVIEWER

Syntax

favGetUseAxisLabel (*hv, axis, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the label. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

onoroff

An input argument of type INTEGER(4). It returns 1 if display of axis labels is enabled for the given axis; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetUseAxisLabel](#)

favGetUseColorPalette

AVVIEWER Subroutine: Gets the current setting of use color palette.

Module: USE AVVIEWER

Syntax

favGetUseColorPalette (*hv*, *usePalette*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

usePalette

An output argument of type INTEGER(4). It returns the value 1 if UseColorPalette is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetUseColorPalette](#)

favGetUseDefaultFormat

AVVIEWER Subroutine: Gets the current UseDefaultFormat state.

Module: USE AVVIEWER

Syntax

favGetUseDefaultFormat (*hv, use, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

use

An input argument of type INTEGER(4). It returns 1 if UseDefaultFormat is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetUseDefaultFormat](#)

favGetUseExp

AVVIEWER Subroutine: Gets the current UseExp state.

Module: USE AVVIEWER

Syntax

favGetUseExp (*hv, use, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

use

An input argument of type INTEGER(4). It returns 1 if UseExp is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetUseExp](#)

favGetUseHex

AVVIEWER Subroutine: Gets the current UseHex state.

Module: USE AVVIEWER**Syntax****favGetUseHex** (*hv, use, status*)*hv*A handle that references an instance of Array Viewer created by [favStartViewer](#).*use*

An input argument of type INTEGER(4). It returns 1 if UseHex is enabled; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also[favSetUseHex](#)**favGetXClamp****AVVIEWER Subroutine:** Gets the upper and lower bounds for X-coordinate values in the Graph View.**Module: USE AVVIEWER****Syntax****favGetXClamp** (*hv, minval, maxval, status*)*hv*A handle that references an instance of Array Viewer created by [favStartViewer](#).*minval*

An output argument of type REAL(8). It returns the lower bound for the X axis.

maxval

An output argument of type REAL(8). It returns the upper bound for the X axis.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

RemarksSee Remarks under **favSetXClamp**.

See Also

[favSetXClamp](#)

favGetYClamp

AVVIEWER Subroutine: Gets the upper and lower bounds for Y-coordinate values in the Graph View.

Module: USE AVVIEWER

Syntax

favGetYClamp (*hv, minval, maxval, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

minval

An output argument of type REAL(8). It returns the lower bound for the Y axis.

maxval

An output argument of type REAL(8). It returns the upper bound for the Y axis.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

See Remarks under **favSetYClamp**.

See Also

[favSetYClamp](#)

favGetZClamp

AVVIEWER Subroutine: Gets the upper and lower bounds for Z-coordinate values in the Graph View.

Module: USE AVVIEWER

Syntax

favGetZClamp (*hv, minval, maxval, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

minval

An output argument of type REAL(8). It returns the lower bound for the Z axis.

maxval

An output argument of type REAL(8). It returns the upper bound for the Z axis.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

See Remarks under **favSetZClamp**.

See Also

[favSetZClamp](#)

favGetZScale

AVVIEWER Subroutine: Gets the Z scale value.

Module: USE AVVIEWER

Syntax

favGetZScale (*hv*, *scale*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

scale

An output argument of type REAL(4). It returns the current ZScale value.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetZScale](#)

favIsVisible

AVVIEWER Subroutine: Returns the Array Viewer visibility state.

Module: USE AVVIEWER**Syntax**

favIsVisible (*hv*, *bVisible*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

bVisible

An output argument of type INTEGER(4). It returns 1 if the Array Viewer window is visible; otherwise, 0.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If the user closes the Array Viewer window while the program that invoked it is still active, the Array Viewer is still running, but is not visible on the desktop. This routine can be used to determine if the user has closed the Array Viewer.

See Also

[favShowWindow](#)

favSetAnnotation

AVVIEWER Subroutine: Sets the annotation string.

Module: USE AVVIEWER**Syntax**

favSetAnnotation (*hv*, *string*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

string

The character string to be displayed in the Array Viewer's annotation dialog.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

favSetArray

AVVIEWER Subroutine: Lets you view a specified array.

Module: USE AVVIEWER

Syntax

favSetArray (*hv*, *array*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

array

The name of an array. The name must be one that the application has previously used in a call to **faglStartWatch**.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[faglStartWatch](#)

favSetArrayName

AVVIEWER Subroutine: Sets the character string to be displayed on the Array Viewer title bar.

Module: USE AVVIEWER

Syntax

favSetArrayName (*hv*, *name*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

name

The character string to be displayed on the title bar.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

favSetAxisAutoDetail

AVVIEWER Subroutine: Sets a value that determines whether the number of both large and small tick marks on the axes is to be handled automatically by Array Visualizer, or explicitly by the use of other properties.

Module: USE AVVIEWER

Syntax

favSetAxisAutoDetail (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to 1, AxisAutoDetail is enabled; if set to 0, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When AxisAutoDetail is set to FALSE, **favSetNumMajorTickmarks** and **favSetNumMinorTickmarks** can be used to explicitly control the number of tickmarks.

See Also

[favGetAxisAutoDetail](#), [favSetNumMajorTickmarks](#), [favSetNumMinorTickmarks](#)

favSetAxisAutoScale

AVVIEWER Subroutine: Determines whether axis auto scaling is automatically adjusted to the data range.

Module: USE AVVIEWER

Syntax

favSetAxisAutoScale (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the axis scales will be adjusted to the minimum and maximum data values within the Region of Interest (ROI). If set to FALSE, the

XMinClamp, XMaxClamp, YMinClamp, YMaxClamp, ZMinClamp, and ZMaxClamp values will be used.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetAxisAutoScale](#), [favSetXClamp](#), [favSetYClamp](#), [favSetZClamp](#)

favSetAxisLabel

AVVIEWER Subroutine: Sets the label to be displayed along the Graph View's axis.

Module: USE AVVIEWER

Syntax

favSetAxisLabel (*hv*, *axis*, *label*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the label. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

label

An input argument of type CHARACTER string. It is the label to be displayed along the axis. Only AV_MAX_LABEL_LEN characters of the label are used.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The string used in **favSetAxisLabel** will not be displayed if the given axis is mapped to either the row or column dimensions, and SetUseAxisLabel has been disabled.

See Also

[favSetRowColDim](#), [favSetUseAxisLabel](#), [favGetAxisLabel](#)

favSetAxisStyle

AVVIEWER Subroutine: Sets the current axis style.

Module: USE AVVIEWER**Syntax**

favSetAxisStyle (*hv, nstyle, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

nstyle

An input argument of type INTEGER(2). It specifies a value that can be used to select among several different axis styles.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

See the **AxisStyle** Avis2D property for a description of the different axis styles.

See Also

[AxisStyle property](#), [favGetAxisStyle](#)

favSetCameraCoi

AVVIEWER Subroutine: Sets the camera center of interest (COI).

Module: USE AVVIEWER**Syntax**

favSetCameraCoi (*hv, xpos, ypos, zpos, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

xpos

An input argument of type REAL(4). It specifies the X position.

ypos

An input argument of type REAL(4). It specifies the Y position.

zpos

An input argument of type REAL(4). It specifies the Z position.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine defines the point that the camera is looking to.

See Also

[favGetCameraCoi](#), [favSetCameraPosition](#)

favSetCameraPosition

AVVIEWER Subroutine: Sets the camera position.

Module: USE AVVIEWER

Syntax

favSetCameraPosition (*hv*, *xpos*, *ypos*, *zpos*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

xpos

An input argument of type REAL(4). It specifies the X position.

ypos

An input argument of type REAL(4). It specifies the Y position.

zpos

An input argument of type REAL(4). It specifies the Z position.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine can be used to adjust the virtual camera position for 3D graphs (that is, Height Plot and Vector Graphs with 3 or 4 components). The graph is defined as existing within a volume defined by (0, 0, 0) and (1, 1, 1).

For example, if the camera position is set to (0.5, 0.5, 2.0), and the center of interest (the position which the camera is pointed toward is (0.5, 0.5, 0.0), the viewpoint will appear to be directly over the graph.

See Also

[favGetCameraPosition](#), [favSetCameraCoi](#)

favSetCellEditEnabled

AVVIEWER Subroutine: Enables or disables Cell Editing in the Data View window.

Module: USE AVVIEWER

Syntax

favSetCellEditEnabled (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to 1, Cell Editing is enabled; if set to 0, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When Cell Editing is enabled, double-clicking on a cell turns the cell into edit mode, so you can edit the cell. When disabled, double-clicking does not turn the cell into edit mode.

See Also

[favGetCellEditEnabled](#)

favSetCompIndex

AVVIEWER Subroutine: Sets the component index values.

Module: USE AVVIEWER

Syntax

favSetCompIndex (*hv, XComp, YComp, ZComp, WComp, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

XComp

An input argument of type INTEGER(4). It specifies the X component index.

YComp

An input argument of type INTEGER(4). It specifies the Y component index.

ZComp

An input argument of type INTEGER(4). It specifies the Z component index.

WComp

An input argument of type INTEGER(4). It specifies the W component index.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Each index value must be an integer between 0 and n , where n is the number of columns (that is, the extent of the c dimension of the array, where c is the current column dimension). If a value of 0 is used, that component is not included in the graph. Otherwise, the vector component is derived from the given array index.

In Vector Graph mode, data values are graphed as a sequence of $\langle x, y, z, w \rangle$ vectors, where each vector component is extracted from the array based on the component index values.

See Also

[favGetCompIndex](#), [favGetRowColDim](#)

favSetCustomPalette

AVVIEWER Subroutine: Creates a custom palette.

Module: USE AVVIEWER

Syntax

favSetCustomPalette (*hv*, *paletteData*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

paletteData

A reference to an INTEGER(4) array of 256 elements that defines the custom palette.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The palette won't be displayed in the Graph View until **favSetPaletteId** (USER_DEFINED) is called.

See Also

[favSetPaletteId](#)

favSetDataClamp

AVVIEWER Subroutine: Enables or disables data clamping.

Module: USE AVVIEWER

Syntax

favSetDataClamp (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, data clamping is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Data clamping determines whether the Graph View clamps data values that are outside the clamp range.

If data clamping is enabled, data values will be "clamped" to the clamp range. If data clamping is disabled and array values within the Region of Interest (ROI) extend beyond the clamp range, the Graph View will "clip" out data points outside the clamp range.

See Also

[favSetXClamp](#), [favSetYClamp](#), [favSetZClamp](#), [favSetAxisAutoScale](#), [favGetDataClamp](#)

favSetDataRefreshEnable

AVVIEWER Subroutine: Enables or disables the Data...Refresh menu item.

Module: USE AVVIEWER

Syntax

favSetDataRefreshEnable (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the Data...Refresh menu item is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

You can use the Data...Refresh menu item to update your display to reflect the current array data.

See Also

[favGetDataRefreshEnable](#)

favSetDataSelectEnable

AVVIEWER Subroutine: Enables or disables data selection.

Module: USE AVVIEWER

Syntax

favSetDataSelectEnable (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, data selection is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Turning data selection off disables the tooltip windows.

See Also

[favGetDataSelectEnable](#)

favSetDepthcue

AVVIEWER Subroutine: Enables or disables depth cueing.

Module: USE AVVIEWER

Syntax

favSetDepthcue (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, depth cueing is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When enabled, depth cueing fades out more distant parts of the graph to provide an enhanced perception of depth. Depthcue has no effect unless the GraphType property is HeightField.

See Also

[favGetDepthcue](#), [favGetGraphType](#)

favSetDimName

AVVIEWER Subroutine: Associates a name with the given dimension.

Module: USE AVVIEWER

Syntax

favSetDimName (*hv, dim, name, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dim

An input argument of type INTEGER(2). It specifies the array dimension that the following argument will be applied to. The dimension must be in the range 0 to rank, where *rank* is the rank of the currently loaded array.

name

An input argument of type CHARACTER string. It is the label to be associated with the given dimension. Only AV_MAX_LABEL_LEN characters of the label are used.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The Graph View displays the dimension name as an axis label when the given dimension is assigned as either a row or column dimension, and SetUseAxisLabel has been set to FALSE for the indicated axis.

The Data View will display the dimension names as long as SetShowDimLabels has been enabled.

See Also

[favSetRowColDim](#), [favSetUseAxisLabel](#), [favGetDimName](#)

favSetDimScale

AVVIEWER Subroutine: Associates an axis dimension scale with a dimension.

Module: USE AVVIEWER

Syntax

favSetDimScale (*hv*, *dim*, *scaleData*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dim

An input argument of type INTEGER(2). It specifies the array dimension that the axis scale will be applied to. The dimension must be in the range 0 to rank, where *rank* is the rank of the currently loaded array.

scaleData

A reference to a one-dimensional array of type INTEGER(4) that defines the axis scale values.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The array pointed to by *scaleData* must have previously been used as an argument to **favStartWatch**.

See Also

[favStartWatch](#)

favSetFieldWidth

AVVIEWER Subroutine: Sets a value that determines the width allocated to display each data value in the Data View window.

Module: USE AVVIEWER

Syntax

favSetFieldWidth (*hv*, *width*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

width

An input argument of type INTEGER(4). It specifies the field width to be used.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

favSetFieldWidth is ignored if the UseDefaultFormat property is TRUE. It is also ignored if UseHex is TRUE when displaying integer data, or if UseExp is TRUE when displaying floating-point data.

See Also

[favGetUseDefaultFormat](#), [favGetUseExp](#), [favGetUseHex](#), [favGetFieldWidth](#)

favSetFileName

AVVIEWER Subroutine: Loads a specified file.

Module: USE AVVIEWER

Syntax

favSetFileName (*hv, filename, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

filename

The path to the file to be loaded by this Viewer instance.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

favSetFixedAspect

AVVIEWER Subroutine: Enables or disables the fixed aspect ratio setting.

Module: USE AVVIEWER

Syntax

favSetFixedAspect (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, Fixed Aspect Ratio is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

For Image Map graphs, when Fixed Aspect Ratio is enabled, the relative lengths of the X and Y axis don't change as the graph is resized. If it is disabled, the X and Y axis will be independently scaled to fit the view area.

See Also

[favGetFixedAspect](#)

favSetFontAutoScale

AVVIEWER Subroutine: Sets a value that determines whether the font size used in axis labels changes as the size of the graph view window changes, or is held fixed.

Module: USE AVVIEWER**Syntax**

favSetFontAutoScale (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to 1, FontAutoScale is enabled; if set to 0, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetFontAutoScale](#)

favSetGraphStyle

AVVIEWER Subroutine: Sets the graph style (mesh, surface, barchart, lines, or points).

Module: USE AVVIEWER**Syntax**

favSetGraphStyle (*hv, graphStyle, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

graphStyle

An input argument of type INTEGER(4). It specifies the desired graph style, which must be one of the graph style parameters defined in the AVVIEWER module.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Some graph styles are not valid for particular graph types. For a description of the different graph types, see Array Viewer online help.

See Also

[favGetGraphStyle](#)

favSetGraphType

AVVIEWER Subroutine: Sets the graph type (Height Plot, Image Map, Vector, or Plane View).

Module: USE AVVIEWER

Syntax

favSetGraphType (*hv*, *graphType*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

graphType

An input argument of type INTEGER(4). It specifies the desired graph type, which must be one of the graph type parameters defined in the AVVIEWER module.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Some graph types are not valid for particular arrays. For a description of the different graph types, see Array Viewer online help.

See Also

[favGetGraphType](#)

favSetGridDensity

AVVIEWER Subroutine: Sets a value that determines how finely the grid is drawn on the graph surface.

Module: USE AVVIEWER

Syntax

favSetGridDensity (*hv*, *newVal*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

newVal

An input argument of type REAL(4). It specifies a floating-point value that represents the desired number of pixels between adjacent grid lines.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine has no effect unless the ShowGrid value is TRUE and the GraphStyle is Surface.

See Also

[favGetGridDensity](#), [favGetGraphStyle](#)

favSetHiddenLine

AVVIEWER Subroutine: Enables or disables hidden line removal.

Module: USE AVVIEWER

Syntax

favSetHiddenLine (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, HiddenLine removal is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When hidden line removal is enabled (on), and the graph style is Mesh or Line, lines are hidden if they lie behind another part of the graph. Otherwise, the entire line segment is drawn.

See Also

[favGetGraphStyle](#), [favGetHiddenLine](#)

favSetHighLight

AVVIEWER Subroutine: Enables or disables highlighting.

Module: USE AVVIEWER

Syntax

favSetHighLight (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, HighLighting is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If highlighting is enabled, the graph surface will display highlights that change as the graph is rotated (that is, it will look as if the graph was made of a shiny material). If it is disabled, highlights won't be displayed.

See Also

[favGetHighLight](#)

favSetHomePosition

AVVIEWER Subroutine: Sets the current camera position as the home position.

Module: USE AVVIEWER

Syntax

favSetHomePosition (*hv, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetCameraPosition](#), [favToHomePosition](#)

favSetImageFilter

AVVIEWER Subroutine: Enables or disables image linear filtering.

Module: USE AVVIEWER

Syntax

favSetImageFilter (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, Image Linear Filtering is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When enabled, Image Linear Filtering smoothes the color values of each data point in the graph by blending the color with colors of neighboring points.

See Also

[favGetImageFilter](#)

favSetImageOrientation

AVVIEWER Subroutine: Sets the image orientation (IDENTITY, XFLIP, YFLIP, XYFLIP).

Module: USE AVVIEWER

Syntax

favSetImageOrientation (*hv, orientation, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

orientation

An input argument of type INTEGER(4). It specifies the new Image Orientation settings.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

In the Image Map graph type, the default orientation (IDENTITY) of the image is for the (0,0) element to be placed in the lower left corner of the graph. Other orientations can be set by using values XFLIP, YFLIP, or XYFLIP.

See Also

[favGetGraphType](#), [favGetImageOrientation](#)

favSetLineSmooth

AVVIEWER Subroutine: Enables or disables line smoothing.

Module: USE AVVIEWER

Syntax

favSetLineSmooth (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, line smoothing is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When enabled, line smoothing helps reduce the "jaggedness" of lines that are drawn in the graph window.

See Also

[favGetLineSmooth](#)

favSetModifiedFlag

AVVIEWER Subroutine: Sets the modified flag.

Module: USE AVVIEWER**Syntax**

favSetModifiedFlag (*hv, bModified, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

bModified

An input argument of type INTEGER(4). If set to nonzero, the modified flag is set; otherwise, the modified flag is cleared.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

When a change in the Array Viewer state is made, either by a fav routine or user interaction, the Viewer sets an internal flag noting that the state has been modified. If the flag is set, and the user closes the Viewer window, a dialog is displayed asking if the changes should be saved. This method can be used to modify the current value of the modified flag.

For example, if you do not want the user prompted to save changes, you can use the following call after making any fav call that modifies the Array Viewer state:

favSetModifiedFlag(hv, 0, status)

See Also

[favGetModifiedFlag](#)

favSetNumMajorTickmarks

AVVIEWER Subroutine: Sets the number of large tickmarks displayed along the indicated axis.

Module: USE AVVIEWER**Syntax**

favSetNumMajorTickmarks (*hv, axis, num, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the tickmarks. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

num

An input argument of type INTEGER(2). It specifies the number of large tickmarks to be displayed.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This method will not effect the Graph View appearance until AxisAutoDetail is disabled.

See Also

[favSetAxisAutoDetail](#), [favGetNumMajorTickmarks](#), [favSetNumMinorTickmarks](#)

favSetNumMinorTickmarks

AVVIEWER Subroutine: Sets the number of small tickmarks for the desired axis.

Module: USE AVVIEWER

Syntax

favSetNumMinorTickmarks (*hv*, *axis*, *num*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the tickmarks. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

num

An input argument of type INTEGER(2). It specifies the number of small tickmarks to be displayed between each set of large tickmarks.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This method will not effect the Graph View appearance until AxisAutoDetail is disabled.

See Also

[favSetAxisAutoDetail](#), [favGetNumMinorTickmarks](#), [favSetNumMajorTickmarks](#)

favSetPaletteAutoAdjust

AVVIEWER Subroutine: Determines whether the palette range is automatically adjusted to the data range.

Module: USE AVVIEWER

Syntax

favSetPaletteAutoAdjust (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the palette range will be set to the minimum and maximum data values within the Region of Interest (ROI). If set to FALSE, the values defined by **favSetPaletteRange** will be used.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetPaletteRange](#)

favSetPaletteId

AVVIEWER Subroutine: Sets the color palette.

Module: USE AVVIEWER

Syntax

favSetPaletteId (*hv*, *paletteId*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

paletteId

An input argument of type INTEGER(4). It specifies the palette identifier, which must be one of the standard palette IDs defined in the AVVIEWER module.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetPaletteId](#), [favSetCustomPalette](#)

favSetPaletteRange

AVVIEWER Subroutine: Sets the range of data values that the color palette will be associated with.

Module: USE AVVIEWER

Syntax

favSetPaletteRange (*hv*, *minval*, *maxval*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

minval

An input argument of type REAL(8). It specifies the minimum value in the data range.

maxval

An input argument of type REAL(8). It specifies the maximum value in the data range.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine only affects the graph's appearance when PaletteAutoAdjust is disabled.

See Also

[favGetPaletteRange](#), [favSetPaletteAutoAdjust](#)

favSetPrecision

AVVIEWER Subroutine: Sets the precision for floating-point numbers displayed in the Data View.

Module: USE AVVIEWER

Syntax

favSetPrecision (*hv, newVal, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

newVal

An input argument of type INTEGER(4). It specifies the number of digits to be displayed to the right of the decimal point.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetPrecision](#)

favSetRoi

AVVIEWER Subroutine: Sets the position number of the first and last elements in the Region of Interest (ROI) of the specified array dimension.

Module: USE AVVIEWER

Syntax**favSetRoi** (*hv, dim, roilb, roiub, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

dim

An input argument of type INTEGER(2). It specifies the array dimension that the following two arguments will be applied to. The dimension must be in the range 0 to rank, where *rank* is the rank of the currently loaded array.

roilb

An input argument of type INTEGER(4). It specifies the lower array index to be included in the ROI. The array index must be in the range lbound to ubound, where *lbound* and *ubound* are the lower bound and upper bound of dimension *dim*.

roiub

An input argument of type INTEGER(4). It specifies the upper array index to be included in the ROI. The array index must be in the range lbound to ubound, where *lbound* and *ubound* are the lower bound and upper bound of dimension *dim*.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call.

The value 0 indicates success.

Remarks

This subroutine can be used to select and view a subregion of the current array. You must call the **favUpdate** subroutine before the new ROI is mapped to the graph.

See Also

[favUpdate](#)

favSetRoi2D

AVVIEWER Subroutine: Sets the Region of Interest (ROI) within the current 2D array "slice".

Module: USE AVVIEWER

Syntax

favSetRoi2D (*hv, col_start, row_start, num_cols, num_rows, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

col_start

An input argument of type INTEGER(4). It specifies the lower array index of the current column dimension to be included in the ROI.

row_start

An input argument of type INTEGER(4). It specifies the lower array index of the current row dimension to be included in the ROI.

num_cols

An input argument of type INTEGER(4). It specifies the number of columns to be included in the ROI.

num_rows

An input argument of type INTEGER(4). It specifies the number of rows to be included in the ROI.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine is similar to the [favSetRoi](#) subroutine, which also updates the current ROI. However, **favSetRoi2D** is more convenient to use in cases where the current 2D slice of the array is not being

changed.

favSetRowCol

AVVIEWER Subroutine: Sets the row and column position.

Module: USE AVVIEWER

Syntax

favSetRowCol (*hv, row, col, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

row

An input argument of type INTEGER(4). It specifies the new row index. The row should be in the range 0 to $n-1$, where n is the number of rows in the current array.

col

An input argument of type INTEGER(4). It specifies the new column index. The column should be in the range 0 to $n-1$, where n is the number of columns in the current array.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Setting the row and column position will update both the Data and Graph Views.

See Also

[favGetRowCol](#)

favSetRowColDim

AVVIEWER Subroutine: Sets the row and column dimensions.

