many megabytes are required to store your entire (selected) MS in memory. If the slider setting is above this, the whole selected MS will fit into the memory buffer. Otherwise, some data planes will be 'greyed out', and the selected data will have to be viewed one buffer at a time, which is somewhat less convenient. The largest process called 'glishtk' contains this buffer memory; it actually contains the entire display library, but the memory buffer can dominate its size.

---

### 1.3.6 viewerdisplaypanel - Tool

**Package display**
**Module viewer**

Tool for providing drawing windows (canvases)
include "viewerdisplaypanel.g"

**Constructors**

| | |
|---|---|
| viewerdisplaypanel | Create a Viewerdisplaypanel `tool` |

**Functions**

| | |
|---|---|
| addgui | Add various GUI components to this Viewerdisplaypanel `tool` |
| adjust | Display adjustment panel for Viewerdisplaydata `tools` |
| animator | Return the Vieweranimator `tool` used by this `tool` |
| annotationdd | Return the agent of the annotation layer |
| annotator | Return the viewerannotations created by this Viewerdisplaypanel `tool` on construction |
| autoregister | Set or query the auto-register state of this Viewerdisplaypanel `tool` |
| canvasmanager | Return the Viewercanvasmanager `tool` used by this `tool` |
| canvasprintmanager | Return the Viewercanvasprintmanager `tool` used by this Viewerdisplaypanel `tool` |
| disablecontrols | Temporarily disable the controls on this Viewerdisplaypanel `tool` |
| done | Close down (destroy) this Viewerdisplaypanel `tool` |
| dying | Is this Viewerdisplaypanel `tool` in the process of destroying itself? |
| enablecontrols | Re-enable the controls on this Viewerdisplaypanel `tool` |
| getoptions | Return a record describing some modifiable state of this `tool` |
| hold | Prevent refreshes occuring on this Viewerdisplaypanel `tool` |
| holdsdata | Returns whether this Viewerdisplaypanel `tool` can hold Viewerdisplaydata `tools` |
| isregistered | Query whether a Viewerdisplaydata `tool` is registered on this Viewerdisplaypanel `tool` |
| leftframe | Return the left frame in the Viewerdisplaypanel `tool` window |
| lowerframe | Return the lower frame in the Viewerdisplaypanel `tool` window |
| newcanvasmanagergui | Create and (probably) show a new graphical user interface for this `tool`'s Viewercanvasmanager `tool` |
| newcanvasprintmanagergui | Create and (probably) show a new graphical user interface for this `tool`'s Viewercanvasprintmanager `tool` |
| newcmap | Query whether this Viewerdisplaypanel `tool` is using a new colormap |
| register | Register a Viewerdisplaydata `tool` for drawing |

| | |
|---|---|
| registercolormap | Explicitly register a colormap on the Viewerdisplaypanel `tool` |
| registerednames | Return a vector of the names of registered Viewerdisplaydata `tools` |
| registrationflags | Return a record describing which Viewerdisplaydata `tools` are registered on this `tool` |
| release | Allow refreshes to occur on this Viewerdisplaypanel `tool` |
| rightframe | Return the right frame in the Viewerdisplaypanel `tool` window |
| setoptions | Set the modifiable state of this `tool` |
| setzoom | Set a specific zoom |
| status | Return a record describing various attributes of the Viewerdisplaypanel `tool` |
| testpattern | Turn a test pattern on or off on the Viewerdisplaypanel `tool` |
| type | Return the type of this `tool` |
| unregister | Unregister a Viewerdisplaydata `tool` from this Viewerdisplaypanel `tool` |
| unregisterall | Unregister all Viewerdisplaydata `tools` from this Viewerdisplaypanel `tool` |
| unregistercolormap | Explicitly unregister a colormap on the Viewerdisplaypanel `tool` |
| unzoom | Unzoom the current view |
| upperframe | Return the upper frame in the Viewerdisplaypanel `tool` window |
| viewer | Return the Viewer `tool` which manages this `tool` |
| writexpm | Write an X11 Pixmap of the Viewerdisplaypanel `tool` to disk |
| zoom | Zoom by a given factor |

### General description

### Description

This `tool` is a fundamental component of the Viewer `tool`. It provides a drawing area (or "canvas") on which Viewerdisplaydata `tools` can draw themselves. Multiple Viewerdisplaypanel `tools` can be placed on the screen, and each can have multiple Viewerdisplaydata `tools` registered to draw themselves. In this way, for example, multiple views of the same data can be simultaneously displayed, and contour Viewerdisplaydata `tools` can be drawn "on top" of raster Viewerdisplaydata `tools`, for example. Exact copies of Viewerdisplaypanel `tools` can be written to X11 Pixmap files, and high-quality PostScript representations of Viewerdisplaypanel `tools` can be saved and printed.

