AppendLayoutObject

AppendLayoutObject [***flags***] ***objectType objectName***

The AppendLayoutObject operation appends a single object to the top layout or to the layout specified via the /W flag. It targets the active page or the page specified by the /PAGE flag.

Unlike the AppendToLayout operation, AppendLayoutObject can be used in user-defined functions. Therefore, AppendLayoutObject should be used in new programming instead of AppendToLayout.

Parameters

objectType identifies the type of object to be appended. It is one of the following keywords: `graph`, `table`, `picture`, `gizmo`.

objectName is the name of the graph, table, picture or Gizmo window to be appended.

Use a space between *objectType* and *objectName*. A comma is not allowed.

Flags

/D=*fidelity* Draws layout objects in low fidelity (*fidelity*=0) or high fidelity (*fidelity*=1; default). This affects drawing on the screen only, not exporting or printing. Low fidelity is somewhat faster but less accurate and should be used only for graphs that take a very long time to draw.

/F=*frame* Specifies the type of frame enclosing the object.

- *frame* =1 Single frame (default).
- *frame* =2 Double frame.
- *frame* =3 Triple frame.
- *frame* =4 Shadow frame.

/T=*trans* Sets the transparency of the object background to opaque (*trans* =0; default) or transparent (*trans* =1).

For transparency to be effective, the object itself must also be transparent. Annotations have their own transparent/opaque settings. Graphs are transparent only if their backgrounds are white. PICTs may have been created transparent or opaque. Opaque PICTs cannot be made transparent.

/R=(*l*, *t*, *r*, *b*) Sets the size and position of the object. If omitted, the object is placed with a default size and position. *l*, *t*, *r*, and *b* are the left, top, right, and bottom coordinates of the object, respectively. Coordinates are expressed in units of points, relative to the top/left corner of the paper.

/PAGE=*page* Appends the object to the specified page.

Page numbers start from 1. To target the active page, omit /PAGE or use *page*=0.

The /PAGE flag was added in Igor Pro 7.00.

/W=*winName* Appends the object to the named page layout window. If /W is omitted or if *winName* is `$""`, the top page layout is used.

See Also

NewLayout, **ModifyLayout**, **LayoutPageAction**, **RemoveLayoutObjects**, **TextBox**, **Legend**

AppendMatrixContour

AppendMatrixContour [***axisFlags***][**/F=*formatStr* /W=*winName***] ***zWave***
[**vs** {***xWave, yWave***}]

The AppendMatrixContour operation appends to the target or named graph a contour plot of a matrix of z values with autoscaled contour levels, using the Rainbow color table.

Note: There is no DisplayContour operation. Use `Display; AppendMatrixContour`.

Parameters

zWave must be a matrix (2D wave).

To contour a set of XYZ triplets, use **AppendXYZContour**.

If you provide the *xWave* and *yWave* specification, *xWave* provides X values for the rows, and *yWave* provides Y values for the columns. This results in an "uneven grid" of Z values.

If you omit the *xWave* and *yWave* specification, Igor uses the *zWave*'s X and Y scaled indices as the X and Y values. Igor also uses the *zWave*'s scaled indices if you use * (asterisk symbol) in place of *xWave* or *yWave*.

In a macro, to modify the appearance of contour levels before the contour is calculated and displayed with the default values, append "`;DelayUpdate`" and immediately follow the AppendMatrixContour command with the appropriate **ModifyContour** commands. All but the last ModifyContour command should also have `;DelayUpdate` appended. DelayUpdate is not needed in a function, but DoUpdate is useful in a function to force the contour traces to be built immediately rather than the default behavior of waiting until all functions have completed.

On the command line, the Display command and subsequent AppendMatrixContour commands and any ModifyContour commands can be typed all on one line with semicolons between:

```
Display; AppendMatrixContour MyMatrix; ModifyContour ...
```

Flags

axisFlags	Flags /L, /R, /B, /T are the same as used by **AppendToGraph**.
/F=*formatStr*	Determines the names assigned to the contour level traces. See **Details**.
/W=*winName*	Appends to the named graph window or subwindow. When omitted, action will affect the active window or subwindow. This must be the first flag specified when used in a Proc or Macro or on the command line. When identifying a subwindow with *winName*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.

Details

AppendMatrixContour creates and displays contour level traces. You can modify these all together using the Modify Contour Appearance dialog or individually using the Modify Trace Appearance dialog. In most cases, you will not need to modify the individual traces.

By default, Contour level traces are automatically named with names that show the *zWave* and the contour level, for example, "zWave=1.5". You will see these trace names in the Modify Trace Appearance dialog and in Legends. In most cases, the default trace names will be just fine.

If you want to control the names of the contour level traces (which you might want to do for names in a Legend), use the /F=*formatStr* flag. This flag uses a format string as described for the **printf** operation. The default format string is "`%.17s=%g`", resulting in trace names such as "zWave=1.5". *formatStr* must contain at least `%f` or `%g` (used to insert the contour level) or `%d` (used to insert the zero-based index of the contour level). Include `%s`, to insert the *zWave* name.

Here are some examples of format strings.

formatStr	Examples of Resulting Name	Format
`"%g"`	"100", "1e6", "-2.05e-2"	(<level>)
`"z=%g"`	"z=100", "z=1e6", "z=-2.05e-2"	(z=<level>)
`"%s %f"`	"zWave 100.000000"	(<wave>, space, <level>)
`"[%d]=%g"`	"[0]=100", "[1]=1e6"	([<index>]=<level>)

Examples

```
Make/O/N=(25,25) w2D                    // Make a matrix
SetScale x -1, 1, w2D                   // Set row scaling
SetScale y -1, 1, w2D                   // Set column scaling
w2D = sin(x) * cos(y)                   // Store values in the matrix
Display; AppendMatrixContour w2D
ModifyContour w2D autoLevels={*,*,9}    // Roughly 9 automatic levels
```

See also

Display, **AppendToGraph**, **AppendXYZContour**, **ModifyContour**, **RemoveContour**, **FindContour**.