Module: USE AVVIEWER

Syntax

favSetRowColDim (*hv, rowdim, coldim, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

rowdim

An input argument of type INTEGER(2). It specifies the array dimension to be viewed as rows.

coldim

An input argument of type INTEGER(2). It specifies the array dimension to be viewed as columns.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

For arrays of rank 3 or higher, the Array Viewer displays the array data as a 2D sub-array of the higher dimension array. The *rowdim* and *coldim* values define which two dimensions of the array get mapped as rows and columns, respectively.

For arrays of rank 2, this subroutine can be used to control which dimension gets mapped as rows. For arrays of rank 1, this subroutine has no effect.

See Also

[favGetRowColDim](#)

favSetShading

AVVIEWER Subroutine: Enables or disables shading.

Module: USE AVVIEWER

Syntax

favSetShading (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, shading is enabled; if set to FALSE, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The shading state determines how color values are interpolated across the graph's surface. If shading

is enabled, the colors are interpolated between neighboring data points. If shading is disabled, each data element of the graph is drawn in a solid color.

See Also

[favGetShading](#)

favSetShowAxis

AVVIEWER Subroutine: Displays or hides the graph axis.

Module: USE AVVIEWER

Syntax

favSetShowAxis (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the graph axis will be displayed; if set to FALSE, the axis will be hidden.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetShowAxis](#)

favSetShowDimLabels

AVVIEWER Subroutine: Sets a value that determines whether to display the column and row dimension labels in the Data View window.

Module: USE AVVIEWER

Syntax

favSetShowDimLabels (*hv*, *show*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

show

An input argument of type INTEGER(4). If set to 1, dimension labels are displayed; if set to 0,

they are hidden.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetShowDimLabels](#)

favSetShowGrid

AVVIEWER Subroutine: Displays or hides grid lines.

Module: USE AVVIEWER

Syntax

favSetShowGrid (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, grid lines will be displayed in the graph; if set to FALSE, they will not be displayed.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine applies only to graphs whose GraphStyle is Surface.

See Also

[favGetShowGrid](#), [favGetGraphStyle](#)

favSetShowMarker

AVVIEWER Subroutine: Displays or hides the marker.

Module: USE AVVIEWER

Syntax

favSetShowMarker (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the marker will be displayed; if set to FALSE, it will be hidden.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetShowMarker](#)

favSetShowPalette

AVVIEWER Subroutine: Displays or hides the color palette.

Module: USE AVVIEWER

Syntax

favSetShowPalette (*hv, onoroff, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the current color palette will be displayed in the Graph View. If set to FALSE, the palette will not be displayed.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetShowPalette](#)

favSetTextureMode

AVVIEWER Subroutine: Enables or disables 1D textured color mapping.

Module: USE AVVIEWER

Syntax

favSetTextureMode (*hv, mode, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

mode

An input argument of type INTEGER(2). It specifies the desired texture mode. If set to 1, texture mapping is enabled; if set to 0, it is disabled.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

Texture mapping results in a more precise mapping of palette color values to the graph surface, but may result in slower graph updates. The visual difference between different TextureMode values is most noticeable when the number of data points in the graph is small.

TextureMode has no effect if Shading is disabled.

See Also

[favGetTextureMode](#), [favGetShading](#)

favSetUseAxisLabel

AVVIEWER Subroutine: Enables or disables display of Axis labels.

Module: USE AVVIEWER

Syntax**favSetUseAxisLabel** (*hv, axis, onoroff, status*)*hv*

A handle that references an instance of Array Viewer created by [favStartViewer](#).

axis

An input argument of type INTEGER(4). It specifies the desired axis for the label. Use one of the named constants X_AXIS, Y_AXIS, or Z_AXIS.

onoroff

An input argument of type INTEGER(4). If set to 1, the axis label for the indicated dimension is displayed; if set to 0, the dimension name, if available, is displayed.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If the given axis doesn't correspond to an array dimension (for example: the Z-Axis in a Height Plot graph), the axis label will not be displayed.

See Also

[favGetUseAxisLabel](#), [favSetAxisLabel](#), [favSetDimName](#)

favSetUseColorPalette

AVVIEWER Subroutine: Enables or disables use of the color palette.

Module: USE AVVIEWER

Syntax

favSetUseColorPalette (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the current color palette will be used to map data values to colors. If set to FALSE, the current graph color will be used.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetPaletteRange](#), [favSetPaletteAutoAdjust](#), [favSetPaletteId](#), [favGetUseColorPalette](#)

favSetUseDefaultFormat

AVVIEWER Subroutine: Sets a value that determines whether to display data in the Data View window using the default format.

Module: USE AVVIEWER

Syntax

favSetUseDefaultFormat (*hv*, *use*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

use

An input argument of type INTEGER(4). If set to 1, data values will be displayed in the default format. If set to 0, the format can be controlled explicitly by using the **favSetUseHex**, **favSetFieldWidth**, and **favSetUseExp** routines.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favGetUseDefaultFormat](#), [favGetDefaultFormat](#), [favSetUseHex](#), [favSetFieldWidth](#), [favSetUseExp](#)

favSetUseExp

AVVIEWER Subroutine: Sets a value that determines whether to display floating-point data in scientific notation in the Data View window.

Module: USE AVVIEWER

Syntax

favSetUseExp (*hv*, *use*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

use

An input argument of type INTEGER(4). If set to 1, floating-point data is displayed in scientific notation; if set to 0, the data is displayed in decimal format.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The value passed to **favSetUseExp** is ignored when displaying integer data, or if UseDefaultFormat has been set to TRUE.

See Also

[favGetUseDefaultFormat](#), [favGetUseExp](#)

favSetUseHex

AVVIEWER Subroutine: Sets a value that determines whether to display integer data in hexadecimal or decimal format in the Data View window.

Module: USE AVVIEWER

Syntax

favSetUseHex (*hv, use, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

use

An input argument of type INTEGER(4). If set to 1, integer data is displayed in hexadecimal format; if set to 0, the data is displayed in decimal format.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

The value passed to **favSetUseHex** is ignored when displaying floating-point data, or if UseDefaultFormat has been set to TRUE.

See Also

[favGetUseDefaultFormat](#), [favGetUseHex](#)

favSetXClamp

AVVIEWER Subroutine: Sets the upper and lower bounds for X-coordinate values in the Graph View.

Module: USE AVVIEWER

Syntax

favSetXClamp (*hv, minval, maxval, status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

minval

An input argument of type REAL(8). It specifies the lower bound for the X axis.

maxval

An input argument of type REAL(8). It specifies the upper bound for the X axis.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine only affects the graph appearance when the graph type is Vector and AxisAutoScale is disabled.

See Also

[favSetDataClamp](#), [favSetGraphType](#), [favSetAxisAutoScale](#), [favGetXClamp](#)

favSetYClamp

AVVIEWER Subroutine: Sets the upper and lower bounds for Y-coordinate values in the Graph View.

Module: USE AVVIEWER

Syntax

favSetYClamp (*hv*, *minval*, *maxval*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

minval

An input argument of type REAL(8). It specifies the lower bound for the Y axis.

maxval

An input argument of type REAL(8). It specifies the upper bound for the Y axis.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine only affects the graph appearance when the graph type is Vector and AxisAutoScale is disabled.

See Also

[favSetDataClamp](#), [favSetGraphType](#), [favSetAxisAutoScale](#), [favGetYClamp](#)

favSetZClamp

AVVIEWER Subroutine: Sets the upper and lower bounds for Z-coordinate values in the Graph View.

Module: USE AVVIEWER

Syntax

favSetZClamp (*hv*, *minval*, *maxval*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

minval

An input argument of type REAL(8). It specifies the lower bound for the Z axis.

maxval

An input argument of type REAL(8). It specifies the upper bound for the Z axis.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine only affects the graph appearance when AxisAutoScale is disabled.

See Also

[favSetDataClamp](#), [favSetAxisAutoScale](#), [favGetZClamp](#)

favSetZScale

AVVIEWER Subroutine: Sets the height of the Z axis relative to the X and Y axes.

Module: USE AVVIEWER

Syntax

favSetZScale (*hv*, *scale*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

scale

An input argument of type REAL(4). It specifies the Z scaling factor.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

If *scale* is 1.0, the Z height will be equal to the X and Y axes. Other values will change the Z axis proportionally.

See Also

[favGetZScale](#)

favShowWindow

AVVIEWER Subroutine: Displays or hides the Array Viewer window.

Module: USE AVVIEWER

Syntax

favShowWindow (*hv*, *onoroff*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

onoroff

An input argument of type INTEGER(4). If set to TRUE, the window will be displayed; if set to FALSE, it will be hidden.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favIsVisible](#)

favStartViewer

AVVIEWER Subroutine: Creates an Array Viewer instance.

Module: USE AVVIEWER

Syntax

favStartViewer (*hv*, *status*)

hv

An output argument of type INTEGER(4). It receives the handle of the created Array Viewer object.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

Remarks

This subroutine constructs an Array Viewer application object. When calling other routines in the AVViewer module, use the handle returned by this call to reference this viewer instance.

The Viewer window is initially hidden. Use **favShowWindow** to make it visible. Use **favEndViewer** to close the viewer and destroy this Viewer instance.

See Also

[favShowWindow](#), [favEndViewer](#)

favToHomePosition

AVVIEWER Subroutine: Returns the camera position to the home position.

Module: USE AVVIEWER

Syntax

favToHomePosition (*hv*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

status

An output argument of type INTEGER(4). It returns a status code for the success of this call. The value 0 indicates success.

See Also

[favSetHomePosition](#)

favUpdate

AVVIEWER Subroutine: Forces Array Viewer to redraw its view to reflect any data changes since the last Update.

Module: USE AVVIEWER

Syntax

favUpdate (*hv*, *count*, *status*)

hv

A handle that references an instance of Array Viewer created by [favStartViewer](#).

count

An input argument of type INTEGER(4). You can pass 0 or 1 as the argument.

If the value is 0, **favUpdate** returns immediately.

If the value is 1, **favUpdate** won't return until the Array Viewer views have been updated.

status

An output argument of type INTEGER(4). It returns a status code for the success of this call.

The value 0 indicates success.

Remarks

Invoke this subroutine when the data or Region of Interest (ROI) has been modified and you want Array Viewer to reflect the new data.

API Calls for C Programmers: agl Routines

C and C++ programmers can use the following API calls (aglxxx routines) to incorporate the Array Viewer in their applications:

API Calls for C Programmers

aglAlloc	aglMallocEx
aglClose	aglName
aglEndWatch	aglReshape
aglFree	aglSaveAsFile
aglGetShareName	aglShow
aglHide	aglStartWatch
aglMalloc	aglUpdate

It is the nature of these APIs that typically you will call them in roughly the same sequence.

First, you call **aglAlloc**, **aglMalloc** or **aglMallocEx** to allocate memory to hold your data. Then you place data into that memory. Note that this produces and populates a one-dimensional array. This will always be the state of your array initially.

If you want your data to be multi-dimensional, you must call **aglReshape** to recategorize the array as one with the desired number of dimensions and with each dimension of the desired size. Even though **aglReshape** does not move any data, it does reclassify the array as one having the desired structure.

At any time until you return the array's memory to the system, you can save the array by calling **aglSaveAsFile**. This API saves the files in the Array Graphic Language (.AGL) format.

From here, displaying the data requires two calls. First, you must call **aglStartWatch** to alert Array Visualizer that the array has been through the required preliminary steps and is now ready to be displayed. The Array Visualizer places the array in a special class of ready-to-show arrays.

Next, by calling **aglShow**, the Array Visualizer launches a new instance of the Array Viewer program, with the data in it. At this point, the you see your data and can manipulate the data as desired.

If you want the title bar of the instance of Array Viewer to contain particular information, you can call **aglName** to place the appropriate text onto the title bar. Make this call any time after the call to **aglReshape** and before the first call to **aglShow**.

After the initial call to **aglShow**, you can make the instance of Array Viewer invisible by calling **aglHide** and then, if desired, call **aglShow** again.

One reason to hide and then reshow an instance of Array Viewer is to call **aglReshape** to alter how the data displays. For example, if the array is 8x3, you could change it to 6x4. Note that the new shape *must* contain exactly the same number of elements as the previous shape.

When it is time to stop displaying the data, call **aglClose** to shut down the instance of Array Viewer. Then call **aglEndWatch** to remove the array from the ready-to-show list.

Finally, call **aglFree** to return the allocated memory.

You can use **aglGetShareName** to get a string identifier that identifies the shared memory region associated with an array.

You can also call the [CaViewer Class Routines](#) to perform tasks usually done interactively by the user.

Samples of these routines are provided online in folders in `...\ArrayVisualizer\Samples\C\`. For a description of the Array Visualizer Samples, view the file `...\ArrayVisualizer\Samples\Samples.htm` in a Web browser. Most Samples include a project workspace file, which allows you to open and build the project in the visual development environment.

Samples are installed on the hard disk when a Complete installation is performed. You can copy Samples folders from the Array Visualizer CD-ROM to your hard disk (remove the read-only file property).

aglAlloc

Allocates a block of memory with a fixed shape.

```
int aglAlloc(  
    pvoid dims,  
    int numdims,  
    int datatype,  
);
```

Parameters

dims

A pointer to an array of Integers. The size of *dims* must be equal to *numdims*. The *dims* array specifies the dimensions of the array to be allocated.

numdims

The number of dimensions (at most 7) of the array.

datatype

The type of the data in the array, as shown in the following table. You can use the name on the left as a substitute for the number on the right.

Data Type	Code
AGL_UCHAR	4096
AGL_CHAR	4097
AGL_USHORT	4098
AGL_SHORT	4099
AGL_ULONG	4100
AGL_LONG	4101
AGL_FLOAT	4102
AGL_DOUBLE	4103
AGL_ULLONG	4104
AGL_LLONG	4105

Return Value

Returns a pointer to the allocated block of memory, or NULL if **aglAlloc** was unable to allocate the memory.

Remarks

Unlike **aglMalloc**, **aglAlloc** allocates a block of memory using arrays of pointers. This means that you can refer to multi-dimensional array elements using bracket notation rather than pointer arithmetic; for example: `M[i][j]` rather than `M[i*NumberOfCols + j]`. You should eventually deallocate the memory by calling **aglFree**.

Note: **aglReshape** cannot be used with this memory block to modify the array shape.

See Also

[aglMalloc](#)

aglClose

Closes an instance of Array Visualizer.

```
int aglClose(  
    pvoid AVInstance  
);
```

Parameters

AVInstance

A pointer to an array that has been allocated (using **aglAlloc** or **aglMalloc**) or added (using **aglStartWatch**) to the list of data arrays that you can display using **aglShow**.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

The **aglClose** function closes the instance of the Array Visualizer application (if any) associated with *AVInstance*. To create another instance of Array Visualizer, pass the pointer to **aglShow**.

See Also

[aglAlloc](#), [aglMalloc](#), [aglShow](#), [aglStartWatch](#)

aglEndWatch

Removes the specified array from the list of viewable arrays.

```
int aglEndWatch(  
    pvoid array  
);
```

Parameters

array

A pointer to array data that has previously been passed to **aglStartWatch**.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

The **aglEndWatch** function removes the given array from the list of viewable arrays and frees any associated resources used by the library. After **aglEndWatch**, the array must be passed to **aglStartWatch** before it can again be usable in calls such as **aglShow**, **aglReshape**, etc.

Before calling **aglEndWatch**, you might want to save the array as an .AGL file using **aglSaveAsFile**.

See Also

[aglSaveAsFile](#), [aglStartWatch](#)

aglFree

Deallocates or frees a memory block that had previously been allocated by a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**.

```
void aglFree(  
    pvoid array  
);
```

```
pvoid memblock
);
```

Parameters

memblock

A pointer to the memory being released.

Return Value

None.

See Also

[aglAlloc](#), [aglMalloc](#), [aglMallocEx](#)

aglGetShareName

Gets a string identifier that identifies the shared memory region associated with an array.

```
const char *aglGetShareName(
void *array
);
```

Parameters

array

A pointer to an array either allocated by means of a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**, or added to the watch list by means of a call to **aglStartWatch**.

Return Value

A string identifying the array.

Remarks

The string returned by **aglGetShareName** can be passed to the Avis2D or AvisGrid controls as the **FileName** property. This allows array data to be displayed by applications that host these controls without saving the array to a disk file.

See Also

[aglAlloc](#), [aglMalloc](#), [aglMallocEx](#), [aglStartWatch](#), Avis2D [FileName property](#), AvisGrid [FileName property](#)

aglHide

Causes the instance of Array Visualizer (if any) associated with *array* to become invisible.


```
int aglHide(  
    pvoid array  
);
```

Parameters

array

A pointer to an array either allocated by means of a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**, or added to the watch list by means of a call to **aglStartWatch**.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

The **aglHide** function causes the instance of Array Visualizer associated with *array* to become invisible, but it does not close that instance. To close the instance, call **aglClose** or **aglEndWatch**. To make the Array Visualizer instance visible, call **aglShow**. **aglHide** can be used with arrays created with **aglAlloc**.

If the Array Viewer instance associated with *array* was created by using the `CAViewer` class, rather than **aglShow**, this routine will have no effect. Use the [CAViewer::ShowWindow](#) method instead.

See Also

[aglAlloc](#), [aglClose](#), [aglEndWatch](#), [aglMalloc](#), [aglMallocEx](#), [aglShow](#), [aglStartWatch](#)

aglMalloc

Requests a specified amount of memory for use as a new data array.

```
voidp aglMalloc(  
    size_t numbytes  
);
```

Parameters

numbytes

The number of bytes to be allocated.

Return Value

Returns a void pointer to the memory block allocated, or NULL if available memory is not sufficient.

Remarks

Array Visualizer treats the allocated memory block as a one-dimensional array whose elements are of the type UCHAR. To change the structure or data type, call **aglReshape**.

The caller should eventually deallocate the memory by calling **aglFree**.

See Also

[aglFree](#), [aglMallocEx](#), [aglReshape](#)

aglMallocEx

Requests a specified amount of memory for use as a new data array.

```
voidp aglMallocEx(  
    size_t numbytes  
    const char *filename  
);
```

Parameters

numbytes

The number of bytes to be allocated.

filename

The path to a file to be used as a backing store for the requested memory block.

Return Value

Returns a void pointer to the memory block allocated, or NULL if available memory is not sufficient.

Remarks

Array Visualizer treats the allocated memory as a one-dimensional array whose elements are of type AGL_UCHAR. To change the structure or data type, call **aglReshape**. If *filename* is an existing .AGL file, the array data comes from that file.

The caller should eventually deallocate the memory by calling **aglFree**.

For large memory allocations the advantages of using **aglMallocEx**, instead of **aglMalloc**, are that memory is not allocated from the paging store, and the system periodically synchronizes the memory and disk images.

See Also

[aglFree](#), [aglMalloc](#), [aglReshape](#)

aglName

Places a specified string onto the title bar of an Array Viewer instance.

```
int aglName(  
    pvoid array,  
    const char *name  
);
```

Parameters

array

A pointer to an array either allocated by means of a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**, or added to the watch list by means of a call to **aglStartWatch**.

name

The string of characters to be placed in the title bar.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

If the Array Viewer instance associated with array was created by using the CAVIEWER class, rather than **aglShow**, this routine will have no effect. Use the [CAViewer::SetArrayName](#) method instead.

See Also

[aglAlloc](#), [aglMalloc](#), [aglStartWatch](#)

aglReshape

Provides Array Visualizer with information about how the given array data is to be interpreted as an array with particular dimensions and data of a particular type.

```
int aglReshape(  
    pvoid array,  
    int numdims,  
    int *dimsizes,  
    int datatype  
);
```

Parameters

array

A pointer to the array that is to be reshaped.

numdims

The number of dimensions (at most 7) that the reshaped array is to have.

dim sizes

Pointer to a one-dimensional array of Integers, where the first value contains the size of the first dimension, the second value contains the size of the second dimension, etc.

datatype

The type of the data in the array, as shown in the following table. You can use the name on the left as a substitute for the number on the right.

Data Type	Code
AGL_UCHAR	4096
AGL_CHAR	4097
AGL_USHORT	4098
AGL_SHORT	4099
AGL_ULONG	4100
AGL_LONG	4101
AGL_FLOAT	4102
AGL_DOUBLE	4103
AGL_ULLONG	4104
AGL_LLONG	4105

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

Caution: The product of the dimension sizes, when multiplied times the number of bytes required for each data element, must be exactly equal to the number of bytes in the array, as requested of, and allocated by **aglMalloc**. Otherwise, the error **AGL_ERROR_RESHAPE_INVALID** is returned and the call fails.

See Also

[aglMalloc](#)

aglSaveAsFile

Saves the current array as a .AGL file.

```
int aglSaveAsFile(
    pvoid array,
    int *filename
);
```

Parameters

array

A pointer to an array either allocated by means of a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**, or added to the watch list by means of a call to **aglStartWatch**.

filename

Either just the name of the file to be saved (not including the path) in the calling application's directory, or a complete path/filename combination to save the file in any other directory.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

If your end users are likely to use the array again with Array Visualizer, use this function to save it. The users can run Array Viewer as a stand-alone program to display the array. **aglSaveAsFile** can be used with arrays created with **aglAlloc**.

See Also

[aglAlloc](#), [aglMalloc](#), [aglShow](#), [aglStartWatch](#)

aglShow

Creates an instance of Array Visualizer to display data for the specified array.

```
int aglShow(  
    pvoid array  
);
```

Parameters

array

Pointer to an array either allocated by a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**, or added to the watch list by a call to **aglStartWatch**.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

The **aglShow** function creates an instance of the Array View and displays that instance. You can also use **aglShow** to make an existing instance of Array Visualizer visible if it has been rendered invisible by a call to **aglHide**. Before showing the instance of Array Visualizer, you might want to call

aglName to put a suitable heading on its title bar.

If the Array Viewer instance associated with array was created by using the `CAViewer` class, rather than **aglShow**, this routine will have no effect. Use the [CAViewer::ShowWindow](#) method instead.

See Also

[aglAlloc](#), [aglHide](#), [aglMalloc](#), [aglMallocEx](#), [aglName](#), [aglShow](#), [aglStartWatch](#)

aglStartWatch

Adds the specified array to the list of viewable arrays.

```
int aglStartWatch(  
    pvoid array,  
    size_t numbytes  
);
```

Parameters

array
Pointer to the data array that is to be made viewable.

numbytes
Total number of bytes in *array*.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

Array Visualizer treats the allocated memory block as a one-dimensional array whose elements are of the type `UCHAR`. To change the structure or data type, call **aglReshape**.

To create an instance of Array Visualizer that displays the array, call **aglShow** using the pointer to the array. Use **aglEndWatch** to remove the instance of Array Visualizer from the list of viewable arrays.

See Also

[aglEndWatch](#), [aglReshape](#), [aglShow](#)

aglUpdate

Re-synchronizes Array Visualizer's view of the array data with the actual data values.

```
int aglUpdate(  
    pvoid array
```

);

Parameters

array

Pointer to the array to be updated.

Return Value

Returns the value 0 if the call is successful; any other value indicates failure.

Remarks

If your application has modified the values in Array since the last **aglUpdate** or **aglShow** call, and you want Array Visualizer to reflect the new data values, you can do so by calling **aglUpdate**. **aglUpdate** can be used with arrays created with **aglAlloc**.

If the Array Viewer instance associated with array was created by using the CAViewer class, rather than **aglShow**, this routine will have no effect. Use the [CAViewer::Update](#) method instead.

See Also

[aglAlloc](#), [aglShow](#)

Examples

This section contains three simple C language example programs that illustrate the use of the C routine APIs (aglxxx routines):

- [Example 1](#)
- [Example 2](#)
- [Example 3](#)

Array Visualizer Sample programs are installed on your hard disk when you select a Complete installation. You can also copy Samples folders from the Array Visualizer CD-ROM to your hard disk.