Viewerdisplaypanel `tools` can be adorned with various graphical user interface components, including a "controlbox" which allows the user to select controls like zooming and colormap fiddling, and a "tapedeck" which enables the user to manually or automatically move through data having more than two (non-redundant) axes.

Three points related to the registration of Viewerdisplaydata `tools` on Viewerdisplaypanel `tools` are worth making here:

1. *The order of registration of Viewerdisplaydata* `tools` *on a specific Viewerdisplaypanel* `tool` *is important.* The maximum area of sky, or position-velocity space, or of whatever coordinate system is being used, is determined solely by the first registered Viewerdisplaydata `tool`. All

other Viewerdisplaydata `tools` registered for display on a particular Viewerdisplaypanel `tool` simply draw those parts of their data which ought to be visible on the Viewerdisplaypanel `tool`. A consequence of this is that the zoom setting of the Viewerdisplaypanel `tool` must be reset to the full view whenever the first registered Viewerdisplaydata `tool` is removed, replaced, or re-read. Independent of the order in which Viewerdisplaydata `tools` are registered on a given Viewerdisplaypanel `tool`, the raster-type Viewerdisplaydata `tools` are always drawn first, followed by contours, and lastly axis labels.

2. *Axis labels are drawn by Viewerdisplaydata* `tools`*, and only by the first registered Viewerdisplaydata* `tool`. In the Viewer `tool`, axis labelling information belongs to the Viewerdisplaydata `tools` rather than to the Viewerdisplaypanel `tools`, which is commonly seen in other applications. This is because the only entity which knows *exactly* how to draw the axis labels for a particular piece of data is the Viewerdisplaydata `tool` which draws the data itself. Control over the appearance of axis labels is therefore available via the individual Viewerdisplaydata `tools`.

3. *Registering a specific Viewerdisplaydata* `tool` *on more than one Viewerdisplaypanel* `tool` *can reduce display efficiency.* The drawings made of data by a Viewerdisplaydata `tool` are cached, so that they only need to be recalculated under various circumstances, such as the settings for the Viewerdisplaydata `tool` changing, or the size of the Viewerdisplaypanel `tool` on which they are drawing changing. The caching will in general fail to operate efficiently if a Viewerdisplaydata `tool` is registered on more than Viewerdisplaypanel `tool`. This is best avoided by making another Viewerdisplaydata `tool` from the same dataset, and registering this second `tool` on the second Viewerdisplaypanel `tool`.

Additional to the `tool functions` described for the Viewerdisplaypanel `tool`, this `tool` also possesses the `tool functions` described in the Viewerstandardgui interface documentation. Note that some functions may not be documented and therefore shouldn't be used unless you really know what you are doing.

**Graphical User Interface**

**Description**

The standard graphical user interface provided by the Viewerdisplaypanel `tool` is shown in Figure 1.5. Surrounding the actual display area itself are five GUI components: the menu bar at the top, the button bar at the bottom, the tracking bar[s] just above the buttons, the controlbox on the left, and the (currently disabled) "tapedeck" control on the right. Applications

programmers are able to control which of these components appear on the Viewerdisplaypanel `tools` that they create, and hence throughout AIPS++, Viewerdisplaypanel `tools` can appear slightly different from the example given here.
**Note:** All AIPS++ GUIs have a so-called popup help. Just rest your mouse cursor on top of a GUI element (e.g. a button) and a little frame will appear outlining the funtionality of this element. If you then press and hold down the right mouse button you will get a detailed description. The information given there should be sufficient to let you operate that GUI element.

*The menu bar*

In Figure 1.5, a typical Viewerdisplaypanel `tool` is shown in all its glory. The menu bar along the top has just three menus:

- File, offering selections to create and display new Viewerdisplaypanel `tools`, access to various managers associated with the Viewer `tool`, and selections to finish the Viewerdisplaypanel `tool` itself, and to finish use of the entire Viewer `tool`.