Example 1

The first example uses the [aglMalloc](#) call to dynamically allocate memory for an array and then view the array using the Array Viewer. This example is a Sample program that can be found in the folder `...\ArrayVisualizer\Samples\C\malloc2d\`:

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
// avdef.h is the header file for the AView library
#include <avdef.h>
```

```

void
pause()
{
    printf("press any key to continue\n");
    _getch();
}

int
main(int argc, char *argv[])
{
    int dims[] = {40, 50};
    int rc, i, j;
    float x, y, z, rval, *fmat;
    const float fPi = 3.14159f;

    // Use aglMalloc to allocate memory for the array
    fmat = (float *)aglMalloc(dims[0]*dims[1]*sizeof(float));
    if (fmat == NULL) {
        printf("aglMalloc failed\n");
        return -1;
    }

    // Initialize the data
    for (j=0; j<dims[0]; j++) {
        y = (float)j/(float)dims[0];
        for (i=0; i<dims[1]; i++) {
            x = (float)i/(float)dims[1];
            z = (float)(sin(y*fPi) + cos(x*fPi));
            // Note that we have to do a bit of pointer arithmetic
            // to assign the (j, i) element of the array to z.
            *(fmat + j*dims[1]+ i) = z;
        }
    }

    // Call aglReshape to let the AView library know about the
    // array dimensions and type.
    rc = aglReshape(fmat, 2, dims, AGL_FLOAT);
    if (rc != 0) {
        printf("aglReshape failed!\n");
        return -1;
    }

    // Set the title bar on ArrayViewer
    aglName(fmat, "sin(x) + cos(y)");

    printf("Starting Array Viewer\n");
    // aglShow will bring up Array Viewer with a view of our array.
    rc = aglShow(fmat);
    if (rc != 0) {
        printf("aglShow failed\n");
        return -1;
    }
    pause();

    // Add some psudeo-random fluctuations
    for (j=0; j<dims[0]; j++) {
        y = (float)j/(float)dims[0];
        for (i=0; i<dims[1]; i++) {
            // produce a random number between 0 and 1.
            rval = (float)(rand() % 1000) * 0.001f;
            // Scale and shift to range: -0.1 to 0.1
            rval = rval * 0.2f - 0.1f;
            // Add to array element
            *(fmat + j*dims[1]+ i) += rval;
        }
    }
}

```



```

    }
}

// Inform the viewer that the array data has been changed.
printf("Updating data\n");
aglUpdate(fmat);

// Change the title to reflect the changes in the data set.
aglName(fmat, "sin(x) + cos(y) + noise");

// Wait for a key press
pause();

printf("Closing down the viewer\n");

// Close the Viewer
rc = aglClose(fmat);
if (rc != 0) {
    printf("aglClose failed\n");
    return -1;
}

pause();

// Deallocate memory
aglFree(fmat);

return 0;
}

```

Example 2

The second example uses the [aglAlloc](#) call to allocate memory for the array. This Sample can be found in the folder ...\\ArrayVisualizer\\Samples\\C\\alloc2d\\:

```

#include <windows.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <math.h>
// avdef.h is the header file for the AView library
#include <avdef.h>

void
pause()
{
    printf("press any key to continue\n");
    _getch();
}

int
main(int argc, char *argv[])
{
    int dims[] = {40, 50};
    int rc, i, j;
    float x, y, z, rval, **fMat;
    const float fPi = 3.14159f;

    // Use aglAlloc to allocate memory for the array.
    // This uses slightly more memory than using aglMalloc,
    // but makes it easier to reference the array elements.
    // Note that fMat is of type float **, ie a pointer to

```

```

// a pointer to a float.
fMat = (float **)aglAlloc(2, dims, AGL_FLOAT);
if (fMat == NULL) {
    printf("aglAlloc failed\n");
    return -1;
}

// Initialize the data
for (j=0; j<dims[0]; j++) {
    y = (float)j/(float)dims[0];
    for (i=0; i<dims[1]; i++) {
        x = (float)i/(float)dims[1];
        z = (float)(sin(y*fPi) + cos(x*fPi));
        fMat[j][i] = z;
    }
}

// Set the title bar on ArrayViewer
aglName(fMat, "sin(x) + cos(y)");

printf("Starting Array Viewer\n");
// aglShow will bring up Array Viewer with a view of our array.
rc = aglShow(fMat);
if (rc != 0) {
    printf("aglShow failed\n");
    return -1;
}
pause();

// Add some psudeo-random fluctuations
for (j=0; j<dims[0]; j++) {
    y = (float)j/(float)dims[0];
    for (i=0; i<dims[1]; i++) {
        // produce a random number between 0 and 1.
        rval = (float)(rand() % 1000) * 0.001f;
        // Scale and shift to range: -0.1 to 0.1
        rval = rval * 0.2f - 0.1f;
        // Add to array element
        fMat[j][i] += rval;
    }
}

// Inform the viewer that the array data has been changed.
printf("Updating data\n");
aglUpdate(fMat);

// Change the title to reflect the changes in the data set.
aglName(fMat, "sin(x) + cos(y) + noise");

// Wait for a key press
pause();

printf("Closing down the viewer\n");

// Close the Viewer
rc = aglClose(fMat);
if (rc != 0) {
    printf("aglClose failed\n");
    return -1;
}

pause();

// Deallocate memory
aglFree(fMat);

```

```

    return 0;
}

```

Example 3

The third example calls [aglStartWatch](#) to view an array that has been statically allocated. Calling [aglStartWatch](#) results in more memory being used than with [aglMalloc](#) or [aglAlloc](#), since the array data needs to be copied to the Array Viewer's process space.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
// avdef.h is the header file for the AView library
#include <avdef.h>

#define NCOLS 40
#define NROWS 50

void
pause()
{
    printf("press any key to continue\n");
    _getch();
}

int
main(int argc, char *argv[])
{
    // In this example our array will be statically allocated.
    float fmat[NROWS][NCOLS];
    int dims[] = {NROWS, NCOLS};
    int rc, i, j;
    float x, y, z, rval;
    const float fPi = 3.14159f;

    // Use aglStartWatch for arrays that aren't dynamically allocated
    // with aglMalloc(), or aglAlloc().
    aglStartWatch(fmat, dims[0]*dims[1]*sizeof(float));

    // Initialize the data
    for (j=0; j<dims[0]; j++) {
        y = (float)j/(float)dims[0];
        for (i=0; i<dims[1]; i++) {
            x = (float)i/(float)dims[1];
            z = (float)(sin(y*fPi) + cos(x*fPi));
            fmat[j][i] = z;
        }
    }

    // Call aglReshape to let the AView library know about the
    // array dimensions and type.
    rc = aglReshape((void *)&fmat[0][0], 2, dims, AGL_FLOAT);
    if (rc != 0) {
        printf("aglReshape failed!\n");
        return -1;
    }

    // Set the title bar on ArrayViewer
    aglName(fmat, "sin(x) + cos(y)");
}

```

```
// Inform the AView library to update its view of the data.
aglUpdate(fmat);

printf("Starting Array Viewer\n");
// aglShow will bring up Array Viewer with a view of our array.
rc = aglShow(fmat);
if (rc != 0) {
    printf("aglShow failed\n");
    return -1;
}
// Note in Array Viewer the Data\Refresh menu item is disabled since the
// viewer can't "pull" in data from statically allocated arrays.
pause();

// Add some psudeo-random fluctuations
for (j=0; j<dims[0]; j++) {
    y = (float)j/(float)dims[0];
    for (i=0; i<dims[1]; i++) {
        // produce a random number between 0 and 1.
        rval = (float)(rand() % 1000) * 0.001f;
        // Scale and shift to range: -0.1 to 0.1
        rval = rval * 0.2f - 0.1f;
        // Add to array element
        fmat[j][i] += rval;
    }
}

// Inform the viewer that the array data has been changed.
printf("Updating data\n");
aglUpdate(fmat);

// Change the title to reflect the changes in the data set.
aglName(fmat, "sin(x) + cos(y) + noise");

// Wait for a key press
pause();

printf("Closing down the viewer\n");

// Close the Viewer
rc = aglClose(fmat);
if (rc != 0) {
    printf("aglClose failed\n");
    return -1;
}

pause();

// Call aglEndWatch to free resources allocated by aglStartWatch
rc = aglEndWatch(fmat);

return 0;
}
```

API Calls for C++ Programmers: CAVIEWER Class Routines

C++ programmers can use the following API calls (CAVIEWER Class routines) to control the appearance of the Array Viewer window and the user-interaction of the Array Viewer in their applications. These routines are typically used in conjunction with one or more of the [agl routines](#).

Objects of the CAVIEWER class are instances of the Array Viewer application. The methods of this class can be used to perform most of the same operations that are available through the viewer user interface.

Use this class to allow your program to control the appearance of the Array Viewer window and perform the actions that would otherwise be required of the user of the Array Viewer (or performed by the Avis2D ActiveX control). Also, if you have several arrays that you wish to view sequentially; it is more efficient to create one instance of the Viewer using the CAVIEWER class than by using repeated aglShow and aglClose calls. Once you've created the Viewer object, you can load data arrays as desired, with the CAVIEWER::SetArray method.

Samples of these routines are provided online in folders in `... \ArrayVisualizer \Samples \C \`, such as AXIScale2d, Mandel, and Palette2D. Most Samples include a project workspace file, which allows you to open and build the project in the visual development environment.

Samples are installed on the hard disk when a Complete installation of Array Visualizer is performed. You can copy Samples folders from the Array Visualizer CD-ROM to your hard disk.

The following table alphabetically lists the CAVIEWER Class routines:

CAVIEWER	GetRowCol	SetGraphStyle
GetAnnotation	GetRowColDim	SetGraphType
GetArray	GetShading	SetGridDensity
GetArrayName	GetShowAxis	SetHiddenLine
GetAxisAutoDetail	GetShowDimLabels	SetHighLight
GetAxisAutoScale	GetShowGrid	SetHomePosition
GetAxisLabel	GetShowMarker	SetImageFilter
GetAxisStyle	GetShowPalette	SetImageOrientation
GetCameraCoi	GetTextureMode	SetLineSmooth
GetCameraPosition	GetUseAxisLabel	SetModifiedFlag
GetCellEditEnabled	GetUseColorPalette	SetNumMajorTickmarks

GetCompIndex	GetUseDefaultFormat	SetNumMinorTickmarks
GetCustomPalette	GetUseExp	SetPaletteAutoAdjust
GetDataClamp	GetUseHex	SetPaletteId
GetDataRefreshEnable	GetXClamp	SetPaletteRange
GetDataSelectEnable	GetYClamp	SetPrecision
GetDefaultFormat	GetZClamp	SetRoi
GetDepthcue	GetZScale	SetRoi2D
GetDimName	IsVisible	SetRowCol
GetDimScale	SetAnnotation	SetRowColDim
GetErrorNo	SetArray	SetShading
GetFieldWidth	SetArrayName	SetShowAxis
GetFileName	SetAxisAutoDetail	SetShowDimLabels
GetFixedAspect	SetAxisAutoScale	SetShowGrid
GetFontAutoScale	SetAxisLabel	SetShowMarker
GetGraphStyle	SetAxisStyle	SetShowPalette
GetGraphType	SetCameraCoi	SetTextureMode
GetGridDensity	SetCameraPosition	SetUseAxisLabel
GetHiddenLine	SetCellEditEnabled	SetUseColorPalette
GetHighLight	SetCompIndex	SetUseDefaultFormat
GetImageFilter	SetCustomPalette	SetUseExp
GetImageOrientation	SetDataClamp	SetUseHex
GetLineSmooth	SetDataRefreshEnable	SetXClamp
GetModifiedFlag	SetDataSelectEnable	SetYClamp
GetNumMajorTickmarks	SetDepthcue	SetZClamp
GetNumMinorTickmarks	SetDimName	SetZScale
GetPaletteId	SetDimScale	ShowWindow

GetPaletteRange	SetFieldWidth	ToHomePosition
GetPrecision	SetFileName	Update
GetRoiLb	SetFixedAspect	
GetRoiUb	SetFontAutoScale	

Samples are provided online in folders in ... \ArrayVisualizer\Samples\C\. For a description of the Array Visualizer Samples, view the file ... \ArrayVisualizer\Samples\Samples.htm in a Web browser. Most Samples include a project workspace file, which allows you to open and build the project in the visual development environment.

Samples are installed on the hard disk when a Complete installation is performed. You can copy Samples folders from the Array Visualizer CD-ROM to your hard disk (remove the read-only file property).

CAViewer::CAViewer

CAViewer class

Creates an Array Viewer instance.

```
CAViewer();
```

Parameters

None.

Return Value

None.

Remarks

This function constructs an Array Viewer application object. The Viewer window is initially hidden. Use **CAViewer::ShowWindow** to make it visible.

See Also

[CAViewer::ShowWindow](#)

CAViewer::GetAnnotation

CAViewer class

Gets the current annotation string.

```
const char *GetAnnotation( );
```

Parameters

None.

Return Value

Returns a pointer to the annotation string, or NULL if no annotation has been defined.

See Also

[CAViewer::SetAnnotation](#)

CAViewer::GetArray

CAViewer class

Gets a pointer to the currently loaded array.

```
voidp GetArray( );
```

Parameters

None.

Return Value

Returns a pointer to the array data currently loaded by this instance of the Viewer. If no array is loaded, NULL is returned.

See Also

[CAViewer::SetArray](#)

CAViewer::GetArrayName

CAViewer class

Gets the character string currently being displayed on the Array Viewer title bar.

```
const char *GetArrayName( );
```

Parameters

None.

Return Value

Returns a pointer to the string.

See Also

[CAViewer::SetArrayName](#)

CAViewer::GetAxisAutoDetail

CAViewer class

Gets the current AxisAutoDetail state.

```
BOOL GetAxisAutoDetail();
```

Parameters

None.

Return Value

Returns TRUE if AxisAutoDetail is enabled; otherwise, FALSE.

See Also

[CAViewer::SetAxisAutoDetail](#)

CAViewer::GetAxisAutoScale

CAViewer class

Gets the AxisAutoScale setting.

```
BOOL GetAxisAutoScale();
```

Parameters

None.

Return Value

Returns TRUE if AxisAutoScale is enabled; otherwise, FALSE.

See Also

[CAViewer::SetAxisAutoScale](#), [CAViewer::SetXClamp](#), [CAViewer::SetYClamp](#), [CAViewer::SetZClamp](#)

CAViewer::GetAxisLabel

CAViewer class

Sets the label to be displayed along the Graph View's axis.

```
const char *GetAxisLabel(enum Axis axis );
```

Parameters

axis

The desired axis.

Return Value

The label associated with the given axis.

See Also

[CAViewer::SetAxisLabel](#)

CAViewer::GetAxisStyle

CAViewer class

Sets the current axis style.

```
short GetAxisStyle( );
```

Parameters

None.

Return Value

The current axis style.

See Also

[CAViewer::SetAxisStyle](#)

CAViewer::GetCameraCoi

CAViewer class

Gets the current camera Center of Interest (COI).

```
BOOL GetCameraCoi(  
    float *xpos,  
    float *ypos,  
    float *zpos  
);
```

Parameters

xpos

A pointer to float that receives the X component of the camera COI.

ypos

A pointer to float that receives the Y component of the camera COI.

zpos

A pointer to float that receives the Z component of the camera COI.

Return Value

Returns TRUE if the operation is successful; otherwise, FALSE.

See Also

[CAViewer::SetCameraCoi](#)

CAViewer::GetCameraPosition

CAViewer class

Gets the current camera position.

```
BOOL GetCameraPosition(  
    float *xpos,  
    float *ypos,  
    float *zpos  
);
```

Parameters

xpos

A pointer to float that receives the X component of the camera position.

ypos

A pointer to float that receives the Y component of the camera position.

zpos

A pointer to float that receives the Z component of the camera position.

Return Value

Returns TRUE if the operation is successful; otherwise, FALSE.

See Also

[CAViewer::SetCameraPosition](#)

CAViewer::GetCellEditEnabled

CAViewer class

Gets the current Cell Edit state.

```
BOOL GetCellEditEnabled( );
```

Parameters

None.

Return Value

Returns TRUE if Cell Editing is enabled; otherwise, FALSE.

See Also

[CAViewer::SetCellEditEnabled](#)

CAViewer::GetCompIndex

CAViewer class

Gets the current component index values.

```
BOOL GetCompIndex(  
    long *XComp,  
    long *YComp,  
    long *ZComp,  
    long *WComp  
);
```

Parameters

XComp

A pointer to long that receives the X component index value.

YComp

A pointer to long that receives the Y component index value.

ZComp

A pointer to long that receives the Z component index value.

WComp

A pointer to long that receives the W component index value.

Return Value

Returns TRUE if the operation is successful; otherwise, FALSE.

See Also

[CAViewer::SetCompIndex](#)

CAViewer::GetCustomPalette

CAViewer class

Returns a pointer to a custom color palette.

```
void *GetCustomPalette();
```

Parameters

None.

Return Value

Returns a pointer to the custom palette data. Returns NULL if no custom palette has been defined.

See Also

[CAViewer::SetCustomPalette](#)

CAViewer::GetDataClamp

CAViewer class

Gets the data clamp setting.

```
BOOL GetDataClamp();
```

Parameters

None.

Return Value

Returns TRUE if data clamp is enabled; otherwise, FALSE.

Remarks

See Remarks under **CAViewer::SetDataClamp**.

See Also

[CAViewer::SetDataClamp](#)

CAViewer::GetDataRefreshEnable

CAViewer class

Gets the setting of the Data...Refresh menu item.

```
BOOL GetDataRefreshEnable();
```

Parameters

None.

Return Value

Returns TRUE if Data...Refresh is enabled; otherwise, FALSE.

See Also

[CAViewer::SetDataRefreshEnable](#)

CAViewer::GetDataSelectEnable

CAViewer class

Gets the data selection setting.

```
BOOL GetDataSelectEnable();
```

Parameters

None.

Return Value

Returns TRUE if Data Select is enabled; otherwise, FALSE.

See Also

[CAViewer::SetDataSelectEnable](#)

CAViewer::GetDefaultFormat

CAViewer class

Gets the default field width and precision used for displaying data in the Data View window.

```
BOOL GetDefaultFormat(  
    long *fieldwidth  
    long *precision  
);
```

Parameters

fieldwidth

A pointer to long that receives the field width of the default format.

precision

A pointer to long that receives the precision of the default format.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::SetUseDefaultFormat](#)

CAViewer::GetDepthcue

CAViewer class

Gets the depth cueing setting.

```
BOOL GetDepthcue( );
```

Parameters

None.

Return Value

Returns TRUE if depth cueing is enabled; otherwise, FALSE.

See Also

[CAViewer::SetDepthcue](#)

CAViewer::GetDimName

CAViewer class

Gets the name that's associated with the given dimension.

```
const char *GetDimName(  
    short dim  
);
```

Parameters

dim

The array dimension. The dimension must be in the range 0 to rank-1, where *rank* is the rank of the currently loaded array.

Return Value

The label associated with the given dimension.

See Also

[CAViewer::SetDimName](#)

CAViewer::GetDimScale

CAViewer class

Gets a pointer to the dimension scale (if any) associated with the specified dimension.

```
void *GetDimScale(  
    short dim  
);
```

Parameters

dim

The array dimension (in the range 0 to *rank*-1, where *rank* is the rank of the currently loaded array).

Return Value

Returns a pointer to the dimension scale data.

See Also

[CAViewer::SetDimScale](#)

CAViewer::GetErrorNo

CAViewer class

Gets an error number.

```
long GetErrorNo( );
```

Parameters

None.

Return Value

Returns the error number.

Remarks

When a method returns failure, **GetErrorNo** can be used to get an error code. Calling this method clears the error code.

CAViewer::GetFieldWidth

CAViewer class

Gets the current field width in the Data View window.

```
long GetFieldWidth( );
```

Parameters

None.

Return Value

The current FieldWidth.

See Also

[CAViewer::SetFieldWidth](#)

CAViewer::GetFileName

CAViewer class

Gets the path to the currently loaded file.

```
const char *GetFileName( );
```

Parameters

None.

Return Value

Returns a pointer to a string specifying the path of the file currently loaded by this Viewer instance, or NULL if no array is loaded.

See Also

[CAViewer::SetFileName](#)

CAViewer::GetFixedAspect

CAViewer class

Gets the current fixed aspect ratio setting.

```
BOOL GetFixedAspect( );
```

Parameters

None.

Return Value

Returns TRUE if Fixed Aspect Ratio is enabled; otherwise, FALSE.

See Also

[CAViewer::SetFixedAspect](#)

CAViewer::GetFontAutoScale

CAViewer class

Gets the current FontAutoScale state.

```
BOOL GetFontAutoScale( );
```

Parameters

None.

Return Value

Returns TRUE if FontAutoScale is enabled; otherwise, FALSE.

See Also

[CAViewer::SetFontAutoScale](#)

CAViewer::GetGraphStyle

CAViewer class

Gets the current graph style.

```
enum CAViewer::GraphStyle GetGraphStyle( );
```

Parameters

None.

Return Value

Returns the current graph style.

See Also

[CAViewer::SetGraphStyle](#)

CAViewer::GetGraphType

CAViewer class

Gets the current graph type.

```
enum CAVIEWER::GraphType GetGraphType();
```

Parameters

None.

Return Value

Returns the current graph type.

See Also

[CAVIEWER::SetGraphType](#)

CAVIEWER::GetGridDensity

CAVIEWER class

Gets the grid density value.

```
float GetGridDensity();
```

Parameters

None.

Return Value

Returns a floating-point value that represents the approximate number of pixels between adjacent grid lines.

See Also

[CAVIEWER::SetGridDensity](#)

CAVIEWER::GetHiddenLine

CAVIEWER class

Gets the current hidden line state.

```
BOOL GetHiddenLine();
```

Parameters

None.

Return Value

Returns TRUE if Hidden Line removal is enabled; otherwise, FALSE.

See Also

[CAViewer::SetHiddenLine](#)

CAViewer::GetHighLight

CAViewer class

Gets the current highlighting state.

```
BOOL GetHighLight();
```

Parameters

None.

Return Value

Returns TRUE if HighLighting is enabled; otherwise, FALSE.

See Also

[CAViewer::SetHighLight](#)

CAViewer::GetImageFilter

CAViewer class

Gets the current image linear filter settings.

```
BOOL GetImageFilter();
```

Parameters

None.

Return Value

Returns TRUE if Image Linear Filtering is enabled; otherwise, FALSE.

See Also

[CAViewer::SetImageFilter](#)

CAViewer::GetImageOrientation

CAViewer class

Gets the current image orientation setting.

```
enum CAViewer::ImageOrientation GetImageOrientation();
```

Parameters

None.

Return Value

Returns the current image orientation.

See Also

[CAViewer::SetImageOrientation](#)

CAViewer::GetLineSmooth

CAViewer class

Gets the line smoothing setting.

```
BOOL GetLineSmooth();
```

Parameters

None.

Return Value

Returns TRUE if line smoothing is enabled; otherwise, FALSE.

See Also

[CAViewer::SetLineSmooth](#)

CAViewer::GetModifiedFlag

CAViewer class

Gets the modified flag.

```
BOOL GetModifiedFlag( );
```

Parameters

None.

Return Value

Returns TRUE if the Array Viewer document is "dirty"; otherwise, FALSE.

Remarks

See Remarks under **CAViewer::SetModifiedFlag**.

See Also

[CAViewer::SetModifiedFlag](#)

CAViewer::GetNumMajorTickmarks

CAViewer class

Gets the number of large tickmarks for the desired axis.

```
short GetNumMajorTickmarks(  
    enum Axis axis  
    );
```

Parameters

axis

The desired axis.

Return Value

The number of large tickmarks.

See Also

[CAViewer::SetNumMajorTickmarks](#)

CAViewer::GetNumMinorTickmarks

CAViewer class

Gets the number of large tickmarks for the desired axis.

```
short GetNumMinorTickmarks(  
    enum Axis axis  
);
```

Parameters

axis

The desired axis.

Return Value

The number of small tickmarks that are displayed between each set of large tickmarks.

See Also

[CAViewer::SetNumMinorTickmarks](#), [CAViewer::SetNumMajorTickmarks](#)

CAViewer::GetPaletteId

CAViewer class

Gets the color palette identifier.

```
int GetPaletteId( );
```

Parameters

None.

Return Value

Returns the color palette identifier.

See Also

[CAViewer::SetPaletteId](#), [CAViewer::SetCustomPalette](#)

CAViewer::GetPaletteRange

CAViewer class

Gets the range of data values that will be associated with the color palette.

```
BOOL GetPaletteRange(
```



```
    double *minval,  
    double *maxval  
);
```

Parameters

minval

A pointer to double that receives the lower bound palette range.

maxval

A pointer to double that receives the upper bound palette range.

Return Value

None.

See Also

[CAViewer::SetPaletteRange](#)

CAViewer::GetPrecision

CAViewer class

Gets the current Data View precision value.

```
    long GetPrecision();
```

Parameters

None.

Return Value

Returns the current precision setting.

See Also

[CAViewer::SetPrecision](#)

CAViewer::GetRoiLb

CAViewer class

Gets the position number of the first element in the Region of Interest (ROI) of the specified dimension.

```
    long GetRoiLb(
```

```
    short dim
);
```

Parameters

dim

The array dimension (in the range 0 to *rank*-1, where *rank* is the rank of the currently loaded array).

Return Value

Returns the position index.

See Also

[CAViewer::GetRoiUb](#), [CAViewer::SetRoi](#)

CAViewer::GetRoiUb

CAViewer class

Gets the position number of the last element in the Region of Interest (ROI) of the specified dimension.

```
    GetRoiUb(
        short dim
    );
```

Parameters

dim

The array dimension (in the range 0 to *rank*-1, where *rank* is the rank of the currently loaded array).

Return Value

Returns the position index.

See Also

[CAViewer::GetRoiLb](#), [CAViewer::SetRoi](#)

CAViewer::GetRowCol

CAViewer class

Gets the current row and column position

```
GetRowCol(  
    long *row,  
    long *col  
);
```

Parameters

row

A pointer to long that receives the current row.

col

A pointer to long that receives the current column.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::SetRowCol](#)

CAViewer::GetRowColDim

CAViewer class

Gets the row and column dimensions.

```
GetRowColDim(  
    short *rowdim,  
    short *coldim  
);
```

Parameters

rowdim

A pointer to a short that receives the row dimension of the current array.

coldim

A pointer to a short that receives the row dimension of the current array.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

See [CAViewer::SetRowColDim](#) for an explanation of how row and column dimensions are used in

Array Viewer.

See Also

[CAViewer::SetRowColDim](#)

CAViewer::GetShading

CAViewer class

Gets the current shading state.

```
BOOL GetShading( );
```

Parameters

None.

Return Value

Returns TRUE if shading is enabled; otherwise, FALSE.

See Also

[CAViewer::SetShading](#)

CAViewer::GetShowAxis

CAViewer class

Gets the show axis setting.

```
BOOL GetShowAxis( );
```

Parameters

None.

Return Value

Returns TRUE if the graph axis is displayed; otherwise, FALSE.

See Also

[CAViewer::SetShowAxis](#)

CAViewer::GetShowDimLabels

CAViewer class

Gets the current ShowDimLabels state.

```
BOOL GetShowDimLabels();
```

Parameters

None.

Return Value

Returns TRUE if ShowDimLabels is enabled; otherwise, FALSE.

See Also

[CAViewer::SetShowDimLabels](#)

CAViewer::GetShowGrid**CAViewer class**

Gets the show grid setting.

```
BOOL GetShowGrid();
```

Parameters

None.

Return Value

Returns TRUE if grid lines are displayed; otherwise, FALSE.

See Also

[CAViewer::SetShwowGrid](#)

CAViewer::GetShowMarker**CAViewer class**

Gets the current show marker state.

```
BOOL GetShowMarker();
```

Parameters

None.

Return Value

Returns TRUE if show marker is enabled; otherwise, FALSE.

See Also

[CAViewer::SetShowMarker](#)

CAViewer::GetShowPalette

CAViewer class

Gets the current setting of show color palette.

```
BOOL GetShowPalette();
```

Parameters

None.

Return Value

Returns TRUE if ShowColorPalette is enabled; otherwise, FALSE.

See Also

[CAViewer::SetShowPalette](#)

CAViewer::GetTextureMode

CAViewer class

Gets the current texture mode state.

```
short GetTextureMode( );
```

Parameters

None.

Return Value

Returns 1 if texture mode is enabled; otherwise, 0.

See Also

[CAViewer::SetTextureMode](#)

CAViewer::GetUseAxisLabel

CAViewer class

Gets the current use axis label state for the given axis.

```
BOOL GetUseAxisLabel(enum Axis axis );
```

Parameters

axis

The desired axis.

Return Value

Returns TRUE if the display of axis labels is enabled for the given axis; otherwise, FALSE.

See Also

[CAViewer::SetUseAxisLabel](#)

CAViewer::GetUseColorPalette

CAViewer class

Gets the current setting of use color palette.

```
BOOL GetUseColorPalette();
```

Parameters

None.

Return Value

Returns TRUE if UseColorPalette is enabled; otherwise, FALSE.

See Also

[CAViewer::SetUseColorPalette](#)

CAViewer::GetUseDefaultFormat

CAViewer class

Gets the current UseDefaultFormat state.

```
    BOOL GetUseDefaultFormat();
```

Parameters

None.

Return Value

Returns TRUE if UseDefaultFormat is enabled; otherwise, FALSE.

See Also

[CAViewer::SetUseDefaultFormat](#)

CAViewer::GetUseExp

CAViewer class

Gets the current UseExp state.

```
    BOOL GetUseExp();
```

Parameters

None.

Return Value

Returns TRUE if UseExp is enabled; otherwise, FALSE.

See Also

[CAViewer::SetUseExp](#)

CAViewer::GetUseHex

CAViewer class

Gets the current UseHex state.


```
BOOL GetUseHex( );
```

Parameters

None.

Return Value

Returns TRUE if UseHex is enabled; otherwise, FALSE.

See Also

[CAViewer::SetUseHex](#)

CAViewer::GetXClamp

CAViewer class

Gets the upper and lower bounds for X-coordinate values in the Graph View.

```
void GetXClamp(  
    *double minval,  
    *double maxval  
);
```

Parameters

minval

A pointer to double that receives the lower bound for the X axis.

maxval

A pointer to double that receives the upper bound for the X axis.

Return Value

None.

Remarks

See Remarks under **CAViewer::SetXClamp**.

See Also

[CAViewer::SetXClamp](#)

CAViewer::GetYClamp

CAViewer class

Gets the upper and lower bounds for Y-coordinate values in the Graph View.

```
void GetYClamp(  
    *double minval,  
    *double maxval  
);
```

Parameters

minval

A pointer to double that receives the lower bound for the Y axis.

maxval

A pointer to double that receives the upper bound for the Y axis.

Return Value

None.

Remarks

See Remarks under **CAViewer::SetYClamp**.

See Also

[CAViewer::SetYClamp](#)

CAViewer::GetZClamp

CAViewer class

Gets the upper and lower bounds for Z-coordinate values in the Graph View.

```
void GetZClamp(  
    *double minval,  
    *double maxval  
);
```

Parameters

minval

A pointer to double that receives the lower bound for the Z axis.

maxval

A pointer to double that receives the upper bound for the Z axis.

Return Value

None.

Remarks

See Remarks under **CAViewer::SetZClamp**.

See Also

[CAViewer::SetZClamp](#)

CAViewer::GetZScale

CAViewer class

Gets the Z scale value.

```
float GetZScale();
```

Parameters

None.

Return Value

Returns the current ZScale value.

See Also

[CAViewer::SetZScale](#)

CAViewer::IsVisible

CAViewer class

Returns the Array Viewer visibility state.

```
BOOL IsVisible();
```

Parameters

None.

Return Value

Returns TRUE if the Array Viewer window is currently being displayed; FALSE if it is hidden.

Remarks

If the user closes the Array Viewer window while the program that invoked it is still active, the Array Viewer is still running, but is not visible on the desktop. This routine can be used to determine if the user has closed the Array Viewer.

See Also

[CAViewer::ShowWindow](#)

CAViewer::SetAnnotation

CAViewer class

Sets the annotation string.

```
BOOL SetAnnotation(  
    const char *string  
);
```

Parameters

string

The string to be displayed in the Array Viewer's annotation dialog.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::GetAnnotation](#)

CAViewer::SetArray

CAViewer class

Lets you view a specified array.

```
BOOL SetArray(  
    void array  
);
```

Parameters

array

A pointer to an array allocated by a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**; or added to the watch list by a call to **aglStartWatch**.

Return Value

Returns TRUE if the array was successfully loaded; otherwise, FALSE.

See Also

[aglAlloc](#), [aglMalloc](#), [aglMallocEx](#), [aglStartWatch](#), [CAViewer::GetArray](#)

CAViewer::SetArrayName

CAViewer class

Sets the character string to be displayed on the Array Viewer title bar.

```
BOOL SetArrayName(  
    const char *title  
);
```

Parameters

title

The character string to be displayed on the title bar.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::GetArrayName](#)

CAViewer::SetAxisAutoDetail

CAViewer class

Sets a value that determines whether the number of both large and small tick marks on the axes is to be handled automatically by Array Visualizer, or explicitly by using other properties.

```
BOOL SetAxisAutoDetail(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, AxisAutoDetail is enabled; if set to FALSE, it is disabled.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

When AxisAutoDetail is set to FALSE, **SetNumMajorTickmarks** and **SetNumMinorTickmarks** can be used to explicitly control the number of tickmarks.

See Also

[CAViewer::GetAxisAutoDetail](#), [CAViewer::SetNumMajorTickmarks](#),
[CAViewer::SetNumMinorTickmarks](#)

CAViewer::SetAxisAutoScale

CAViewer class

Determines whether axis auto scaling is automatically adjusted to the data range.

```
void SetAxisAutoScale(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, the axis scales will be adjusted to the minimum and maximum data values within the Region of Interest (ROI). If set to FALSE, the XMinClamp, XMaxClamp, YMinClamp, YMaxClamp, ZMinClamp, and ZMaxClamp values will be used.

Return Value

None.

See Also

[CAViewer::GetAxisAutoScale](#), [CAViewer::SetXClamp](#), [CAViewer::SetYClamp](#),
[CAViewer::SetZClamp](#)

CAViewer::SetAxisLabel

CAViewer class

Sets the label to be displayed along the Graph View's axis.

```
BOOL SetAxisLabel(  
    enum Axis axis  
    const char *label  
);
```

Parameters

axis

The desired axis.

label

The string to be displayed along the axis.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

The string used in **SetAxisLabel** will not be displayed if the given axis is mapped to either the row or column dimensions, and **SetUseAxisLabel** has been disabled.

See Also

[CAViewer::SetRowColDim](#), [CAViewer::SetUseAxisLabel](#), [CAViewer::GetAxisLabel](#)

CAViewer::SetAxisStyle

CAViewer class

Sets the current axis style.

```
BOOL SetAxisStyle(  
    short nstyle  
);
```

Parameters

nstyle

A value that can be used to select among several different axis styles.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

See the **AxisStyle** Avis2D property for a description of the different axis styles.

See Also

[AxisStyle property](#), [CAViewer::GetAxisStyle](#)

CAViewer::SetCameraCoi

CAViewer class

Sets the camera center of interest (COI).

```
BOOL SetCameraCoi(  
    float xpos,  
    float ypos,  
    float zpos  
);
```

Parameters

xpos
The X position.

ypos
The Y position.

zpos
The Z position.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

Defines the point that the camera is looking to.

See Also

[CAViewer::GetCameraCoi](#), [CAViewer::SetCameraPosition](#)

CAViewer::SetCameraPosition

CAViewer class

Sets the camera position.

```
BOOL SetCameraPosition(  
    float xpos,  
    float ypos,  
    float zpos  
);
```

Parameters

xpos
The X position.

ypos
The Y position.

zpos
The Z position.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

This function can be used to adjust the virtual camera position for 3D graphs (that is, Height Plot and Vector Graphs with 3 or 4 components). The graph is defined as existing within a volume defined by (0, 0, 0) and (1, 1, 1).

For example, if the camera position is set to (0.5, 0.5, 2.0), and the center of interest (the position which the camera is pointed toward is (0.5, 0.5, 0.0), the viewpoint will appear to be directly over the graph.

See Also

[CAViewer::GetCameraPosition](#), [CAViewer::SetCameraCoi](#)

CAViewer::SetCellEditEnabled

CAViewer class

Enables or disables Cell Editing in the Data View window.

```
BOOL SetCellEditEnabled(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, Cell Editing is enabled; if set to FALSE, Cell Editing is disabled.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

When Cell Editing is enabled, double-clicking on a cell turns the cell into edit mode, so you can edit the cell. When disabled, double-clicking does not turn the cell into edit mode.

See Also

[CAViewer::GetCellEditEnabled](#)

CAViewer::SetCompIndex

CAViewer class

Sets the component index values.

```
BOOL SetCompIndex(  
    long XComp,  
    long YComp,  
    long ZComp = -1,  
    long WComp = -1  
);
```

Parameters

XComp

The X component index.

YComp

The Y component index.

ZComp

The Z component index.

WComp

The W component index.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

Each index value must be an integer between -1 and $n-1$, where n is the number of columns (that is, the extent of the c dimension of the array, where c is the current column dimension). If a value of -1 is used, that component is not included in the graph. Otherwise, the vector component is derived from the given array index.

In Vector Graph mode, data values are graphed as a sequence of $\langle x, y, z, w \rangle$ vectors, where each vector component is extracted from the array based on the component index values.

See Also

[CAViewer::GetCompIndex](#), [CAViewer::GetRowColDim](#)

CAViewer::SetCustomPalette

CAViewer class

Creates a custom palette.

```
    BOOL SetCustomPalette(  
        void *paletteData  
    );
```

Parameters

paletteData

A pointer to a 1024-byte data array that defines the custom palette.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

The palette won't be displayed in the Graph View until **CAViewer::SetPaletteId**(USER_DEFINED) is called.

See Also

[CAViewer::SetPaletteId](#), [CAViewer::GetCustomPalette](#)

CAViewer::SetDataClamp

CAViewer class

Enables or disables data clamping.

```
void SetDataClamp(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, data clamping is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

Data clamping determines whether the Graph View clamps data values that are outside the clamp range.

If data clamping is enabled, data values will be "clamped" to the clamp range. If data clamping is disabled and array values within the Region of Interest (ROI) extend beyond the clamp range, the Graph View will "clip" out data points outside the clamp range.

See Also

[CAViewer::SetXClamp](#), [CAViewer::SetYClamp](#), [CAViewer::SetZClamp](#),
[CAViewer::SetAxisAutoScale](#), [CAViewer::GetDataClamp](#)

CAViewer::SetDataRefreshEnable

CAViewer class

Enables or disables the Data...Refresh menu item.

```
void SetDataRefreshEnable(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, the Data...Refresh menu item is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

You can use the Data...Refresh menu item to update your display to reflect the current array data.

See Also

[CAViewer::GetDataRefreshEnable](#)

CAViewer::SetDataSelectEnable

CAViewer class

Enables or Disables data selection.

```
SetDataSelectEnable(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, data selection is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

Turning data selection off disables the tooltip windows.

See Also

[CAViewer::GetDataSelectEnable](#)

CAViewer::SetDepthcue

CAViewer class

Enables or disables depth cueing.

```
void SetDepthcue(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, depth cueing is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

When enabled, depth cueing fades out more distant parts of the graph to provide an enhanced perception of depth. Depthcue has no effect unless the GraphType property is HeightField.

See Also

[CAViewer::GetDepthcue](#), [CAViewer::GetGraphType](#)

CAViewer::SetDimName

CAViewer class

Associates a name with the given dimension.

```
BOOL SetDimName(  
    short dim  
    const char *name  
);
```

Parameters

dim

The array dimension that the following argument will be applied to. The dimension must be in the range 0 to rank-1, where *rank* is the rank of the currently loaded array.

name

The text string to be associated with the given dimension.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

The Graph View displays the dimension name as an axis label when the given dimension is assigned as either a row or column dimension, and SetUseAxisLabel has been set to FALSE for the indicated axis.

The Data View will display the dimension names as long as SetShowDimLabels has been enabled.

See Also

[CAViewer::SetShowDimLabels](#), [CAViewer::SetRowColDim](#), [CAViewer::SetUseAxisLabel](#),
[CAViewer::GetDimName](#)

CAViewer::SetDimScale

CAViewer class

Associates an axis dimension scale with a dimension.

```
BOOL SetDimScale(  
    short dim,  
    void *scaleData  
);
```

Parameters

dim

The array dimension (in the range 0 to *rank*-1, where *rank* is the rank of the currently loaded array) that the axis scale will be applied to.

scaleData

A pointer to a one-dimensional array that defines the axis scale values.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

The array pointed to by *scaleData* must have previously been allocated by a call to **aglAlloc**, **aglMalloc**, or **aglMallocEx**; or added to the watch list by a call to **aglStartWatch**.

See Also

[aglAlloc](#), [aglMalloc](#), [aglMallocEx](#), [aglStartWatch](#), [CAViewer::GetDimScale](#)

CAViewer::SetFieldWidth

CAViewer class

Sets a value that determines the width allocated to display each data value in the Data View window.

```
BOOL SetFieldWidth(  
    long width  
);
```

Parameters

width

The field width to be used.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

SetFieldWidth is ignored if the UseDefaultFormat property is TRUE. It is also ignored if UseHex is TRUE when displaying integer data, or if UseExp is TRUE when displaying floating-point data.

See Also

[CAViewer::GetUseDefaultFormat](#), [CAViewer::GetUseExp](#), [CAViewer::GetUseHex](#), [CAViewer::GetFieldWidth](#)

CAViewer::SetFileName

CAViewer class

Loads a specified file.

```
BOOL SetFileName(  
    const char *filename  
);
```

Parameters

filename

The path to the file to be loaded by this Viewer instance.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

None.

See Also

[CAViewer::GetFileName](#)

CAViewer::SetFixedAspect

CAViewer class

Enables or disables the fixed aspect ratio setting.

```
void SetFixedAspect(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, Fixed Aspect Ratio is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

For Image Map graphs, when Fixed Aspect Ratio is enabled, the relative lengths of the X and Y axis don't change as the graph is resized. If it is disabled, the X and Y axis will be independently scaled to fit the view area.

See Also

[CAViewer::GetFixedAspect](#)

CAViewer::SetFontAutoScale

CAViewer class

Sets a value that determines whether the font size used in axis labels changes as the size of the graph view window changes, or is held fixed.

```
BOOL SetFontAutoScale(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, FontAutoScale is enabled; if set to FALSE, it is disabled.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::GetFontAutoScale](#)

CAViewer::SetGraphStyle

CAViewer class

Sets the graph style (mesh, surface, barchart, lines, or points).

```
BOOL SetGraphStyle(  
    enum GraphStyle graphStyle  
    );
```

Parameters

graphStyle
The desired graph style.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

Some graph styles are not valid for particular graph types. For a description of the different graph types, see Array Viewer online help.

See Also

[CAViewer::GetGraphStyle](#)

CAViewer::SetGraphType

CAViewer class

Sets the graph type (Height Plot, Image Map, Vector, or Plane View).

```
BOOL SetGraphType(  
    enum GraphType graphType  
    );
```

Parameters

graphType
The desired graph type.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

Some graph types are not valid for particular arrays. For a description of the different graph types, see Array Viewer online help.

See Also

[CAViewer::GetGraphType](#)

CAViewer::SetGridDensity

CAViewer class

Sets a value that determines how finely the grid is drawn on the graph surface.

```
BOOL SetGridDensity(  
    float newVal  
);
```

Parameters

newVal

A floating-point value that represents the desired number of pixels between adjacent grid lines.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

This function has no effect unless the ShowGrid value is TRUE and the GraphStyle is Surface.

See Also

[CAViewer::GetGridDensity](#), [CAViewer::GetGraphStyle](#)

CAViewer::SetHiddenLine

CAViewer class

Enables or disables hidden line removal.

```
void SetHiddenLine(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, HiddenLine removal is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

When hidden line removal is enabled (on), and the graph style is Mesh or Line, lines are hidden if they lie behind another part of the graph. Otherwise, the entire line segment is drawn.

See Also

[CAViewer::GetGraphStyle](#), [CAViewer::GetHiddenLine](#)

CAViewer::SetHighLight

CAViewer class

Enables or disables highlighting.

```
void SetHighLight(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, HighLighting is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

If highlighting is enabled, the graph surface will display highlights that change as the graph is rotated (that is, it will look as if the graph was made of a shiny material). If it is disabled, highlights won't be displayed.

See Also

[CAViewer::GetHighLight](#)

CAViewer::SetHomePosition

CAViewer class

Sets the current camera position as the home position.

```
void SetHomePosition( );
```

Parameters

None.

Return Value

None.

See Also

[CAViewer::GetCameraPosition](#), [CAViewer::ToHomePosition](#)

CAViewer::SetImageFilter

CAViewer class

Enables or disables image linear filtering.

```
void SetImageFilter(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, Image Linear Filtering is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

When enabled, Image Linear Filtering smoothes the color values of each data point in the graph by

blending the color with colors of neighboring points.

See Also

[CAViewer::GetImageFilter](#)

CAViewer::SetImageOrientation

CAViewer class

Sets the image orientation (IDENTITY, XFLIP, YFLIP, XYFLIP).

```
BOOL SetImageOrientation(  
    enum ImageOrientation newVal  
);
```

Parameters

newVal

The new Image Orientation settings.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

In the Image Map graph type, the default orientation (IDENTITY) of the image is for the (0,0) element to be placed in the lower left corner of the graph. Other orientations can be set by using values XFLIP, YFLIP, or XYFLIP.

See Also

[CAViewer::GetGraphType](#), [CAViewer::GetImageOrientation](#)

CAViewer::SetLineSmooth

CAViewer class

Enables or disables line smoothing.

```
void SetLineSmooth(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, line smoothing is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

When enabled, line smoothing helps reduce the "jaggedness" of lines that are drawn in the graph window.

See Also

[CAViewer::GetLineSmooth](#)

CAViewer::SetModifiedFlag

CAViewer class

Sets the modified flag.

```
BOOL SetModifiedFlag(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, the modified flag is set; if set to FALSE, the modified flag is cleared.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

When a change in the Array Viewer state is made, either by the CAVIEWER method or user interaction, the Viewer sets an internal flag noting that the state has been modified. If the flag is set, and the user closes the Viewer window, a dialog is displayed asking if the changes should be saved. This method can be used to modify the current value of the modified flag.

For example, if you do not want the user prompted to save changes, you can use the following call after making any CAVIEWER call that modifies the Array Viewer state:

```
CAViewer::SetModifiedFlag(FALSE)
```

See Also

[CAViewer::GetModifiedFlag](#)

CAViewer::SetNumMajorTickmarks

CAViewer class

Sets the number of large tickmarks displayed along the indicated axis.

```
BOOL SetNumMajorTickmarks(  
    enum Axis axis  
    short num  
);
```

Parameters

axis

The desired axis.

num

The number of large tickmarks to be displayed.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

This method will not effect the Graph View appearance until AxisAutoDetail is disabled.

See Also

[CAViewer::SetAxisAutoDetail](#), [CAViewer::GetNumMajorTickmarks](#),
[CAViewer::SetNumMinorTickmarks](#)

CAViewer::SetNumMinorTickmarks

CAViewer class

Sets the number of small tickmarks for the desired axis.

```
BOOL SetNumMinorTickmarks(  
    enum Axis axis  
    short num  
);
```

Parameters

axis

The desired axis.

num

The number of small tickmarks to be displayed between each set of large tickmarks.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

This method will not effect the Graph View appearance until AxisAutoDetail is disabled.