  The File menu is where additional Viewerdisplaypanel `tools` can be created for use. Multiple Viewerdisplaypanels can be on-screen, and to the best of their ability they will share the available color resources. Three classes of Viewerdisplaypanel can be made: *Index* panels, on which normal raster maps can be displayed, with pixel intensities mapped to colors from a colormap; and *RGB* (red-green-blue) and *HSV* (hue-saturation-value) panels, on which multi-component raster maps can be put together. Viewerdisplaydatas which are set up to use a colormap (eg. 'Rainbow', 'Greyscale', etc.) will only draw when registered on an *Index* panel. Similarly, Viewerdisplaydatas which are set up to draw to the red, green or blue color channel will only draw themselves on RGB panels. And likewise, Viewerdisplaydatas which are set up to draw to the hue, saturation or value color channel will only draw when registered on HSV panels.

  For the *RGB* and *HSV* panel types, up to three raster maps can be displayed concurrently, overlaid on each other, with each mapped to a different color component. For example, an optical image could be displayed using the green component, and an X-ray image "overlaid" using the red component. In locations of strong X-ray emission and weak optical emission, the map will be predominantly red; weak X-ray emission coincident with strong optical emission will be predominantly green, and in between will be various levels of yellow.

  Two sub-classes of Viewerdisplaypanels can be made: *non-flashing* and *most colors* panels. The latter provides the most colors available, but with the cost of ugly "color flashing" behaviour as the mouse cursor is moved in to and out of the Viewerdisplaypanel window. The former provides generally less colors, but avoids this color flashing, and is generally more appropriate when analysis involving multiple windows is underway.

The menu also gives access to Viewerslicepanel `tools` in all color modes. Viewerdisplaydata `tool` with at least three axes can be displayed on these in slice mode. See the documentation for Viewerslicepanel `tool` for more.

- DisplayData, which is where various Viewerdisplaydata `tool` can be registered (or unregistered) for display on the Viewerdisplaypanel `tool`. Viewerdisplaydata `tool` are registered for display by ensuring that their checkbox is lit in this menu. When more than one Viewerdisplaydata `tool` is available for registration, additional convenience selections are placed at the bottom of the menu, enabling the user to register or unregister all the Viewerdisplaydata `tool` in the list.

- Tools contains a submenu ImageAnalysis ImageAnalysis tools all listen to regions and postions emitted by the region/positioning tool. The Tools menu also contains a Annotations... menu item, which opens the the Annotations GUI.

  You can interactively make a region (boxes and polygons) from the display with the cursor. First you select the appropriate button (or icon; box or polygon) from the viewer control buttons arranged vertically along the left-hand side of the display. Whatever mouse button you used to select the icon is now active for creating the region. Either drag out a box, or select vertices of a polygon.

  You create the region by double clicking inside of it. When you do this, the Viewerdisplaypanel `tool` will emit an event called `region`. See below for a description of that event.

  – ImageAnalysis This menu brings up a window with roll-ups for several image analysis operations. Each roll-up has a tab widget for each of the registered Viewerdisplaydatas. You can select the one you want to analyze by pressing the button showing the image name.

    ∗ Statistics This roll-up shows image statistics on the selected image. It listens to event created by the region tools (of the control box. Whenever you emit such a region the fields in this GUI will be updated.
      When you generate an interactive region the statistics are computed only for the current displayed plane of the image. You can extend this spatial region over a range of animation-axis pixels via the Extend check box and by entering a pixel range in the adjacent entry box. For example, entering 10 20 would apply the spatial region to planes 10 through 20 along the animation axis.

Statistics can also be computed for the full data Full or the current plane Plane as selected with the animator tapedeck. Check the Robust button to also retrieve robust statistics. You can also bring up a pgplotter tool to show a statistic histogram by clicking the plot button or alternatively Auto-plot to have that plot updated automatically. The next row of buttons allow you to copy the values shown to the clip board (retrieve with `x := dcb.paste()`). Auto-copy automatically updates the clip board.