See Also

[CAViewer::SetAxisAutoDetail](#), [CAViewer::GetNumMinorTickmarks](#),
[CAViewer::SetNumMajorTickmarks](#)

CAViewer::SetPaletteAutoAdjust

CAViewer class

Determines whether the palette range is automatically adjusted to the data range.

```
void SetPaletteAutoAdjust(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, the palette range will be set to the minimum and maximum data values within the Region of Interest (ROI). If set to FALSE, the values defined by **CAViewer::SetPaletteRange** will be used.

Return Value

None.

See Also

[CAViewer::SetPaletteRange](#)

CAViewer::SetPaletteId

CAViewer class

Sets the color palette.

```
void SetPaletteId(  
    enum StandardPalettes paletteId  
);
```

Parameters

paletteId
The palette identifier.

Return Value

None.

See Also

[CAViewer::GetPaletteId](#), [CAViewer::SetCustomPalette](#)

CAViewer::SetPaletteRange

CAViewer class

Sets the range of data values that the color palette will be associated with.

```
void SetPaletteRange(  
    double minval,  
    double maxval  
);
```

Parameters

minval
The minimum value in the data range.

maxval
The maximum value in the data range.

Return Value

None.

Remarks

This subroutine only affects the graph's appearance when PaletteAutoAdjust is disabled.

See Also

[CAViewer::GetPaletteRange](#), [CAViewer::SetPaletteAutoAdjust](#)

CAViewer::SetPrecision

CAViewer class

Sets the precision for floating-point numbers displayed in the Data View.

```
BOOL SetPrecision(  
    long newVal  
);
```

Parameters

newVal

The number of digits to be displayed to the right of the decimal point.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::GetPrecision](#)

CAViewer::SetRoi

CAViewer class

Sets the position number of the first and last elements in the Region of Interest (ROI) of the specified array dimension.

```
BOOL SetRoi(  
    short dim,  
    long lbound,  
    long ubound  
);
```

Parameters

dim

The array dimension that the following two arguments will be applied to. The dimension must be in the range 0 to *rank*-1, where *rank* is the rank of the currently loaded array.

lbound

The lower array index to be included in the ROI. The array index must be in the range 0 to *extent*-1, where *extent* is the extent of dimension *dim*.

ubound

The upper array index to be included in the ROI. This array index must be greater than or equal to *lbound*, and in the range 0 to *extent-1*, where *extent* is the extent of dimension *dim*.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

This function can be used to select and view a subregion of the current array. You must call **CAViewer::Update** before the new ROI is mapped to the graph.

See Also

[CAViewer::Update](#)

CAViewer::SetRoi2D

CAViewer class

Sets the Region of Interest (ROI) within the current 2D array "slice".

```
BOOL SetRoi2D(  
    long col_start,  
    long row_start,  
    long num_cols,  
    long num_rows  
);
```

Parameters

col_start

The lower array index of the current column dimension to be included in the ROI.

row_start

The lower array index of the current row dimension to be included in the ROI.

num_cols

The number of columns to be included in the ROI.

num_rows

The number of rows to be included ROI.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

This subroutine is similar to the [SetRoi](#) subroutine, which also updates the current ROI. However, **SetRoi2D** is more convenient to use in cases where the current 2D slice of the array is not being changed.

CAViewer::SetRowCol

CAViewer class

Sets the row and column position.

```
BOOL SetRowCol(  
    long row,  
    long col  
);
```

Parameters

row

The new row index. The row should be in the range 0 to $n-1$, where n is the number of rows in the current array.

col

The new column index. The column should be in the range 0 to $n-1$, where n is the number of columns in the current array.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

Setting the row and column position will update both the Data and Graph Views.

See Also

[CAViewer::GetRowCol](#)

CAViewer::SetRowColDim

CAViewer class

Sets the row and column dimensions.

```
BOOL SetRowColDim(  
    short rowdim,
```

```
    short coldim
);
```

Parameters

rowdim

The array dimension to be viewed as rows.

coldim

The array dimension to viewed as columns.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

For arrays of rank 3 or higher, the Array Viewer displays the array data as a 2D sub-array of the higher dimension array. The *rowdim* and *coldim* values define which two dimensions of the array get mapped as rows and columns, respectively.

For arrays of rank 2, this function can be used to control which dimension gets mapped as rows. For arrays of rank 1, this function has no effect.

See Also

[CAViewer::GetRowColDim](#)

CAViewer::SetShading

CAViewer class

Enables or disables shading.

```
    void SetShading(
        BOOL onoroff
    );
```

Parameters

onoroff

If set to TRUE, shading is enabled; if set to FALSE, it is disabled.

Return Value

None.

Remarks

The shading state determines how color values are interpolated across the graph's surface. If shading is enabled, the colors are interpolated between neighboring data points. If shading is disabled, each data element of the graph is drawn in a solid color.

See Also

[CAViewer::GetShading](#)

CAViewer::SetShowAxis

CAViewer class

Displays or hides the graph axis.

```
void SetShowAxis(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, the graph axis will be displayed; if set to FALSE, the axis will be hidden.

Return Value

None.

See Also

[CAViewer::GetShowAxis](#)

CAViewer::SetShowDimLabels

CAViewer class

Sets a value that determines whether to display the column and row dimension labels in the Data View window.

```
BOOL SetShowDimLabels(  
    BOOL show  
);
```

Parameters

show

If set to TRUE, dimension labels are displayed; if set to FALSE, they are hidden.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::GetShowDimLabels](#)

CAViewer::SetShowGrid

CAViewer class

Displays or hides grid lines.

```
void SetShowGrid(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, grid lines will be displayed in the graph; if set to FALSE, they will not be displayed.

Return Value

None.

Remarks

This function applies only to graphs whose GraphStyle is Surface.

See Also

[CAViewer::GetShowGrid](#), [CAViewer::GetGraphStyle](#)

CAViewer::SetShowMarker

CAViewer class

Displays or hides the marker.

```
void SetShowMarker(  
    BOOL onoroff  
);
```


Parameters*onoroff*

If set to TRUE, the marker will be displayed; if set to FALSE, it will be hidden.

Return Value

None.

See Also

[CAViewer::GetShowMarker](#)

CAViewer::SetShowPalette

CAViewer class

Displays or hides the color palette.

```
void SetShowPalette(  
    BOOL onoroff  
);
```

Parameters*onoroff*

If set to TRUE, the current color palette will be displayed in the Graph View. If set to FALSE, the palette will not be displayed.

Return Value

None.

See Also

[CAViewer::GetShowPalette](#)

CAViewer::SetTextureMode

CAViewer class

Enables or disables 1D textured color mapping.

```
void SetTextureMode(  
    short mode  
);
```

Parameters

mode

The desired texture mode. If set to 1, texture mapping is enabled; if set to 0, it is disabled.

Return Value

None.

Remarks

Texture mapping results in a more precise mapping of palette color values to the graph surface, but may result in slower graph updates. The visual difference between different TextureMode values is most noticeable when the number of data points in the graph is small.

TextureMode has no effect if Shading is disabled.

See Also

[CAViewer::GetTextureMode](#), [CAViewer::GetShading](#)

CAViewer::SetUseAxisLabel

CAViewer class

Enables or disables display of Axis labels.

```
BOOL SetUseAxisLabel(  
    enum Axis axis  
    BOOL onoroff  
);
```

Parameters

axis

The desired axis.

onoroff

If set to TRUE, the axis label for the indicated dimension is displayed. If set to FALSE, the dimension name (if available) is displayed.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

If the given axis doesn't correspond to an array dimension (for example: the Z-Axis in a Height Plot graph), calling **SetUseAxisLabel** with a value of TRUE will have no effect.

See Also

[CAViewer::GetUseAxisLabel](#), [CAViewer::SetAxisLabel](#), [CAViewer::SetDimName](#)

CAViewer::SetUseColorPalette

CAViewer class

Enables or disables use of the color palette.

```
void SetUseColorPalette(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, the current color palette will be used to map data values to colors. If set to FALSE, the current graph color will be used.

Return Value

None.

See Also

[CAViewer::SetPaletteRange](#), [CAViewer::SetPaletteAutoAdjust](#), [CAViewer::SetPaletteId](#),
[CAViewer::GetUseColorPalette](#)

CAViewer::SetUseDefaultFormat

CAViewer class

Sets a value that determines whether to display data in the Data View window using the default format.

```
BOOL SetUseDefaultFormat(  
    BOOL use  
);
```

Parameters

use

If set to TRUE, data values will be displayed in the default format. If set to FALSE, the format

can be controlled explicitly by using the **SetUseHex**, **SetFieldWidth**, and **SetUseExp** methods.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::GetUseDefaultFormat](#), [CAViewer::GetDefaultFormat](#), [CAViewer::SetUseHex](#), [CAViewer::SetFieldWidth](#), [CAViewer::SetUseExp](#)

CAViewer::SetUseExp

CAViewer class

Sets a value that determines whether to display floating-point data in scientific notation or decimal format in the Data View window.

```
BOOL SetUseExp(  
    BOOL use  
);
```

Parameters

use

If set to TRUE, floating-point data is displayed in scientific notation; if set to FALSE, the data is displayed in decimal format.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

The value passed to **SetUseExp** is ignored when displaying integer data, or if **UseDefaultFormat** has been set to TRUE.

See Also

[CAViewer::GetUseDefaultFormat](#), [CAViewer::GetUseExp](#)

CAViewer::SetUseHex

CAViewer class

Sets a value that determines whether to display integer data in hexadecimal or decimal format in the Data View window.

```
BOOL SetUseHex(  
  BOOL use  
);
```

Parameters

use

If set to TRUE, integer data is displayed in hexadecimal format; if set to FALSE, the data is displayed in decimal format.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

The value passed to **SetUseHex** is ignored when displaying floating-point data, or if **UseDefaultFormat** has been set to TRUE.

See Also

[CAViewer::GetUseDefaultFormat](#), [CAViewer::GetUseHex](#)

CAViewer::SetXClamp

CAViewer class

Sets the upper and lower bounds for X-coordinate values in the Graph View.

```
void SetXClamp(  
  double minval,  
  double maxval  
);
```

Parameters

minval

The lower bound for the X axis.

maxval

The upper bound for the X axis.

Return Value

None.

Remarks

This function only affects the graph appearance when the graph type is Vector and AxisAutoScale is disabled.

See Also

[CAViewer::SetDataClamp](#), [CAViewer::SetGraphType](#), [CAViewer::SetAxisAutoScale](#), [CAViewer::GetXClamp](#)

CAViewer::SetYClamp

CAViewer class

Sets the upper and lower bounds for Y-coordinate values in the Graph View.

```
void SetYClamp(  
    double minval,  
    double maxval  
);
```

Parameters

minval

The lower bound for the Y axis.

maxval

The upper bound for the Y axis.

Return Value

None.

Remarks

This function only affects the graph appearance when the graph type is Vector and AxisAutoScale is disabled.

See Also

[CAViewer::SetDataClamp](#), [CAViewer::SetGraphType](#), [CAViewer::SetAxisAutoScale](#), [CAViewer::GetYClamp](#)

CAViewer::SetZClamp

CAViewer class

Sets the upper and lower bounds for Z-coordinate values in the Graph View.

```
void SetZClamp(  
    double minval,  
    double maxval  
);
```

Parameters

minval

The lower bound for the Z axis.

maxval

The upper bound for the Z axis.

Return Value

None.

Remarks

This function only affects the graph appearance when AxisAutoScale is disabled.

See Also

[CAViewer::SetDataClamp](#), [CAViewer::SetAxisAutoScale](#), [CAViewer::GetZClamp](#)

CAViewer::SetZScale

CAViewer class

Sets the height of the Z axis relative to the X and Y axes.

```
void SetZScale(  
    float scale  
);
```

Parameters

scale

The Z scaling factor.

Return Value

None.

Remarks

If *scale* is 1.0, the Z height will be equal to the X and Y axes. Other values will change the Z axis

proportionally.

See Also

[CAViewer::GetZScale](#)

CAViewer::ShowWindow

CAViewer class

Displays or hides the Array Viewer window.

```
BOOL ShowWindow(  
    BOOL onoroff  
);
```

Parameters

onoroff

If set to TRUE, the window will be displayed; if set to FALSE, it will be hidden.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

See Also

[CAViewer::IsVisible](#)

CAViewer::ToHomePosition

CAViewer class

Returns the camera position to the home position.

```
void ToHomePosition( );
```

Parameters

None.

Return Value

None.

See Also

[CAViewer::SetHomePosition](#)

CAViewer::Update

CAViewer class

Forces Array Viewer to redraw its view to reflect any data changes since the last Update.

```
BOOL Update(  
    long count  
);
```

Parameters

count

Can be 0 or 1.

If the value is 0, **Update** returns immediately.

If the value is 1, **Update** won't return until the Array Viewer views have been updated.

Return Value

Returns TRUE if the operation was successful; otherwise, FALSE.

Remarks

Invoke this function when the data or Region of Interest (ROI) has been modified and you want Array Viewer to reflect the new data.

Array Visualizer Controls

The Avis2D and AvisGrid ActiveX (OCX) controls can be used by any development environment that supports ActiveX controls (Visual C++, Visual Basic) to display array data in a variety of graphing modes. The Avis2D and AvisGrid controls provide properties, methods, and events that developers can use to customize their behavior.

The Avis2D and AvisGrid controls let you choose which features to expose to users. You can also use them with other controls.

Samples are provided online in folders under `...\ArrayVisualizer\Samples\`. For a description of the Array Visualizer Samples, view the file `...\ArrayVisualizer\Samples\Samples.htm` in a Web browser.

For more information, see the following sections:

- [Array Visualizer Control Reference](#)
- Array Visualizer 2D Control:
 - [Properties](#)
 - [Methods](#)
 - [Events](#)
- Array Visualizer Grid Control:
 - [Properties](#)
 - [Methods](#)
 - [Events](#)

Array Visualizer Control Reference

The Avis2D and AvisGrid ActiveX controls have properties, methods, and events that you can use in your program to control their on-screen appearance and behavior. Attributes that you can control include the following:

- The data file used
- The part of the data set that is viewable at any given time
- The angle of view and the distance from the graph
- The appearance of the axes and how they are labeled
- The color scheme

Array Visualizer 2D Control Properties

The following tables list the available 2D control properties:

Grouped by Function:

Array Mapping and Properties

[ColDim](#)

[RowDim](#)

[DataExtent](#)

Axis Display

[AxisAutoDetail](#)

[XAxisLabel](#)

[AxisAutoScale](#)

[XScale](#)

[AxisBackColor](#)

[YAxisLabel](#)

[AxisColor](#)

[YScale](#)

[AxisLabelColor](#)

[ZAxisLabel](#)

[AxisLabelFont](#)

[ZScale](#)

[AxisStyle](#)

Camera Position

[CamAzimuth](#)

[CamXPos](#)

[CamDistance](#)

[CamYPos](#)

[CamElevation](#)

[CamZPos](#)

[CamViewType](#)

Center of Interest

[CamXCoi](#)

[CamZCoi](#)

[CamYCoi](#)

Colors[AxisBackColor](#)[BackColor](#)[AxisColor](#)[GraphColor](#)[AxisLabelColor](#)**Constraints on Graph**[XMaxClamp](#)[YMinClamp](#)[XMinClamp](#)[ZMaxClamp](#)[YMaxClamp](#)[ZMinClamp](#)**Data Selection and Display**[DataSelectEnable](#)[ShowDataTip](#)**Fonts**[AxisFont](#)[FontAutoScale](#)[AxisFontName](#)**Graph Rendering**[BarChartArea](#)[Lighting](#)[Depthcue](#)[LineWidth](#)[FixedAspect](#)[OrthographicProj](#)[GraphColor](#)[PointSize](#)[GraphStyle](#)[Resolution](#)[GraphType](#)[ResolutionAutoAdjust](#)[GridColor](#)[Shading](#)[GridDensity](#)[ShowGrid](#)

[GridDensityAutoAdjust](#)[TextureMode](#)[HiddenLine](#)[WComp](#)[HighLight](#)[XComp](#)[Hold](#)[YComp](#)[ImageFilter](#)[ZComp](#)[ImageOrientation](#)

Marker Properties

[Col](#)[Row](#)[CursorColor](#)[ShowCursor](#)[CursorTransparency](#)

Palette

[PaletteAutoAdjust](#)[PaletteMinVal](#)[PaletteFileName](#)[ShowPalette](#)[PaletteId](#)[ShowPaletteLabels](#)[PaletteMaxVal](#)[UseColorPalette](#)

Region of Interest

[ColumnStart](#)[NumRows](#)[NumColumns](#)[RowStart](#)

Scaling of Axes

[AxisAutoScale](#)[XScale](#)[AxisScaleFileName](#)[YScale](#)[UseAxisScales](#)[ZScale](#)

Tick Marks on Axes

[AxisAutoDetail](#) [YNumMinorTickMarks](#)
[XNumMajorTickMarks](#) [ZNumMajorTickMarks](#)
[XNumMinorTickMarks](#) [ZNumMinorTickMarks](#)
[YNumMajorTickMarks](#)

Miscellaneous

[ContextPopup](#) [ViewerMode](#)
[FileName](#)

Listed Alphabetically (see [functional grouping](#) for links):

A

[AxisAutoDetail](#) [AxisFontName](#)
[AxisAutoScale](#) [AxisLabelColor](#)
[AxisBackColor](#) [AxisLabelFont](#)
[AxisColor](#) [AxisScaleFileName](#)
[AxisFont](#) [AxisStyle](#)

B

[BackColor](#)
[BarChartArea](#)

C

[CamAzimuth](#) [CamZCoi](#)
[CamDistance](#) [CamZPos](#)
[CamElevation](#) [Col](#)
[CamViewType](#) [ColDim](#)

[CamXCoi](#)[ColumnStart](#)[CamXPos](#)[ContextPopup](#)[CamYCoi](#)[CursorColor](#)[CamYPos](#)[CursorTransparency](#)**D**[DataExtent](#)[Depthcue](#)[DataSelectEnable](#)**F**[FileName](#)[FontAutoScale](#)[FixedAspect](#)**G**[GraphColor](#)[GridColor](#)[GraphStyle](#)[GridDensity](#)[GraphType](#)[GridDensityAutoAdjust](#)**H**[HiddenLine](#)[Hold](#)[HighLight](#)**I**[ImageFilter](#)[ImageOrientation](#)**L**

[Lighting](#)

[LineWidth](#)

N

[NumColumns](#)

[NumRows](#)

O

[OrthographicProj](#)

P

[PaletteAutoAdjust](#)

[PaletteMaxVal](#)

[PaletteFileName](#)

[PaletteMinVal](#)

[PaletteId](#)

[PointSize](#)

R

[Resolution](#)

[RowDim](#)

[ResolutionAutoAdjust](#)

[RowStart](#)

[Row](#)

S

[Shading](#)

[ShowGrid](#)

[ShowAxis](#)

[ShowPalette](#)

[ShowCursor](#)

[ShowPaletteLabels](#)

[ShowDataTip](#)

T

[TextureMode](#)

U

[UseAxisScales](#)

[UseColorPalette](#)

V

[ViewerMode](#)

W

[WComp](#)

X

[XAxisLabel](#)

[XNumMajorTickMarks](#)

[XComp](#)

[XNumMinorTickMarks](#)

[XMaxClamp](#)

[XScale](#)

[XMinClamp](#)

Y

[YAxisLabel](#)

[YNumMajorTickMarks](#)

[YComp](#)

[YNumMinorTickMarks](#)

[YMaxClamp](#)

[YScale](#)

[YMinClamp](#)

Z

[ZAxisLabel](#)

[ZNumMajorTickMarks](#)

[ZComp](#)

[ZNumMinorTickMarks](#)

[ZMaxClamp](#)[ZScale](#)[ZMinClamp](#)

AxisAutoDetail Property

Gets or sets the value that determines whether the number of both large and small tick marks on the axes is to be handled automatically by Array Visualizer or explicitly by the use of other properties.

Syntax

```
object.AxisAutoDetail [= value]
```

Parameters

value

A **Boolean** expression that determines whether the management of tick marks is to be handled automatically by Array Visualizer. Valid values are TRUE (for automatic tick mark management) and FALSE (to let the **XNumMajorTickMarks**, **XNumMinorTickMarks**, **YNumMajorTickMarks**, **YNumMinorTickMarks**, **ZNumMajorTickMarks**, and **ZNumMinorTickMarks** properties determine the number of both large and small tick marks).

See Also

[XNumMajorTickMarks](#), [XNumMinorTickMarks](#), [YNumMajorTickMarks](#), [YNumMinorTickMarks](#), [ZNumMajorTickMarks](#), [ZNumMinorTickMarks](#)

AxisAutoScale Property

Gets or sets a value that determines how array data values are scaled either to the graph's x- and y-coordinate axes or to the graph's z-coordinate axis.

Syntax

```
object.AxisAutoScale [= value]
```

Parameters

value

A **Boolean** expression that enables or disables autoscaling of the z-range. Valid values are TRUE (to enable) and FALSE (to disable).

Remarks

The effects of this property interact with the setting of the **GraphType** property to determine whether axis clamping affects how axis scaling behaves. The following table shows the dependencies:

IF	THEN		
GraphType property	AxisAutoScale property	x and y clamping can have an effect	z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No
Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

See Also

[GraphType](#), [XMaxClamp](#), [XMinClamp](#), [YMaxClamp](#), [YMinClamp](#), [ZMaxClamp](#), [ZMinClamp](#)

AxisBackColor Property

Gets or sets the color of the bottom and two back sides of the view volume.

Syntax

```
object.AxisBackColor [= color]
```

Parameters

color
An OLE_COLOR.

Remarks

This property has no effect if the **AxisScale** property has a value specifying that back faces should not be filled in.

See Also

[AxisStyle](#)**AxisColor Property**

Gets or sets the color used to draw a graph's axis lines.

Syntax

```
object.AxisColor [= color]
```

Parameters

color

An OLE_COLOR.

Remarks

The **AxisColor** property has no effect if the **ShowAxis** property is set to FALSE.

See Also[ShowAxis](#)**AxisFont Property**

Gets or sets the font used to display the values corresponding to the tick marks on the axes.

Syntax

```
object.AxisFont [= fontname]
```

Parameters

fontname

A pointer expression that describes the properties of the font to be used.

Remarks

The font used must be a TrueType font.

If you are a Visual Basic programmer, you can get or set axis font properties just as you would get or set font properties for any other object that has a font.

If you are using any other language, consult the documentation for that language for information about how to place the appropriate value in the OLE

structure named `IFontDisp` to get or set font properties other than the name of the font.

See Also

[AxisLabelFont](#)

AxisFontName Property

Gets or sets the name of the Font used for Axis Labels.

Syntax

```
object.AxisFontName [= value]
```

Parameters

value

A **String** expression whose value is the name of TrueType Font.

Remarks

Use the **AxisFont** property if you need to control specific font characteristics, such as font size, italic face, underlining, and so forth.

See Also

[AxisFont](#)

AxisLabelColor Property

Gets or sets the color of axis labels, such as x, y, and z.

Syntax

```
object.AxisLabelColor [= color]
```

Parameters

color

An `OLE_COLOR`.

See Also

[AxisLabelFont](#)

AxisLabelFont Property

Gets or sets the font used to display the axis labels, such as x, y, and z.

Syntax

object.**AxisLabelFont** [= *fontname*]

Parameters

fontname

An IFontDisp pointer expression that describes the properties of the font to be used.

Remarks

The font used must be a TrueType font.

If you are a Visual Basic programmer, you can get or set axis font properties just as you would get or set font properties for any other object that has a font.

If you are using any other language, consult the documentation for that language for information about how to place the appropriate value in the OLE structure named IFontDisp to get or set font properties other than the name of the font.

See Also

[AxisLabelColor](#), [AxisFont](#)

AxisScaleFileName Property

Gets or sets the axis scale data filename for the specified dimension.

Syntax

object.**AxisScaleFileName** (*dim*) [= *filename*]

Parameters

dim

A **Short** value that contains the array dimension (indexed from 0 to Rank - 1).

filename

A **String** expression whose value is a path to a valid .AGL data file that contains the scale data.

Remarks

See the **UseAxisScales** property for a description of axis scales. The array specified by filename can be of any type, but must be of rank one. In the Visual Basic language, the **CopyAxisScaleData** method can be used to copy scale data directly as a Basic array.

See Also

[UseAxisScales](#), [CopyAxisScaleData](#)

AxisStyle Property

Gets or sets a single value that determines how many [faces](#) of the [view volume](#) are to be shown, how many of those faces may be filled in, and, separately for each of the X, Y, and Z [grid lines](#), whether those grid lines are to be displayed on the faces that are shown.

Syntax

object.**AxisStyle** [= *stylecode*]

Parameters

stylecode

A **Short** whose value is in the 0-to-20 range and whose effects are as given in the following table:

<i>stylecode</i>	Number of faces	Fill back faces?	Show x grid lines?	Show y grid lines?	Show z grid lines?
0	0	No	No	No	No
1	3	Yes	Yes	Yes	Yes
2	3	Yes	Yes	No	No
3	3	Yes	No	Yes	No
4	3	Yes	No	No	Yes
5	3	Yes	No	No	No
6	3	No	Yes	Yes	Yes
7	3	No	Yes	No	No

8	3	No	No	Yes	No
9	3	No	No	No	Yes
10	3	No	No	No	No
11	6	Yes	Yes	Yes	Yes
12	6	Yes	Yes	No	No
13	6	Yes	No	Yes	No
14	6	Yes	No	No	Yes
15	6	Yes	No	No	No
16	6	No	Yes	Yes	Yes
17	6	No	Yes	No	No
18	6	No	No	Yes	No
19	6	No	No	No	Yes
20	6	No	No	No	No

Remarks

When you choose "3" as the number of faces to be filled, Array Visualizer fills in the bottom face and the two vertical faces farthest from the z-axis.

BackColor Property

Gets or sets the background color of the entire control window.

Syntax

object.**BackColor** [= *color*]

Parameters

color

An OLE_COLOR.

BarChartArea Property

Gets or sets the thickness of bars used in graphs when **GraphStyle** is set to BarChart. The valid range is 0.0 to 1.0.

Syntax

object.**BarChartArea** [= *value*]

Parameters

value

A **Float** expression, in the 0.0 to 1.0 range, that specifies the desired thickness of the bars in bar charts. Smaller values display thin bars and larger values display thicker bars. The value 0 displays bars of 0 thickness, so they are not visible. The value 1 displays bars of maximum thickness, so they are packed together, with no space between them. The default value is 1.

Remarks

The **BarChartArea** property applies only to graphs displayed in the BarChart graph style.

See Also

[GraphStyle](#)

CamAzimuth Property

Gets or sets the positive or negative angle, in degrees, that describes the horizontal arc from line to another.

Syntax

object.**CamAzimuth** [= *angle*]

Parameters

angle

A **Float** expression, with $-180 < \textit{angle} \leq 180$.

Remarks

The **CamAzimuth** property is one of three properties (along with **CamElevation** and **CamDistance**) that determine the position of the camera, relative to the center of interest.

You can calculate the **CamAzimuth** property's value by measuring, in degrees, the counter-clockwise angle from one line to another. The starting line is the one in the x-y plane that is parallel to the x-axis and contains the projection of the COI onto the x-y plane. The ending line is the projection onto the x-y plane of the line between the [camera](#) and the [COI](#). If the calculated angle is larger

than 180°, then subtract 360° from it to place it into the -180° to 180° range, as required.

If the two lines are identical, then the value is either 0 or 180. If the camera is farther from the origin than the COI, then the value is 0. If the COI is farther from the origin, the value is 180.

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamDistance](#), [CamElevation](#), [CamViewType](#)

CamDistance Property

Gets or sets the distance between the [camera](#) and the [center of interest](#).

Syntax

```
object.CamDistance [= distance]
```

Parameters

distance

A positive **Float** expression representing the distance between the camera and the center of interest.

Remarks

The **CamDistance** property is one of three properties (along with **CamAzimuth** and **CamElevation**) that determine the position of the camera relative to the center of interest.

CamDistance is in the same units as the [view volume](#), which is a box-shaped region with (0,0,0) and (**XScale**, **YScale**, **ZScale**) as opposite corners.

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamAzimuth](#), [CamElevation](#), [XScale](#), [YScale](#), [ZScale](#), [CamViewType](#)

CamElevation Property

Gets or sets the [camera](#) elevation angle, relative to the [center of interest](#).

Syntax

object.**CamElevation** [= *angle*]

Parameters

angle

A **Float** expression, with $-90 < \textit{angle} \leq 90$.

Remarks

The **CamElevation** property is one of three properties (along with **CamAzimuth** and **CamDistance**) that determine the position of the camera relative to the center of interest.

It is based on the angle, in degrees, of the line from the COI to the camera, as measured with respect to the x-y plane. If the camera is higher than the COI, then **CamElevation** is the angle measured. If the measurement is downward, then **CamElevation** is the negative of the angle measured.

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamAzimuth](#), [CamDistance](#), [CamViewType](#)

CamViewType Property

Gets or sets the camera view type (3D or 2D).

Syntax

object.**CamViewType** [= *value*]

Parameters

value

An enumerated graph type expression. Valid values are as described in the following table:

value	View Type	Description
0	View3D	3D View -- Camera can be set to any (x,y,z) position
1	ViewXZ	XZ Plane View -- Fixed orthographic view of XZ Plane
2	ViewXY	XY Plane View -- Fixed orthographic view of XY Plane
3	ViewYZ	YZ Plane View -- Fixed orthographic view of YZ Plane

Remarks

The **CamViewType** property provides a convenient way to create a 2D graph view as opposed to setting the camera coordinates to achieve the same effect.

The following properties only have effect when the **CamViewType** is View3D: **OrthographicProj**, **CamDistance**, **CamElevation**, **CamAzimuth**, **CamXPos**, **CamYPos**, **CamZPos**, **CamXCoi**, **CamYCoi**, and **CamZCoi**.

See Also

[OrthographicProj](#), [CamDistance](#), [CamElevation](#), [CamAzimuth](#), [CamXPos](#), [CamYPos](#), [CamZPos](#), [CamXCoi](#), [CamYCoi](#), [CamZCoi](#), [GraphType](#)

CamXCoi Property

Gets or sets the x-coordinate of the [center of interest](#).

Syntax

object.**CamXCoi** [= *value*]

Parameters

value

A **Float** expression equal to the x coordinate of the center of interest (COI), in [view volume](#) coordinates.

Remarks

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamXPos](#), [CamYPos](#), [CamZPos](#), [CamYCoI](#), [CamZCoI](#), [OrthographicProj](#),
[CamViewType](#)

CamXPos Property

Gets or sets the x-coordinate of the [camera](#) viewing position.

Syntax

```
object.CamXPos [= value]
```

Parameters

value

A **Float** expression equal to the x-coordinate of the camera position, in [view volume](#) coordinates.

Remarks

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamYPos](#), [CamZPos](#), [CamXCoI](#), [CamYCoI](#), [CamZCoI](#), [OrthographicProj](#),
[CamViewType](#)

CamYCoI Property

Gets or sets the y-coordinate of the [center of interest](#).

Syntax

```
object.CamYCoI [= value]
```

Parameters

value

A **Float** expression equal to the y-coordinate of the center of interest, in [view volume](#) coordinates.

Remarks

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamXPos](#), [CamYPos](#), [CamZPos](#), [CamXCoi](#), [CamZCoi](#), [OrthographicProj](#), [CamViewType](#)

CamYPos Property

Gets or sets the y-coordinate of the [camera](#) viewing position.

Syntax

```
object.CamYPos [= value]
```

Parameters

value

A **Float** expression equal to the y-coordinate of the camera position, in [view volume](#) coordinates.

Remarks

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamXPos](#), [CamZPos](#), [CamXCoi](#), [CamYCoi](#), [CamZCoi](#), [OrthographicProj](#), [CamViewType](#)

CamZCoi Property

Gets or sets the z-coordinate of the [center of interest](#).

Syntax

```
object.CamZCoi [= value]
```

Parameters

value

A **Float** expression equal to the z-coordinate of the center of interest, in [view volume](#) coordinates.

Remarks

Setting this property has an effect only when the **CamViewType** property is

View3D.

See Also

[CamXPos](#), [CamYPos](#), [CamZPos](#), [CamXCoi](#), [CamYCoI](#), [OrthographicProj](#), [CamViewType](#)

CamZPos Property

Gets or sets the z-coordinate of the [camera](#) viewing position.

Syntax

```
object.CamZPos [= value]
```

Parameters

value

A **Float** expression equal to the z-coordinate of the camera position, in [view volume](#) coordinates.

Remarks

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamXPos](#), [CamYPos](#), [CamXCoi](#), [CamYCoI](#), [CamZCoI](#), [OrthographicProj](#), [CamViewType](#)

Col Property

Gets or sets the column index value that will be used (in conjunction with the [Row](#) property) to place the marker.

Syntax

```
object.Col [= value]
```

Parameters

value

A **Long** expression that indicates the new column position.

ColDim Property

Gets or sets the array dimension that will get mapped to the graph's columns.

Syntax

object.**ColDim** [= *dimension*]

Parameters

dimension

A **Long** expression that specifies the dimension to be mapped to the graph's columns (x-axis).

Remarks

Note: You must call the **Update** method before any change to **ColDim** takes effect.

Both dimension numbers and array indexes begin with index 0.

For arrays of rank 3 or greater, the Avis2D control can only display graphs of a single 2D "slice" of the array. For these higher dimensionality data sets, you can use the **ColDim** and **RowDim** properties to control the direction of this slice and the **SetRoiUB**, and **SetRoiLB** methods to determine which particular slice of the dataset gets mapped to the graph.

See Also

[SetRoiLB](#), [SetRoiUB](#), [RowDim](#), [Update](#)

ColumnStart Property

Gets or sets the index of the leftmost column of the [region of interest \(ROI\)](#).

Syntax

object.**ColumnStart** [= *value*]

Parameters

value

A **Long** expression between 0 and $n-1$, where n is the number of columns in the data array.

Remarks

Collectively, **ColumnStart**, **RowStart**, **NumColumns**, and **NumRows** define

the ROI.

Note: You must call the **Update** method before the new ROI is mapped to the graph.

See Also

[NumColumns](#), [NumRows](#), [RowStart](#)

ContextPopup Property

Gets or sets a value that determines whether the context popup menu will be displayed when the user clicks on the graph with the right mouse button.

Syntax

```
object.ContextPopup [= value]
```

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) the display of the context popup menu.

Remarks

The context popup menu lets you display the Avis2D control's property pages.

CursorColor Property

Gets or sets the color of the row/column marker in the data window.

Syntax

```
object.CursorColor [= color]
```

Parameters

color

An OLE_COLOR.

See Also

[CursorTransparency](#)

CursorTransparency Property

Gets or sets the transparency of the row/column marker in the data window.

Syntax

object.**ColorTransparency** [= *value*]

Parameters

value

A **Float** expression in the range 0.0 (fully transparent) through 1.0 (not at all transparent). Full transparency means invisible and not at all transparent means opaque.

See Also

[CursorColor](#)

DataExtent Property

Gets the extent for the specified array dimension.

Syntax

object.**DataExtent** (*dim*)

Parameters

dim

A **Short** expression that contains the dimension number of an array dimension.

Remarks

Array indexing begins with 0.

DataSelectEnable Property

Gets or sets a value that determines whether DataSelection is enabled.

Syntax

object.**DataSelectEnable** [= *value*]

Parameters

value

A **Boolean** expression that enables (TRUE), or disables (FALSE) Data Selection.

Remarks

If DataSelection is enabled, you can change the current row and col selection by double-clicking on the graph with the left mouse button. If the **ShowDataTip** property is also enabled, datatip windows will be displayed when you hold the mouse over a data element in the graph.

See Also

[ShowDataTip](#)

Depthcue Property

Gets or sets the depth cueing state.

Syntax

```
object.Depthcue [= value]
```

Parameters

value

A **Boolean** expression. When **Depthcue** is TRUE, depth cueing is enabled; otherwise, it is disabled.

Remarks

When enabled, depth cueing “fades out” more distant parts of the graph to provide an enhanced perception of depth. **Depthcue** has no effect unless the **GraphType** property is HeightField.

See Also

[GraphType](#)

FileName Property

Gets or sets the current data file.

Syntax

```
object.FileName [= file]
```

Parameters

file

A **String** expression whose value is a path to the data file.

Remarks

When the **FileName** property points to a valid Array Viewer .AGL file and that file loads, the data in that file replaces any currently loaded data set.

The entire data set in the new file automatically becomes the [region of interest](#). However, if the data set has more than three dimensions, then the data set consists of those array elements whose index is 0 for all dimensions after the third dimension.

Only AGL files can be used as **FileName** properties. The Avis2D control cannot read ascii, or HDF files.

In the Array Viewer application, when a dataset is saved to the AGL format, attribute data describing the current view state is appended to the file. However when the **FileName** property is set in the Avis2D control, only the array data values are read by the control. Attribute data (if any) contained in the file is ignored.

You can also set the **FileName** property to a string obtained by **aglGetShareName** or **faglGetShareName**. This allows the Avis2D control to display array data without saving the array to a file.

See Also

[aglGetShareName](#), [faglGetShareName](#)

FixedAspect Property

Gets or sets a value that determines whether the aspect ratio (the ratio of width to height) is maintained when the control window is resized.

Syntax

```
object.FixedAspect [= value]
```

Parameters

value

A **Boolean** expression, that when TRUE, causes the aspect ratio to be fixed, and when FALSE, allows the aspect ratio to vary with the window size.

Remarks

For Image Map graphs, when Fixed Aspect Ratio is enabled, the relative lengths of the X and Y axis don't change as the graph is resized. If it is disabled, the X and Y axis are independently scaled to fit the view area. This property has no effect for other graph types.

See Also

[GraphType](#)

FontAutoScale Property

Gets or sets the automatic font-scaling state.

Syntax

```
object.FontAutoScale [= value]
```

Parameters

value

A **Boolean** expression that when TRUE, enables automatic font scaling and, when FALSE, disables automatic font scaling.

Remarks

When TRUE, font sizes change automatically with changes in the size of the control window. When FALSE, font size is independent of graph size and, instead, remains as set by the **AxisFont** and **AxisLabelFont** properties.

See Also

[AxisFont](#), [AxisLabelFont](#)

GraphColor Property

Gets or sets a color to be used exclusively for graphing.

Syntax

```
object.GraphColor [= color]
```

Parameters

color

An OLE_COLOR.

Remarks

This property has an effect only when the **UseColorPalette** property is set to FALSE.

See Also

[UseColorPalette](#)

GraphStyle Property

Gets or sets the current graph style.

Syntax

object.**GraphStyle** [= *value*]

Parameters

value

An enumerated graph style expression. Valid values are as described in the following table:

value	View Type	Description
0	Wire Mesh	Graphs drawn as wire frames
1	Surface	Graphs drawn as solid surfaces
2	BarChart	Graphs drawn as collections of vertical bars
3	Lines	Graphs drawn as collections of lines
4	Points	Graphs drawn as collections of points

Remarks

Not all **GraphStyle** values are valid for all **GraphType** property values. The following table shows which combinations are valid and which are not. For example, it is valid to use the Wire Mesh graph style with the Height Field graph type, but not with any other graph type.

	GraphType		
GraphStyle	Height Field	Raster Image	Vector Plot
Wire Mesh	Valid	Invalid	Invalid

Surface	Valid	Valid	Invalid
BarChart	Valid	Invalid	Valid
Lines	Valid	Invalid	Valid
Points	Valid	Invalid	Valid

See Also

[GraphType](#)

GraphType Property

Gets or sets how array data maps to the graph object.

Syntax

object.**GraphType** [= *value*]

Parameters

value

An enumerated graph type expression. Valid values are as described in the following table:

value	Graph Type	Description
--------------	-------------------	--------------------

- | | | |
|---|--------------|--|
| 1 | Height Field | For each data element in the region of interest , there is an (x,y,z) point. The x value is the data element's column index to x, its row index to y, and its data elements value to z. Depending on the current GraphStyle , these points may be displayed as individual points, or connected to form either lines or a surface. |
| 2 | Raster Image | For each data element in the ROI, there is an (x,y) point. The x value is the data element's column index, the y value is its row index, and the color is based on the current color palette . The collection of points is drawn as a 2D image. |
| 3 | Vector Plot | Specifies a graph of vector data, using only the first three columns of data in the array. Each row in the data array maps to a point of the graph, where the x-, y-, and z-coordinates of the point are, respectively, the values in the first three columns of that row of the data array. |

Remarks

Not all **GraphStyle** values are valid for all **GraphType** values. For details, see the [Remarks](#) section of the **GraphStyle** description.

When **GraphType** is set to Vector Plot, the **XComp**, **YComp**, and **ZComp** properties identify the columns in the data array that are to be used for x, y, and z values, respectively.

The combination of the **GraphType** value and the **AxisAutoScale** value can determine which types of clamping can have an effect on how graphs display, as shown in the following table:

IF	THEN		
GraphType property	AxisAutoScale property	X and Y clamping can have an effect	Z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No
Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

There's no "Plane View" **GraphType**. To create an XY Plot, set the **GraphType** to Height Field and the **CamViewType** to ViewXZ.

See Also

[GraphStyle](#), [XComp](#), [YComp](#), [ZComp](#), [CamViewType](#)

GridColor Property

Gets or sets the color of any grid lines drawn.

Syntax

object.**GridColor** [= *color*]

Parameters

color

An OLE_COLOR.

See Also

[AxisStyle](#)

GridDensity Property

Gets or sets a value that determines how finely the grid is drawn on the graph surface.

Syntax

object.**GridDensity** [= *value*]

Parameters

value

A **Float** expression that depends on the data in the data array and on the size of the graph as it is currently being displayed. The number of pixels between adjacent [grid lines](#) is approximately equal to the **GridDensity** value.

Remarks

This property has no effect if the **ShowGrid** property is FALSE or if the **GraphStyle** property is not Surface.

See Also

[ShowGrid](#)

GridDensityAutoAdjust Property

Gets or sets the auto adjust state for grid density.

Syntax

object.**GridDensityAutoAdjust** [= *value*]

Parameters

value

A **Boolean** expression. If TRUE, Avis2D adjusts grid density automatically; if FALSE, grid density is constant.

Remarks

When **GridDensityAutoAdjust** is FALSE, the number of grid lines in the graph stays constant as the size of the control window changes. When **GridDensityAutoAdjust** is TRUE, the number of gridlines varies as the control window size changes, to keep the spacing between grid lines constant.

This property has no effect if the **ShowGrid** property is FALSE or if the **GraphStyle** property is not Surface.

See Also

[GraphStyle](#), [GridDensity](#), [ShowGrid](#)

HiddenLine Property

Gets or sets the status of the automatic removal of line segments that are behind another part of the graph.

Syntax

```
object.HiddenLine [= value]
```

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) hidden line removal.

Remarks

The **HiddenLine** property has no effect unless **GraphStyle** is Wire Mesh or Lines.

See Also

[GraphStyle](#)

HighLight Property

Gets or sets the HighLight state.

Syntax

object.**HighLight** [= *value*]

Parameters

value

A **Boolean** expression that enables (TRUE) or disables glossy highlights on the graph's surface.

Remarks

This property only has effect when **Lighting** is enabled, the **GraphType** is HeightField and the **TextureMode** is NO_TEXTURE.

When highlights are enabled, the surface has a more photo-realistic appearance, but the highlights can make it hard to determine the true color values of that region of the graph.

See Also

[Lighting](#), [GraphType](#), [TextureMode](#)

Hold Property

Gets or sets a value that determines whether the graph is updated immediately upon a change of any property value, or only when the **Update** method is called.

Syntax

object.**Hold** [= *value*]

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) the Hold value.

Remarks

Set Hold to TRUE if you want to make changes to several property values without having the graph update until desired.

See Also

[Update](#)

ImageFilter Property

Gets or sets the ImageFilter state.

Syntax

```
object.ImageFilter [= value]
```

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) Image Filtering.

Remarks

When enabled, image filtering blends adjacent colors of ImageMap graphs. When disabled, ImageMap graphs will display a "tiled" appearance. The visual difference between these two modes is most noticeable in graphs with fewer datapoints.

Setting **ImageFilter** has no effect on other graph types.

See Also

[GetGraphType](#), [SetGraphType](#)

ImageOrientation Property

Gets or sets the image orientation. In Raster Image graphs, this property controls how the graph is mapped to the window.

Syntax

```
object.ImageOrientation [= value]
```

Parameters

value

An enumerated graph type expression. Valid values are as described in the following table:

value	Orientation Type	Description
0	Identity	Index values increase from left to right and bottom to top
1	XFlip	Index value increase from right to left and bottom to top
2	YFlip	Index value increase from left to right and top to bottom
3	ZFlip	Index values increase from right to left and top to bottom

Remarks

This property applies only to graphs whose **GraphType** property is ImageMap.

See Also

[GraphType](#)

Lighting Property

Gets or sets the lighting effects state.

Syntax

```
object.Lighting [= value]
```

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) the use of lighting effects.

Remarks

When **Lighting** is enabled, the color of each point on the graph is determined by a mathematical relationship between the light source, the current [camera](#) position, and the base color of the graph at the [center of interest](#).

When **Lighting** is disabled, the color of each point on the graph is determined solely by the base color for that point.

LineWidth Property

Gets or sets the thickness, in pixels, of lines (not grid lines) in graphs.

Syntax

```
object.LineWidth [= value]
```

Parameters

value

A **Float** expression, greater than 0, that sets the line width.

Remarks

The **LineWidth** property applies only to the Line and Wire Mesh graph styles.

See Also

[GraphStyle](#)

NumColumns Property

Gets or sets the number of columns in the [region of interest \(ROI\)](#).

Syntax

```
object.NumColumns [= value]
```

Parameters

value

A **Long** expression between 0 and the number of columns in the data array.

Remarks

Collectively, **ColumnStart**, **RowStart**, **NumColumns**, and **NumRows** define the ROI.

Note: You must call the **Update** method before the new ROI is mapped to the graph.

See Also

[ColumnStart](#), [NumRows](#), [RowStart](#), [Update](#)

NumRows Property

Gets or sets the number of rows in the [region of interest \(ROI\)](#)

Syntax

```
object.NumRows [= value]
```

Parameters

value

A **Long** expression between 0 and the number of rows in the data array.

Remarks

Collectively, **ColumnStart**, **RowStart**, **NumColumns**, and **NumRows** define the ROI.

Note: You must call the **Update** method before the new ROI is mapped to the graph.

See Also

[ColumnStart](#), [NumColumns](#), [RowStart](#), [Update](#)

OrthographicProj Property

Gets or sets a value that determines whether graphs are rendered orthographically (where distance does not affect size) or in perspective (where size varies according to distance).