* Positions The Position rollup consists of a plot button (only if the image has more than 2 dimensions), a copy button, an autocopy switch and position fields to identify the current position of the cursor.
  The position rollup listen's to position events generated by it's parent Viewerdisplaypanel `tool` (via the crosshair tool) as well as motion events generated by the Viewerdisplaypanel `tool` from the movement of the mouse over the display area (when position tracking is turned on). When one of these events are received by the position rollup, the current position details are updated.
  You can plot the profile (at the cursor position and along the movie axis) with the button labelled `Plot`.
  When you submit a `region event` (see below), the plot will be updated with a new profile. This profile respresents the statistics in the submitted region along the movie axis of the image. The default statistic is `mean` and can be changed in via the `Adjust` button near the bottom of the profile window.
  You can copy the position details to the default Clipboard `tool` (**dcb**) with the `Copy` button. This copies a record to the clipboard with the format below. If you prefer, you can autocopy the positions to the Clipboard by checking the `Autocopy` button. Then, every time you generate a position, a record is automatically copied the Clipboard.
  Format of copied record:
  rec.ddname
  rec.position .pixel .world .numeric .string .intensity .value .mask
  rec.region .pixelblc .pixeltrc .stat
  rec.profile .cs .cs .x .pixel .abs .world .abs .y .data .mask .unit
  The profile can be customized via the `Adjust` button near the bottom of the profile window. Features such as autoscaling, rest frequency labelling and region profiles are available here. You can zoom in the profile window by using the left mouse button using the same method that is used to zoom in the viewer panel. Similarly, an `Unzoom` button is also available.

* Regions

Every time you generate a region, it is also inserted into the region entry widget (a check box flashes to indicate something went in) at the top of the regions rollup (roll it down to see the entry widget !). This enables you to copy it to the default Clipboard `tool` (**dcb**) and also paste into it from the default Clipboard `tool`. When a region is captured, a little indicator light next to the entry widget flashes to indicate something went in. There is also a `Show` button which prints the bounding box of the region to the logger and shows the regions already generated as a white outline. This button turns into `Hide` once it is pressed to disable the region drawing.

It is also possible to accumulate regions, and then make a compound region from all of them. This is done via the `Regions` rollup; roll it down and look at the row of buttons at the bottom. In this mode, if you have pressed the `Show` button, the regions are shown individually as they are generated.

The second button from the left controls the accumulation. If it is labelled `Start`, when you press it (and its label will then change to `Finish`), regions are internally accumulated (no `region` event is emitted and the region is not inserted in the region entry widget). When you have accumulated all the regions you want, you may press `Finish`. When accumulation mode is active, a check box to the right is lit and the number of regions accumulated displayed. Now a compound region is made from all the regions you accumulated according to the select you make from the left-most menu; you can make a union or intersection. When this region is created, the `region` event is emitted.

* Slices Every time you use the polyline tool, a `polyline` event is generated containing the points of the polyline. These points are used here to
  - Generate an interpolated 1-D slice along the polyline and optionally plot it (use the `Plot` and `Autoplot` buttons).
  - For DirectionCoordinates, the distance and position angle between the polyline vertices is computed. If the polyline contains more than two points the sum over all distances and the sum over all angles is shown.

  Use the `Copy` and `Autocopy` buttons to transfer the slice, distance and position angle information to the clip board.
* Summaries This part just shows the same Image.summary would show, wrapped up in a nice gui text area.

- Help, which provides selections with which assistance can be obtained on the Viewer `tool` itself, or more widely within AIPS++.

*The button bar* Along the bottom of the Viewerdisplaypanel `tool` are a few buttons, which are available for commonly executed tasks:

- The Adjust... button is used to make adjustments to the way the various Viewerdisplaydata `tool` draw themselves. Pressing Adjust just once will create adjustment windows for all Viewerdisplaydata `tool` that are presently registered on the Viewerdisplaypanel `tool`. Here is a detailed description of the elements of the Adjust GUI.
- The Unzoom button is a short-cut to fully unzooming the display—when pressed, the Viewerdisplaypanel `tool` will re-draw itself displaying as much of the *first-registered* Viewerdisplaydata `tool` as is possible.
- The Clear button is a short-cut to unregistering all Viewerdisplaydata `tool` which are presently registered on this Viewerdisplaypanel `tool`. When pressed, the Viewerdisplaypanel `tool` will be cleared.
- The Print... button is a short-cut to selecting Print... from the File menu, and will bring up the print window for this Viewerdisplaypanel `tool`, which naturally provides printing operations. For more go to Viewercanvasprintmanager `tool`.
- Finally, the Done button will remove this Viewerdisplaypanel `tool` from the viewer.