Syntax

```
object.OrthographicProj [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, enables orthographic projection.

Remarks

Setting this property has an effect only when the **CamViewType** property is View3D.

See Also

[CamViewType](#)

PaletteAutoAdjust Property

Gets or sets a value that determines how a graphs data values are mapped into colors.

Syntax

```
object.PaletteAutoAdjust [= value]
```

Parameters

value

A **Boolean** expression that, if TRUE, enables **PaletteAutoAdjust**, and, if FALSE, disables **PaletteAutoAdjust**.

Remarks

When **PaletteAutoAdjust** is enabled, the minimum data value in the [region of interest \(ROI\)](#) maps to the bottom of the [palette](#) color range, and the maximum data value to the top. All other data values map to the color palette proportionally. When **PaletteAutoAdjust** is disabled, data values less than or equal to **PaletteMinVal** map to the bottom of the palette color range, and data values greater than or equal to **PaletteMaxVal** map to the top. Data values between **PaletteMinVal** and **PaletteMaxVal** map to the color palette proportionally.

Note that if the **AxisAutoScale** property is disabled, then data values are limited to the **ZMinClamp** to **ZMaxClamp** range.

See Also

[AxisAutoScale](#), [PaletteMaxVal](#), [PaletteMinVal](#), [ZMaxClamp](#), [ZMinClamp](#)

PaletteFileName Property

Gets or sets the current file name for custom palette data.

Syntax

```
object.PaletteFileName [= value]
```

Parameters

value

A **String** expression whose value is a path to a valid .AGL data file.

Remarks

The rank of the specified data file must be 1 and the total size must be 1024 bytes (i.e. 1024 chars, 512 shorts, or 256 longs).

The **Paletteld** property must be set to Custom for the palette data to be displayed.

In the Visual Basic language, the **CopyPaletteData** method can be used to directly copy a Basic array as Palette data.

See Also

[Paletteld](#)

Paletteld Property

Gets or sets the palette ID used to map data values to colors.

Syntax

object.**Paletteld** [= *value*]

Parameters

value

One of the integers in the range 0 through 6, with effects as follows:

value	Name	Description
0	Custom	Customized palette mapping
1	Grayscale	Graduated from white at top of graph to black at bottom
2	Grayscale_Banded	Similar to Grayscale but with horizontal bands of darker gray
3	Grayscale_Inverted	Graduated from black at top of graph to white at bottom
4	Rainbow	Graduated from start of palette at bottom of graph to end of palette at top
5	Rainbow_Banded	Similar to Rainbow, but with horizontal bands of darker shades
6	Rainbow_Inverted	Graduated from end of palette at bottom of graph to start of palette at top

See Also

[CopyPaletteData](#)

PaletteMaxVal Property

Gets or sets a value that (along with **PaletteMinVal** and **PaletteAutoAdjust**) define how a graph's data values map to the current color [palette](#). (See **PaletteAutoAdjust** for a more detailed description.)

Syntax

```
object.PaletteMaxVal [= value]
```

Parameters

value

A **Double** expression that identifies the maximum data value with which a palette entry is to be associated.

Remarks

If **PaletteAutoScale** is FALSE, **PaletteMaxVal** has no effect.

See Also

[PaletteAutoAdjust](#), [PaletteMinVal](#)

PaletteMinVal Property

Gets or sets a value that (along with **PaletteMaxVal** and **PaletteAutoAdjust**) defines how a graph's data values map to the current color [palette](#). (See **PaletteAutoAdjust** for a more detailed description.)

Syntax

```
object.PaletteMaxVal [= value]
```

Parameters

value

A **Double** expression that identifies the minimum data value with which a palette entry is to be associated.

Remarks

If **PaletteAutoScale** is FALSE, **PaletteMinVal** has no effect.

See Also

[PaletteAutoAdjust](#), [PaletteMaxVal](#)

PointSize Property

Gets or sets the diameter of displayed points.

Syntax

```
object.PointSize [= value]
```

Parameters

value

A **Float** expression denoting the desired diameter of a point, in pixels.

Remarks

Depending upon the visual size and density of displayed points, you might need to increase or decrease their size. To do this, use the **PointSize** property.

Resolution Property

Gets or sets the level of detail in which data points within the [region of interest \(ROI\)](#) map to a graph.

Syntax

```
object.Resolution [= value]
```

Parameters

value

A **Long** expression greater than or equal to 1.

Remarks

If **Resolution** is 1, there is a one-to-one mapping of data points to points in the graph. When the resolution is 2, only one out of four data points is displayed. Each further increment of the resolution value decreases the detail level of the graph by a factor of 4. For example, a resolution of 3 would draw one out of 16 points.

Using a lower **Resolution** value (such as 1) enables a more accurate depiction of the data values in the graph, but can result in slow rendering when data sets

are large.

See Also

[ResolutionAutoAdjust](#)

ResolutionAutoAdjust Property

Gets or sets a value that determines whether the level of detail of graphing is to be handled automatically or, using the **Resolution** property, explicitly.

Syntax

```
object.ResolutionAutoAdjust [= value]
```

Parameters

value

A **Boolean** expression that, if TRUE, enables **ResolutionAutoAdjust**, and, if FALSE, disables **ResolutionAutoAdjust**.

Remarks

When this property is TRUE, Array Visualizer dynamically adjusts the resolution of the graph so that performance matches the demands set by any user activity. Otherwise, the resolution is static at the level indicated by the current value of the **Resolution** property.

See Also

[Resolution](#)

Row Property

Gets or sets the row index value that will be used (in conjunction with the [Col](#) property) to place the marker.

Syntax

```
object.Row [= value]
```

Parameters

value

A **Long** expression that indicates the new row position.

RowDim Property

Gets or sets the array dimension that will get mapped to the graph's rows.

Syntax

```
object.RowDim [= dimension]
```

Parameters

dimension

A **Long** expression that specifies the dimension to be mapped to the graph's rows (y-axis).

Remarks

Note: You must call the **Update** method before any change to **RowDim** takes effect.

Both dimension numbers and array indexes begin with index 0.

For arrays of rank 3 or greater, the Avis2D control can only display graphs of a single 2D "slice" of the array. For these higher dimensionality data sets, you can use the **ColDim** and **RowDim** properties to control the direction of this slice and the **SetRoiUB**, and **SetRoiLB** methods to determine which particular slice of the dataset gets mapped to the graph.

See Also

[SetRoiLB](#), [SetRoiUB](#), [ColDim](#), [Update](#)

RowStart Property

Gets or sets the row index of the topmost row in the [region of interest \(ROI\)](#).

Syntax

```
object.RowStart [= value]
```

Parameters

value

A **Long** expression between 0 and $n-1$, where n is the number of rows in the data set.

Remarks

Row numbering starts at 0 and goes to $n-1$, where n is the number of rows in

the data set.

Note: You must call the **Update** method before the new ROI is mapped to the graph.

Collectively, **ColumnStart**, **RowStart**, **NumColumns**, and **NumRows** define the region of interest.

See Also

[ColumnStart](#), [NumColumns](#), [NumRows](#)

Shading Property

Gets or sets a value that determines how color values are interpolated across the graph's surface (or line segment).

Syntax

```
object.Shading [= value]
```

Parameters

value

A **Short** expression that can have one of only two values, as follows:

value	Name	Description
1	FLAT	Pixel values on the graph change only at data points, so the graph has a faceted appearance.
2	SMOOTH	Color interpolation between data points prevents sharp color transitions.

Remarks

This property applies only to graphs whose **GraphType** property is Height Field and whose **GraphStyle** property is Surface, Wire Mesh or Lines.

The effect of this property is more noticeable when the data dimensions are relatively small.

See Also

[GraphStyle](#), [GraphType](#), [TextureMode](#)

ShowAxis Property

Gets or sets a value that determines whether the graph axes are visible.

Syntax

```
object.ShowAxis [= value]
```

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) the display of axes.

ShowCursor Property

Gets or sets a value that determines whether the row and col marker will be displayed.

Syntax

```
object.ShowCursor [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, causes the marker to be displayed. When FALSE, it causes the marker to be hidden.

See Also

[Row](#), [Col](#)

ShowDataTip Property

Gets or sets a value that determines whether data points and their values automatically display in a small [data tip](#) window when you place the mouse over them.

Syntax

```
object.ShowDataTip [= value]
```

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) the display of data points and their values.

Remarks

The **DataSelectEnable** property must be enabled.

See Also

[DataSelectEnable](#)

ShowGrid Property

Gets or sets a value that determines whether [grid lines](#) for graphs display along with their graphs.

Syntax

```
object.ShowGrid [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, displays the grid lines for surface graphs on the surfaces themselves and, when FALSE, does not display any grid lines.

Remarks

This property applies only to graphs whose **GraphStyle** property is Surface.

See Also

[GraphStyle](#)

ShowPalette Property

Gets or sets a value that determines whether the color palette will be displayed to the right of the graph.

Syntax

```
object.ShowPalette [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE causes the palette to be displayed, and when FALSE, causes the palette to be hidden.

Remarks

Show or hiding the screen representation of the palette does not effect how the palette is used to color the graph.

ShowPaletteLabels Property

Gets or sets a value that determines whether labels should be displayed to the right of the color palette.

Syntax

```
object.ShowPaletteLabels [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, causes palette labels to be displayed (providing that the control window is sufficiently large). When FALSE, it causes the palette labels to be hidden.

Remarks

Setting this property has no effect if **ShowPalette** is FALSE.

See Also

[ShowPalette](#)

TextureMode Property

Gets or sets a value that determines how color interpolation is to be handled across the surface of graphs.

Syntax

```
object.TextureMode [= value]
```

Parameters

value

A **Short** whose value determines the precision of color interpolation across the surface of a graph. Valid values follow:

value	Name	Description
0	NO_TEXTURE	Do color interpolation, but emphasize speed, not precision.
1	ALWAYS_TEXTURE	Do color interpolation with full precision.
2	AUTO_TEXTURE	First, render a graph with no color interpolation. Then, if the region of interest (ROI) has not changed, do color interpolation with full precision.

Remarks

If **Shading** is FLAT, **TextureMode** has no effect. The visual difference between different **TextureMode** values is most noticeable when the number of data points in the graph is small.

This property applies only to graphs whose **GraphStyle** property is Surface and whose **GraphType** property is Height Field.

The AUTO_TEXTURE mode is not currently implemented.

See Also

[GraphStyle](#), [Shading](#)

UseAxisScales Property

Gets or sets a value that determines if Axis Scale data should be displayed.

Syntax

object.**UseAxisScales** [= *value*]

Parameters

value

A **Boolean** expression that enables (TRUE) or disables (FALSE) display of Axis Scale labels.

Remarks

Axis Scales allow custom labels to be displayed along the dimension axis, as opposed to array index values. The **UseAxisScales** property can be used to toggle between Axis Scale values and array index values.

The **AxisScaleFileName** property or **CopyAxisScaleData** method must be

invoked to pass the Axis Scale data to the control.

See Also

[AxisScaleFileName](#), [CopyAxisScaleData](#)

UseColorPalette Property

Gets or sets a value that either allows or disallows the use of the current color [palette](#) for graphing.

Syntax

```
object.UseColorPaletter [= value]
```

Parameters

value

A **Boolean** expression that, if TRUE, enables graphing using the current color palette and, if FALSE, maps all data values to the color specified by the **GraphColor** property.

See Also

[GraphColor](#), [Paletteld](#)

ViewerMode Property

Gets or sets the manner in which a user can use the mouse to change the viewpoint.

Syntax

```
object.ViewerMode [= value]
```

Parameters

value

A **Short** expression whose value affects the user's ability to manipulate the graph with the mouse. Valid values follow:

value	Name	Description
0	FIXED	The camera can't be moved using the mouse.
1	EXAMINE	Dragging the mouse with the left button down rotates the camera around the center of interest .

- | | | |
|---|-----|---|
| 2 | PAN | Dragging the mouse with the left button down moves the center of interest and the camera in tandem. |
|---|-----|---|

Remarks

When ViewerMode is set to 1 (EXAMINE), you can zoom in and out by holding the Shift key down and using the left mouse button. Place the mouse cursor in the graph window, press and hold the Shift key and the left mouse button, then drag the mouse up the screen to zoom in, or down the screen to zoom out.

WComp Property

Gets or sets the number of a column whose values are to be used as color identifiers for vector plotting.

Syntax

object.**WComp** [= *value*]

Parameters

value

A **Long** expression containing an OLE_COLOR.

Remarks

This property applies only when the **GraphType** property is set to Vector Plot.

During vector plotting, the data array is used in a special way. The information in each row contributes one point to the graph. From a row, one or more (usually three) columns contribute the x-, y-, and z-coordinates of the point for that row, and another column contributes a value that determines the color of the point. The purpose of this property is to identify a column whose values are to be used as color identifiers. The **XComp**, **YComp**, and **ZComp** properties identify the other columns.

See Also

[GraphType](#), [XComp](#), [YComp](#), [ZComp](#)

XAxisLabel Property

Gets or sets the label of the x-axis on the graph.

Syntax

object.XAxisLabel [= *value*]

Parameters

value

A **BSTR** expression containing the displayed name of the x-axis.

See Also

[AxisLabelColor](#), [AxisLabelFont](#), [ShowAxis](#)

XComp Property

Gets or sets the number of a data array column whose values are to be used as x-coordinate values for vector plotting.

Syntax

object.XComp [= *value*]

Parameters

value

A **Long** expression containing the number of a column in the data array.

Remarks

This property applies only when the **GraphType** property is set to Vector Plot.

During vector plotting, the data array is used in a special way. The information in each row contributes one point to the graph. From a row, one or more (usually three) columns contribute the x-, y-, and z-coordinates of the point for that row, and another column contributes a value that determines the color of the point. The purpose of this property is to identify a column whose values are to be used as the x-coordinates of points. The **YComp**, **ZComp**, and **WComp** properties identify the other columns.

See Also

[GraphType](#), [WComp](#), [YComp](#), [ZComp](#)

XMaxClamp Property

Gets or sets the upper bound for x-coordinate values.

Syntax

object.XMaxClamp [= *value*]

Parameters

value

A **Double** expression containing the upper bound for x-coordinates.

Remarks

This property has no effect under certain circumstances, as the following table shows:

IF	THEN		
GraphType property	AxisAutoScale property	x and y clamping can have an effect	z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No
Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

See Also

[AxisAutoScale](#), [GraphType](#), [XMinClamp](#)

XMinClamp Property

Gets or sets the lower bound for x-coordinate values.

Syntax

object.XMinClamp [= *value*]

Parameters

value

A **Double** expression that contains the lower bound for x-coordinates.

Remarks

This property has no effect under certain circumstances, as the following table shows:

IF	THEN		
GraphType property	AxisAutoScale property	x and y clamping can have an effect	z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No
Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

See Also

[AxisAutoScale](#), [GraphType](#), [XMaxClamp](#)

XNumMajorTickMarks Property

Gets or sets the number of large tick marks shown on the x-axis.

Syntax

object.XNumMajorTickMarks [= *value*]

Parameters

value

A **Short** expression that contains the number of large tick marks shown on the x-axis.

Remarks

This property has no effect unless the **AxisAutoDetail** property is set to FALSE.

See Also

[AxisAutoDetail](#), [XNumMinorTickMarks](#)

XNumMinorTickMarks Property

Gets or sets the number of small tick marks shown between each adjacent pair of large tick marks on the x-axis.

Syntax

```
object.XNumMinorTickMarks [= value]
```

Parameters

value

A **Short** expression that contains the number of small tick marks shown between large tick marks on the x-axis.

Remarks

This property has no effect unless the **AxisAutoDetail** property is set to FALSE.

See Also

[AxisAutoDetail](#), [XNumMajorTickMarks](#)

XScale Property

Gets or sets the scaling factor for the x-axis.

Syntax

```
object.XScale [= value]
```

Parameters

value

A **Float** expression that indicates how the x-axis scales.

Remarks

When **XScale**, **YScale**, and **ZScale** are 1.0 and **AxisAutoScale** is TRUE (all default values), graphs typically fill the graph region of the control window exactly. You can stretch or shrink graphs in any of these dimensions by making the scaling values larger or smaller, respectively.

Note that stretching a graph often removes part of it from view.

See Also

[AxisAutoScale](#), [YScale](#), [ZScale](#)

YAxisLabel Property

Gets or sets the label of the y-axis on the graph.

Syntax

```
object.YAxisLabel [= value]
```

Parameters

value

A **BSTR** expression containing the displayed name of the y-axis.

See Also

[AxisLabelColor](#), [AxisLabelFont](#), [ShowAxis](#)

YComp Property

Gets or sets the number of a column whose values are to be used as y-coordinate values for vector plotting.

Syntax

```
object.YComp [= value]
```

Parameters

value

A **Long** expression containing the number of a column in the data array.

Remarks

This property applies only when the **GraphType** property is set to Vector Plot.

During vector plotting, the data array is used in a special way. The information in each row contributes one point to the graph. From a row, one or more (usually three) columns contribute the x-, y-, and z-coordinates of the point for that row, and another column contributes a value that determines the color of the point. The purpose of this property is to identify a column whose values are to be used as the y-coordinates of points. The **XComp**, **ZComp**, and **WComp** properties identify the other columns.

See Also

[GraphType](#), [WComp](#), [XComp](#), [ZComp](#)

YMaxClamp Property

Gets or sets the upper bound for y-coordinate values.

Syntax

object.**YMaxClamp** [= *value*]

Parameters

value

A **Double** expression that contains the upper bound for y-coordinates.

Remarks

This property has no effect under certain circumstances, as the following table shows:

IF	THEN		
GraphType property	AxisAutoScale property	x and y clamping can have an effect	z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No

Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

See Also

[AxisAutoScale](#), [GraphType](#), [YMinClamp](#)

YMinClamp Property

Gets or sets the lower bound for the y-coordinate values.

Syntax

object.**YMinClamp** [= *value*]

Parameters

value

A **Double** expression that contains the lower bound for y-coordinates.

Remarks

This property has no effect under certain circumstances, as the following table shows:

IF	THEN		
GraphType property	AxisAutoScale property	x and y clamping can have an effect	z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No
Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

See Also

[AxisAutoScale](#), [GraphType](#), [YMaxClamp](#)

YNumMajorTickMarks Property

Gets or sets the number of large tick marks shown on the y-axis.

Syntax

```
object.YNumMajorTickMarks [= value]
```

Parameters

value

A **Short** expression that contains the number of large tick marks shown on the y-axis.

Remarks

This property has no effect unless the **AxisAutoDetail** property is set to FALSE.

See Also

[AxisAutoDetail](#), [YNumMinorTickMarks](#)

YNumMinorTickMarks Property

Gets or sets the number of small tick marks shown between each adjacent pair of large tick marks on the y-axis.

Syntax

```
object.YNumMinorTickMarks [= value]
```

Parameters

value

A **Short** expression that contains the number of small tick marks shown between large tick marks on the y-axis.

Remarks

This property has no effect unless the **AxisAutoDetail** property is set to FALSE.

See Also

[AxisAutoDetail](#), [YNumMajorTickMarks](#)

YScale Property

Gets or sets the scaling factor for the y-axis.

Syntax

```
object.YScale [= value]
```

Parameters

value

A **Float** expression that indicates how the y-axis scales.

Remarks

When **XScale**, **YScale**, and **ZScale** are 1.0 and **AxisAutoScale** is TRUE (all default values), graphs typically fill the graph region of the control window. You can stretch or shrink graphs in any of these dimensions by making the scaling values larger or smaller, respectively.

Note that stretching a graph often removes part of it from view.

See Also

[AxisAutoScale](#), [XScale](#), [ZScale](#)

ZAxisLabel Property

Gets or sets the z-axis label.

Syntax

```
object.ZAxisLabel [= value]
```

Parameters

value

A **BSTR** expression containing the displayed name of the z-axis.

See Also

[AxisLabelColor](#), [AxisLabelFont](#), [ShowAxis](#)

ZComp Property

Gets or sets the number of a column whose values are to be used as z-coordinate values for vector plotting.

Syntax

object.**ZComp** [= *value*]

Parameters

value

A **Long** expression containing the number of a column in the data array.

Remarks

This property applies only when the **GraphType** property is set to Vector Plot.

During vector plotting, the data array is used in a special way. The information in each row contributes one point to the graph. From a row, one or more (usually three) columns contribute the x-, y-, and z-coordinates of the point for that row, and another column contributes a value that determines the color of the point. The purpose of this property is to identify a column whose values are to be used as the z-coordinates of points. The **XComp**, **YComp**, and **WComp** properties identify the other columns.

See Also

[GraphType](#), [WComp](#), [XComp](#), [YComp](#)

ZMaxClamp Property

Gets or sets the upper bound for the z-coordinate values.

Syntax

object.**ZMaxClamp** [= *value*]

Parameters

value

A **Double** expression that contains the upper bound for z-coordinates.

Remarks

This property has no effect under certain circumstances, as the following table shows:

IF	THEN		
GraphType property	AxisAutoScale property	x and y clamping can have an effect	z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No
Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

See Also

[AxisAutoScale](#), [GraphType](#), [ZMinClamp](#)

ZMinClamp Property

Gets or sets the lower bound for the z-coordinate values.

Syntax

object.**ZMinClamp** [= *value*]

Parameters

value

A **Double** expression that contains the lower bound for z-coordinates.

Remarks

This property has no effect under certain circumstances, as the following table shows:

IF	THEN		
GraphType property	AxisAutoScale property	x and y clamping can have an effect	z clamping can have an effect
Height Field	TRUE	No	No
Height Field	FALSE	No	Yes
Raster Image	TRUE	No	No
Raster Image	FALSE	No	No
Vector Plot	TRUE	No	No
Vector Plot	FALSE	Yes	Yes

See Also

[AxisAutoScale](#), [GraphType](#), [ZMaxClamp](#)

ZNumMajorTickMarks Property

Gets or sets the number of large tick marks shown on the z-axis.

Syntax

object.**ZNumMajorTickMarks** [= *value*]

Parameters

value

A **Short** expression that contains the number of large tick marks shown on the z-axis.

Remarks

This property has no effect unless the **AxisAutoDetail** property is set to FALSE.

See Also

[AxisAutoDetail](#), [ZNumMinorTickMarks](#)

ZNumMinorTickMarks Property

Gets or sets the number of small tick marks shown between each adjacent pair of large tick marks on the z-axis.

Syntax

```
object.ZNumMinorTickMarks [= value]
```

Parameters

value

A **Short** expression that contains the number of small tick marks shown between large tick marks on the z-axis.

Remarks

This property has no effect unless the **AxisAutoDetail** property is set to FALSE.

See Also

[AxisAutoDetail](#), [ZNumMajorTickMarks](#)

ZScale Property

Gets or sets the scaling factor for the z-axis.

Syntax

```
object.ZScale [= value]
```

Parameters

value

A **Float** expression that indicates how scaling takes place with respect to the z-axis.

Remarks

When **XScale**, **YScale**, and **ZScale** are 1.0 and **AxisAutoScale** is TRUE (all default values), graphs typically fill the graph region of the control window. You can stretch or shrink graphs in any of these dimensions by making the scaling values larger or smaller, respectively.

Note that stretching a graph often removes part of it from view.

See Also

[AxisAutoScale](#), [XScale](#), [YScale](#)

Array Visualizer 2D Control Methods

The following table lists the available 2D control methods:

Method

[About](#)

[CopyArrayData](#)

[CopyAxisScaleData](#)

[CopyPaletteData](#)

[GetLBound](#)

[GetRoi](#)

[SetHomePosition](#)

[SetLBound](#)

[SetRoi](#)

[SetRoiLB](#)

[SetRoiUB](#)

[ToHomePosition](#)

[Update](#)

About Method

Displays an About dialog box for the Array Visualizer control.

Syntax

```
object.About ( )
```

Parameters

None.

Return Value

None.

CopyArrayData Method

In the Visual Basic language, this method can be used to pass a data array to the control.

Syntax

success = *object*.**CopyArrayData** (*array*)

Parameters

array

A Visual Basic array that contains the data to be graphed.

Return Value

success

A **Boolean** value indicating success (TRUE) or failure (FALSE).

Remarks

CopyArrayData completely replaces any data that may have been previously passed to the control by the **FileName** property or **CopyArrayData** method.

See Also

[FileName](#)

CopyAxisScaleData Method

In the Visual Basic language, this method can be used to pass a axis scale data array to the control.

Syntax

success = *object*.**CopyAxisScaleData** (*dim*, *array*)

Parameters

dim

A **Short** expression that contains the dimension number of an array dimension.

array

A Visual Basic array that contains the axis scale data for the indicated dimension.

Return Value

success

A **Boolean** value indicating success (TRUE) or failure (FALSE).

Remarks

CopyAxisScaleData completely replaces any data that might have been previously passed to the control by the **FileName** property, or **CopyArrayData** method.

See Also

[Filename](#), [CopyArrayData](#)

CopyPaletteData Method

In the Visual Basic language, this method can be used to pass data for a custom palette to the control.

Syntax

```
success = object.CopyPaletteData (array)
```

Parameters

array

A Visual Basic array that contains 256 **Long** values that define the palette.

Return Value

success

A **Boolean** value indicating success (TRUE) or failure (FALSE).

Remarks

To have the control use the custom palette, set the **PaletteId** property to Custom.

See Also

[PaletteId](#)

GetLBound Method

Gets the lower bound value for the specified dimension.

Syntax