*The tracking bar[s]*

Separate tracking bars are generated for each Viewerdisplaydata `tool` registered on the Viewerdisplaypanel `tool`. They continually report the value and position of the point of the display area under the mouse cursor. Pressing `space bar` in the display area will "freeze" the tracking displays or restore them again to continual tracking. Pressing the `L` key will send the displayed values to the logger. (An even more detailed record may also be sent to the clipboard, by using the Positions rollup under the Tools/ImageAnalysis menu item).
Note that if you have "click-to-focus" set in your windowing system, you may need to click on the displaypanel's window before it will recognize the key presses above. This can be confusing, since the tracking bar[s] will respond to mouse movement in any case.

*The control box*

The control box (on the left hand side of most Viewerdisplaypanel `tools`) provides a more convenient way to access the various controls available on Viewerdisplaypanel `tools`. Clicking on a button will assign that control to the mouse button which was used to click, over-riding any previous assignment to that mouse button.
For most tools the following actions apply:

- "Esc" key cancels the tool.

- "Double click" of the button assigned to the tool executes it (eg. with regions it sends out an event).
- "Handles" (little boxes on the edges) can be used to resize the tool.

From the top button downwards, the controls are:

- **Zooming:** The top-most button, depicting a magnifying glass with a plus sign inside, is the zoom control button. Zooming is accomplished by pressing down the selected mouse button at the start point, dragging the mouse away from that point, and releasing the selected mouse button when the zoom box encloses the desired zoom area. Once the button is released, the zoom rectangle can be moved by clicking inside it with the selected mouse button and dragging it around, and resized by selecting one of the handles at the corners and dragging.

  To zoom in, simply double click with the selected button inside the rectangle, and when the Viewerdisplaypanel `tool` is re-drawn, the area enclosed by the rectangle will be expanded to fill the Viewerdisplaypanel `tool`, in as much as other constraints allow this. Double clicking outside the rectangle, will result in a zoom out, whereby the previous full view will be scaled down to fit within the rectangle, in as much as other constraints allow this.

- **Panning:** The next button down, whose icon is the outline of a hand, is the panning control button. Panning is accomplished by pressing down the selected mouse button at the point you wish to move, dragging the mouse to the position where you want the first point moved to, and releasing the selected mouse button.

  **Note:** The arrow keys, Page Up, Page Down, Home and End keys, and scroll wheel (if any) can also be used to scroll through your data once you have zoomed in. For these to work, the mouse must be over the display panel drawing area, but no mouse tool need be active. All but the left and right arrows apply only to the vertical direction. However, you can use the Crtl key in conjunction with the arrows to move one page at a time in any direction.

- **Stretch-shift colormap fiddling:**

- **Brightness-contrast colormap fiddling:**

- **Positioning:** The fifth control is a crosshair icon, which enables the user to place a crosshair marker on the Viewerdisplaypanel to indicate a position. Depending on the context, the emitted positions may be used, for example, to display image spectral profiles or flag measurement set data. After selecting a mouse button for the crosshair, click on the Viewerdisplaypanel to place it, and drag to move it around. Double-click is not needed with this control; the position is emitted whenever the crosshair's assigned mouse button is pressed or the crosshair is moved. **Note: position selection is quite different from *position tracking*.**

- **Rectangle and Polygon region drawing:** Rectangular and polygonal regions can easily be constructed. The rectangle region is generated exactly the same way as the zoom rectangle, and is emitted by double clicking within the rectangle. Polygon regions can be constructed by progressively clicking the selected mouse button at the desired location of each vertex, and clicking in the same location twice to complete the polygon. Once constructed, it can be moved by dragging inside the polygon, and reshaped by dragging the various handles at the vertices. Double click within the polygon to emit the polygonal region. Compound image regions can be built up using these controls. The rectangular control can also be used to specify areas to be flagged in measurement sets.
- **Polyline drawing:** A polyline can be constructed with this tool. This tool is almost identical to the polygon region tool. Create the points by clicking at the positions wanted and the double-click to finish the line. this generates a polyline event.