```
lbound = object.GetLBound (dim)
```

Parameters

dim

A **Short** value that contains the array dimension (indexed from 0 to Rank - 1) of the lower bound value to be retrieved.

Return Value

lbound

A **Long** containing the lower bound value of the given array dimension.

Remarks

See the Remarks in **SetLBound**.

See Also

[SetLBound](#)

GetRoi Method

Gets the position number of the first or last active elements in the [region of interest \(ROI\)](#) for the specified array dimension.

Syntax

indexvalue = *object*.**GetRoi** (*dim*, *LBoRUB*)

Parameters

dim

A **Short** expression that contains the dimension number of an array dimension.

LBoRUB

A **Short** expression that contains the value 0, if the value of first active element in the ROI is desired. It contains the value 1, if the value of the last active element in the ROI is desired.

Return Value

indexvalue

A **Long** containing the position number of the first or last active element in the specified dimension for the array.

Remarks

Both dimension numbering and array indexing begin with 0.

See Also[SetROI](#)**SetHomePosition Method**

Sets the current [camera](#) position as the "home" viewpoint.

Syntax

```
object.SetHomePosition ( )
```

Parameters

None.

Return Value

None.

Remarks

Whatever the camera position is when this method is called remains the home position, until it is changed by another call to **SetHomePosition**.

See Also[ToHomePosition](#)**SetLBound Method**

Sets the lower bound value for the specified dimension.

Syntax

```
object.SetLBound (dim, value)
```

Parameters

dim

The array dimension (indexed from 0 to Rank - 1) of the lower bound value to be modified.

value

The new lower bound value.

Return Value

None.

Remarks

The lower bound value is the value used to index the first array element for that dimension.

By default, arrays created in the Fortran language have a default lower bound value of 1. In the C language, arrays have a default lower bound value of 0. If desired, you can use the **SetLBound** method to specify a different lower bound value.

Regardless of the value specified in **SetLBound** and the language used to create the array, Avis2D properties and methods that take an index argument always use zero-based indexing.

See Also

[GetLBound](#)

SetRoi Method

Sets the position number of the first or last active elements in the [region of interest \(ROI\)](#) for the specified array dimension.

Syntax

success = *object*.**SetRoi** (*dim*, *LB*, *UB*)

Parameters

dim

A **Short** expression that contains the dimension number of an array dimension.

LB

A **Long** expression that contains the value of the desired first element in the ROI.

UB

A **Long** expression that contains the value of the desired last element in the ROI.

Return Value

success

A **Boolean** value indicating success (TRUE) or failure (FALSE).

Remarks

You must call the **Update** method before the new ROI is mapped to the graph.

Both dimension numbering and array indexing begin with 0.

For multi-dimensional arrays, use the **RowDim** and **ColDim** properties to control which array dimensions get mapped to the graph's rows and columns.

See Also

[GetROI](#), [RowDim](#), [ColDim](#)

SetRoiLB Method

Sets the position number of the first element in the [region of interest \(ROI\)](#) of the specified array dimension.

Syntax

indexvalue = *object*.**SetRoiLB** (*dimension*)

Parameters

dimension

A **Short** value that contains the dimension number of an array dimension.

Return Value

indexvalue

A **Long** value that contains the position number of the first active element in the specified dimension for the array.

Remarks

Note: You must call the **Update** method before the new ROI is mapped to the graph.

Both dimension numbers and array indexes begin with index 0.

See Also

[GetRoi](#), [SetRoiUB](#), [Update](#)

SetRoiUB Method

Sets the position number of the last element of the specified array dimension in the [region of interest \(ROI\)](#).

Syntax

```
indexvalue = object.SetRoiUB (dimension)
```

Parameters

dimension

A **Short** value that contains the dimension number of an array dimension.

Return Value

indexvalue

A **Long** value that contains the position number of the last active element in the specified dimension for the array.

Remarks

Note: You must call the **Update** method before the new ROI is mapped to the graph.

Both dimension numbers and array indexes begin with index 0.

See Also

[GetRoi](#), [SetRoiLB](#), [Update](#)

ToHomePosition Method

Returns the [camera](#) to the home position.

Syntax

```
object.ToHomePosition ( )
```

Parameters

None.

Return Value

None.

Remarks

The home position is the default position, unless it has been changed by means of the **SetHomePosition** method.

See Also

[SetHomePosition](#)

Update Method

Updates the graph to reflect any changes in either the array data or the [ROI](#) since the last **Update** call.

Syntax

```
object.Update ( )
```

Parameters

None.

Return Value

None.

Remarks

Call **Update** when the data or ROI has been modified in some way, and you want to update the graph to reflect the changes.

The repainting of the graph occurs in the background. **Update** always returns immediately, while the (perhaps lengthy) processing required to redraw the graph based on the new data values is performed by a background thread.

If you want to make sure that the graph has been redrawn with the new dataset after making an **Update** call, do the following:

1. Call **Update**.
2. Then wait for a **RndrPass** event with an *updatestate* value of 2 to occur.

At that point, the screen image of the graph will reflect the array data.

See Also

[RndrPass](#)

Array Visualizer 2D Control Events

The following table lists the available 2D control events:

Event

[DataSelect](#)

[RndrPass](#)

[RowColChanged](#)

[VDataSelect](#)

DataSelect Event

This event is fired by the control when you double-click on the graph with the left mouse button, or when you hold the mouse pointer steady over a data element in the graph.

Syntax

DataSelect (*row*, *col*, *val*)

Parameters

row

A **Long** value that provides the row index of the selected element.

col

A **Long** value that provides the column index of the selected element.

val

A **Double** value that provides the value of the selected data element.

Remarks

The **DataSelectEnable** property must be TRUE for the **DataSelect** event to be fired.

For Vector graphs, the **DataSelect** event will not be fired. Instead, the **VDataSelect** event will be fired.

See Also

[DataSelectEnable](#), [VDataSelect](#), [GraphType](#)

RndrPass Event

This event is fired by the control when it has completed redrawing the graph.

Syntax

RndrPass (*rendertime, mode, resolution, param4, param5, updatestate*)

Parameters

rendertime

A **Float** value that gives the clock time (in seconds) that the repainting took.

mode

A **Short** value that describes the type of the rendering action, as follows:

- 0 Normal repaint to the screen
- 1 Rendering to the backbuffer to support selection; no onscreen update
- 2 Accumulation update (not currently supported)
- 3 Rendering pass to the OpenGL Feedback buffer; no onscreen update

resolution

A **Short** value that contains the Resolution value for this rendering pass.

param4

This parameter is not currently used.

param5

This parameter is not currently used.

updatestate

A **Long** value that provides synchronization feedback for the **Update** method. The value can be any one of the following:

- 0 This rendering pass does not take the last **Update** call into account.
- 1 This rendering pass does take the last **Update** call into account but is not the highest resolution level.
- 2 This rendering pass does take the last **Update** call into account and is the highest resolution level.

Remarks

When **ResolutionAutoAdjust** is enabled, one, two, or three **RndrPass** events may be triggered each time the graph is updated or the window is exposed.

For large datasets, the multiple passes allow a quick update at a low level of detail, followed by higher detail passes. The resolution parameter can be used to distinguish between these different updates.

See Also

[ResolutionAutoAdjust](#), [Update](#)

RowColChanged Event

This event is fired by the control when the values of the **Row** or **Col** properties change.

Syntax

RowColChanged (*row*, *col*)

Parameters

row

A **Long** value that provides the new row position.

col

A **Long** value that provides the new column position.

Remarks

The **RowColChanged** event may be fired if the **Row** or **Col** properties are set programmatically. It can also be fired by user interaction with the control, such as using the cursor keys to change the marker position.

See Also

[Row](#), [Col](#)

VDataSelect Event

This event is fired by the control when the graph type is Vector and you double-click on the graph with the left mouse button, or when you hold the mouse pointer steady over a data element in the graph.

Syntax

VDataSelect (*index, xval, yval, zval, wval*)

Parameters

index

A **Long** value that provides the index of the selected vector.

xval

A **Double** value that provides the XComp value of the selected vector.

yval

A **Double** value that provides the YComp value of the selected vector.

zval

A **Double** value that provides the ZComp value of the selected vector.

wval

A **Double** value that provides the WComp value of the selected vector.

Remarks

The **DataSelectEnable** property must be TRUE for the **VDataSelect** event to be fired.

For graph types other than Vector, the **VDataSelect** event will not be fired. Instead, the **DataSelect** event will be fired.

See Also

[DataSelectEnable](#), [DataSelect](#), [GraphType](#), [XComp](#), [YComp](#), [ZComp](#), [WComp](#)

Array Visualizer Grid Control Properties

The following tables list the available Grid control properties:

Grouped by Function:

Array Mapping and Properties

[ColDim](#)

[ShowDimLabels](#)

[RowDim](#)

Data Cell Properties

[CellEditEnabled](#)

[UseDefaultFormat](#)

[FieldWidth](#)

[UseExp](#)

[Precision](#)

[UseHex](#)

File Properties

[FileName](#)

Marker Properties

[Col](#)

[ShowCursor](#)

[Row](#)

Region of Interest

[ColSelMax](#)

[RowSelMax](#)

[ColSelMin](#)

[RowSelMin](#)

Listed Alphabetically (see [functional grouping](#) for links):

C

[CellEditEnabled](#)

[ColSelMax](#)

[Col](#)

[ColSelMin](#)

[ColDim](#)

F

[FieldWidth](#)

[FileName](#)

P

[Precision](#)

R

[Row](#)

[RowSelMax](#)

[RowDim](#)

[RowSelMin](#)

S

[ShowCursor](#)

[ShowDimLabels](#)

U

[UseDefaultFormat](#)

[UseHex](#)

[UseExp](#)

CellEditEnabled Property

Gets or sets the state of enabling cell editing.

Syntax

object.**CellEditEnabled** [= *value*]

Parameters

value

A **Boolean** expression that, when TRUE, enables cell editing and, when FALSE, disables cell editing.

Remarks

When TRUE, double-clicking on a cell turns the cell into edit mode so you can edit the cell. When FALSE, double-clicking does not turn the cell into edit mode.

Col Property

Gets or sets the column index value that will be used (in conjunction with the [Row](#) property) to place the marker.

Syntax

object.**Col** [= *value*]

Parameters

value

A **Long** expression that indicates the new column position.

ColDim Property

Gets or sets the array dimension that will get mapped to the grid's columns.

Syntax

object.**ColDim** [= *dimension*]

Parameters

dimension

A **Long** expression that specifies the dimension to be mapped to the grid's columns.

Remarks

Note: You must call the **Update** method before any change to **ColDim** takes effect.

Both dimension numbers and array indexes begin with index 0.

For arrays of rank 3 or greater, the AvisGrid control can only display grids of a single 2D "slice" of the array. For these higher dimensionality data sets, you can use the **ColDim** and **RowDim** properties to control the direction of this slice and the **SetROI** method to determine which particular slice of the dataset gets mapped to the grid.

See Also

[SetROI](#), [RowDim](#), [Update](#)

ColSelMax Property

Gets or sets a value that determines the maximum column index of the selected region.

Syntax

```
object.ColSelMax [= value]
```

Parameters

value

A **Long** expression containing the maximum column index of the selected region.

Remarks

The properties **ColSelMax**, **ColSelMin**, **RowSelMax**, and **RowSelMin** define the region of data that is currently selected. You can select a cell or a region of cells by holding down the left mouse button and dragging in the cell or across the region.

See Also

[ColSelMin](#), [RowSelMax](#), [RowSelMin](#)

ColSelMin Property

Gets or sets a value that determines the minimum column index of the selected region.

Syntax

```
object.ColSelMin [= value]
```

Parameters

value

A **Long** expression containing the minimum column index of the selected region.

Remarks

The properties **ColSelMax**, **ColSelMin**, **RowSelMax**, and **RowSelMin** define the region of data that is currently selected. You can select a cell or a region of cells by holding down the left mouse button and drag in the cell or across the region.

See Also

[ColSelMax](#), [RowSelMax](#), [RowSelMin](#)

FieldWidth Property

Gets or sets a value that determines the width allocated to display each data value.

Syntax

```
object.FieldWidth [= value]
```

Parameters

value

A **Long** expression containing the field width.

Remarks

FieldWidth is ignored if the **UseDefaultFormat** property is TRUE. It is also ignored if property **UseHex** is TRUE when displaying integer data, or if property **UseExp** is TRUE when displaying floating-point data.

See Also

[UseDefaultFormat](#), [Precision](#), [UseHex](#), [UseExp](#)

FileName Property

Gets or sets the current data file.

Syntax

object.**FileName** [= *file*]

Parameters

file

A **String** expression whose value is a path to the data file.

Remarks

When the **FileName** property points to a valid Array Viewer .AGL file and that file loads, the data in that file replaces any currently loaded data set.

The entire data set in the new file automatically becomes the [region of interest](#). However, if the data set has more than three dimensions, then the data set consists of those array elements whose index is 0 for all dimensions after the third dimension.

Only AGL files can be used as **FileName** properties. The AvisGrid control cannot read ascii, or HDF files.

In the Array Viewer application, when a dataset is saved to the AGL format, attribute data describing the current view state is appended to the file. However when the **FileName** property is set in the AvisGrid control, only the array data values are read by the control. Attribute data (if any) contained in the file is ignored.

You can also set the **FileName** property to a string obtained by **aglGetShareName** or **faglGetShareName**. This allows the AvisGrid control to display array data without saving the array to a file.

See Also

[aglGetShareName](#), [faglGetShareName](#)

Precision Property

Gets or sets a value that determines how many digits to the right of the decimal point are displayed for floating-point data.

Syntax

object.**Precision** [= *value*]

Parameters

value

A **Long** expression containing the precision.

Remarks

Precision is ignored when displaying integer data, or if the **UseDefaultFormat** property is TRUE.

See Also

[UseDefaultFormat](#), [FieldWidth](#)

Row Property

Gets or sets the row index value that will be used (in conjunction with the [Col](#) property) to place the marker.

Syntax

```
object.Row [= value]
```

Parameters

value

A **Long** expression that indicates the new row position.

RowDim Property

Gets or sets the array dimension that will get mapped to the grid's rows.

Syntax

```
object.RowDim [= dimension]
```

Parameters

dimension

A **Long** expression that specifies the dimension to be mapped to the grid's rows.

Remarks

Note: You must call the **Update** method before any change to **RowDim** takes effect.

Both dimension numbers and array indexes begin with index 0.

For arrays of rank 3 or greater, the AvisGrid control can only display grids of a

single 2D "slice" of the array. For these higher dimensionality data sets, you can use the **ColDim** and **RowDim** properties to control the direction of this slice and the **SetROI** method to determine which particular slice of the dataset gets mapped to the grid.

See Also

[SetROI](#), [ColDim](#), [Update](#)

RowSelMax Property

Gets or sets a value that determines the maximum row index of the selected region.

Syntax

```
object.RowSelMax [= value]
```

Parameters

value

A **Long** expression containing the maximum row index of the selected region.

Remarks

The properties **ColSelMax**, **ColSelMin**, **RowSelMax**, and **RowSelMin** define the region of data that is currently selected. You can select a cell or a region of cells by holding down the left mouse button and dragging in the cell or across the region.

See Also

[ColSelMax](#), [ColSelMin](#), [RowSelMin](#)

RowSelMin Property

Gets or sets a value that determines the minimum row index of the selected region.

Syntax

```
object.RowSelMin [= value]
```

Parameters

value

A **Long** expression containing the minimum row index of the selected region.

Remarks

The properties **ColSelMax**, **ColSelMin**, **RowSelMax**, and **RowSelMin** define the region of data that is currently selected. You can select a cell or a region of cells by holding down the left mouse button and dragging in the cell or across the region.

See Also

[ColSelMax](#), [ColSelMin](#), [RowSelMax](#)

ShowCursor Property

Gets or sets a value that determines whether to show the marker position in the grid.

Syntax

```
object.ShowCursor [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, shows the marker position.

Remarks

If **ShowCursor** is TRUE, the cell at the marker position acquires a red border.

See Also

[Row](#), [Col](#)

ShowDimLabels Property

Gets or sets a value that determines whether to display the column and row dimension labels.

Syntax

```
object.ShowDimLabels [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, displays the dimension labels.

UseDefaultFormat Property

Gets or sets a value that determines whether to display data using a default format.

Syntax

```
object.UseDefaultFormat [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, displays data using a default format.

Remarks

You can use the **GetDefaultFormat** method to retrieve the field width and precision information of the default format. If the **UseDefaultFormat** property is TRUE, the **UseHex**, **UseExp**, **FieldWidth**, and **Precision** properties are ignored.

See Also

[GetDefaultFormat](#), [UseHex](#), [UseExp](#), [FieldWidth](#), [Precision](#)

UseExp Property

Gets or sets a value that determines whether to display floating-point data in scientific notation.

Syntax

```
object.UseExp [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, displays floating-point data in scientific notation.

Remarks

UseExp is ignored when displaying integer data, or if the **UseDefaultFormat** property is TRUE.

See Also

[UseDefaultFormat](#)

UseHex Property

Gets or sets a value that determines whether to display integer data in hexadecimal or decimal format.

Syntax

```
object.UseHex [= value]
```

Parameters

value

A **Boolean** expression that, when TRUE, displays integer data in hexadecimal format and, when FALSE, displays the data in decimal format.

Remarks

UseHex is ignored when displaying floating-point data, or if the **UseDefaultFormat** property is TRUE.

See Also

[UseDefaultFormat](#)

Array Visualizer Grid Control Methods

The following table lists the available Grid control methods:

Method

[CopyArrayData](#)

[FormatValue](#)

[GetDefaultFormat](#)

[Reset](#)

[SetDimName](#)

[SetLBound](#)

[SetRoi](#)

[SetRowCol](#)

[Update](#)

CopyArrayData Method

In the Visual Basic language, this method can be used to pass a data array to the control.

Syntax

```
success = object.CopyArrayData (array)
```

Parameters

array

A Visual Basic array that contains the data to be displayed.

Return Value

success

A **Boolean** value indicating success (TRUE) or failure (FALSE).

Remarks

CopyArrayData completely replaces any data that may have been previously passed to the control by the **FileName** property or **CopyArrayData** method.

See Also

[FileName](#)

FormatValue Method

Converts a numeric value to a string using the current display format.

Syntax

```
str = object.FormatValue (value)
```

Parameters

value

A **Double** expression that contains the numeric value.

Return Value

str

A **String** containing the formatted value.

GetDefaultFormat Method

Gets the default field width and precision used for displaying data.

Syntax

```
object.GetDefaultFormat (fieldwidth, precision)
```

Parameters

fieldwidth

A **Long** variable that receives the field width of the default format.

precision

A **Long** variable that receives the precision of the default format.

Return Value

None.

Reset Method

Clears the grid window.

Syntax

```
object.Reset ( )
```

Parameters

None.

Return Value

None.

Remarks

Call **Reset** when you want to erase all things from the grid window.

SetDimName Method

Changes the dimension label that is displayed.

Syntax

object.**SetDimName** (*dim*, *value*)

Parameters

dim

A **Long** expression that contains the dimension number of an array dimension.

value

A **String** expression whose value is a label for the dimension.

Return Value

None.

See Also

[ColDim](#), [RowDim](#)

SetLBound Method

Sets the lower bound value for the specified dimension.

Syntax

object.**SetLBound** (*dim*, *value*)

Parameters

dim

The array dimension (indexed from 0 to Rank - 1) of the lower bound value to be modified.

value

The new lower bound value.

Return Value

None.

Remarks

The lower bound value is the value used to index the first array element for that dimension.

By default, arrays created in the Fortran language have a default lower bound value of 1. In the C language, arrays have a default lower bound value of 0. If desired, you can use the **SetLBound** method to specify a different lower bound value.

Regardless of the value specified in **SetLBound** and the language used to create the array, AvisGrid properties and methods that take an index argument always use zero-based indexing.

SetRoi Method

Sets the position number of the first and last active elements in the [region of interest \(ROI\)](#) for the specified array dimension.

Syntax

success = *object*.**SetRoi** (*dim*, *LB*, *UB*)

Parameters

dim

A **Short** expression that contains the dimension number of an array dimension.

LB

A **Long** expression that contains the value of the desired first element in the ROI.

UB

A **Long** expression that contains the value of the desired last element in

success

A **Boolean** value indicating success (TRUE) or failure (FALSE).

Remarks

You must call the **Update** method before the new ROI is mapped to the grid.

Both dimension numbering and array indexing begin with 0.

For multi-dimensional arrays, use the **RowDim** and **ColDim** properties to control which array dimensions get mapped to the grid's rows and columns.

See Also

[RowDim](#), [ColDim](#), [Update](#)

SetRowCol Method

Sets the row and column position.

Syntax

```
success = object.SetRowCol (row, col)
```

Parameters

row

The new row index. The row should be in the range 0 to $n - 1$, where n is the number of rows in the current array.

col

The new column index. The column should be in the range 0 to $n - 1$, where n is the number of columns in the current array.

Return Value

success

A **Boolean** value indicating success (TRUE) or failure (FALSE).

Update Method

Updates the grid to reflect any changes in either the array data or the [ROI](#) since the last **Update** call.

Syntax

object.**Update** ()

Parameters

None.

Return Value

None.

Remarks

Call **Update** when the data or ROI has been modified in some way, and you want to update the grid to reflect the changes.

Array Visualizer Grid Control Events

The following table lists the available Grid control Events:

Event

[ColDimChanged](#)

[DataChanged](#)

[RowColChanged](#)

[RowDimChanged](#)

ColDimChanged Event

This event is fired by the control when the value of the [ColDim](#) property changes.

Syntax

ColDimChanged (*dim*)

Parameters

dim

A **Long** value that provides the new value of the **ColDim** property.

Remarks

The **ColDimChanged** event may be fired if the **ColDim** property is set programatically. It can also be fired by user interaction with the control, such as by right-clicking on a dimension name to select a different dimension.

See Also

[RowDimChanged](#)

DataChanged Event

This event is fired by the control when a cell value is changed.

Syntax

DataChanged (*row, col, d*)

Parameters

row

A **Long** value that provides the new row position.

col

A **Long** value that provides the new column position.

d

A **VARIANT** that provides the new data value.

Remarks

The **DataChanged** event is fired if a cell value is changed because of cell-editing. If **CellEditEnabled** is FALSE, this event is not fired.

See Also

[CellEditEnabled](#)

RowColChanged Event

This event is fired by the control when the values of the **Row** or **Col** properties change.

Syntax

RowColChanged (*row*, *col*)

Parameters

row

A **Long** value that provides the new row position.

col

A **Long** value that provides the new column position.

Remarks

The **RowColChanged** event may be fired if the **Row** or **Col** properties are set programmatically. It can also be fired by user interaction with the control, such as using the cursor keys to change the marker position.

See Also

[Row](#), [Col](#)

RowDimChanged Event

This event is fired by the control when the value of the [RowDim](#) property changes.

Syntax

RowDimChanged (*dim*)

Parameters

dim

A **Long** value that provides the new value of the **RowDim** property.

Remarks

The **RowDimChanged** event may be fired if the **RowDim** property is set programatically. It can also be fired by user interaction with the control, such as by right-clicking on a dimension name to select a different dimension.

See Also

[ColDimChanged](#)

Error Messages

The following are error messages returned by the Fortran and C routines, along with their associated codes:

Error Name	Error Value
AGL_ERROR_ARRAY_INVALID_RANK	20540
AGL_ERROR_CREATEFILEMAPPING_FAILED	20480
AGL_ERROR_FILE_WRITE_ERROR	20550
AGL_ERROR_FILEMAPPING_ALREADYEXISTS	20490
AGL_ERROR_FILEOPEN_FAILED	20510
AGL_ERROR_INVALID_FILE_FORMAT	20520
AGL_ERROR_MAPVIEWOFFILE_FAILED	20500
AGL_ERROR_MAX_ARRAYEXTENT_ARRAY_EXCEEDED	20520
AGL_ERROR_MAX_ARRAY_NUM_ELEMENTS_EXCEEDED	20530
AGL_ERROR_NO_WRITE_ACCESS	20570
AGL_ERROR_RESHAPE_INVALID	20560