*The animation tapedeck*

In its 'Normal' mode, the tapedeck-style controls down the right-hand side of most Display Panels provide basic movement between image planes, along a selected third dimension of the displaydatas. This set of buttons will only be enabled when the first-registered displaydata reports that it has more than one plane along the 'Z axis' specified in its 'Adjust' panel. In the most common case the animator controls the frequency channel being viewed. From the top down, the tape deck controls allow the user to:

- rewind to the start of the sequence (i.e. the first plane)
- step backwards by one plane
- play backwards, or repetitively step backwards
- stop any current play
- play forwards, or repetitively step forwards
- step forwards by one plane
- fast forward to the end of the sequence

Below the tapedack buttons is an editable text box indicating the current frame number and a sunken label showing the total number of frames. One can type a channel number into the current frame box to jump to that channel, for example, when 'Frequency' is on the 'Z axis'.
Below these is a right-arrow button; pressing it will display a popup frame containing sliders for setting the animation speed and the number of frames to skip when playing animations.
Below all these controls is a pair of radio buttons selecting animator mode: either 'Normal' (the default mode described above) or 'Blink'. In 'Blink' mode, when

more than one raster image is registered on the Display Panel, the tapedeck will control which one is being displayed at the moment. The images registered should cover the same portion of the sky, using the same coordinate projection. Several images can also be displayed side-by-side in this mode, with the same zoom applied to all. For details on operating the Display Panel in this mode, see the Display section of the *AIPS++ Cookbook, vol. 1*.

### Events

*The Viewerdisplaypanel* `tool` *emits several events. A few of them can be of interest for the programmer.*

- pseudoregion *The pseudoregion event is emitted after double-clicking a region tool. The record this event holds contains unfromatted information about the region. Use* region *instead.*
- pseudoposition *The pseudoposition event is emitted after double-clicking a positioning tool. The record this event holds contains unfromatted information about the positon. Use* position *instead.*
- motion *The motion event returns a record containing (screen)pixel and world coordinate (and units) of the position the cursor is moved. It is for internal use only.*
- region *Note: This event only occurs when Viewerdisplaydata* `tools` *are registered.*

  *An event called* `region` *is emitted when you interactively create a region with the cursor. This event is always emitted, unless region accumulation mode is active (see above). When you terminate accumulation mode, the* `region` *event is then emitted.*

  *The region event $value record is shown diagramatically below.*

  ```
  \$value
    region               # The region tool
    type                 # The region type (box, polygon etc)
  ```

- position *Note : This event only occurs when Viewerdisplaydata* `tools` *are registered.*

  *An event called* `position` *is emitted when you interactively created a position with the cursor (see above).*

  *The $value of the event is a record with the fields* `position` *and* `profile`.

  *The position event record is shown diagramatically below.*

  ```
  \$value
    position             # Location of cursor
  ```

```
    pixel               #   Absolute integer pixel coordinate
    world               #   Absolute world coordinate
      numeric           #     as a numeric vector
      quantity          #     as vector of quantities
      string            #     as a formatted string
      measure           #     as measures (further subfields not shown)
    intensity           #   Intensity of image
      mask              #     mask value
      value             #     value of image (quanity)
      pixel             #     absolute integer pixel location
  profile               # The z-profile (if > 2D image)
    x                   #   abcissa
      pixel             #     pixel coordinate
        abs             #       absolute
        rel             #       relative
      nativeworld       #     world coordinate in native units
        abs             #       absolute
        rel             #       relative
      name              #     name of abcissa axis
      nativeunit        #     native abcissa unit
      currentworld      #     world coordinate as currently plotted (if plotted
)
      currentunit       #     abcissa unit as currently plotted
      currentveltype    #     velocity type as currently plotted
      currentabs        #     absolute or relative as currently plotted
      isSpectral        #     is the abcissa a spectral coordinate
    y                   #   ordinate
      unit              #     unit of ordinate
      data              #     ordinate values
```

- polyline *This event is generated by the* `polyline` *tool, when the user double-clicks, or drags and drops an existing polyline or its points.*
- statistics *Note: This event only occurs when Viewerdisplaydata* `tools` *are registered.*

  *If the statistics rollup is rolled down, then statistics are written to the listboxes (*`Plane`, `Full` *button press or region event emitted) then a* `statistics` *event is generated.*

  *The statistics event record is shown diagramatically below. The* `stats` *subfield is a record with the same format as that generated by the return record of function Image.statistics.*

```
\$value
  region          # The region from which the statistics were evaluated
  stats           # The statistics record
```

```
npts
sum
sumsq
min
max
mean
sigma
rms
blc
blcf
trc
trcf
minpos
minposf
maxpos
maxposf
```

### viewerdisplaypanel - Constructor

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Create a Viewerdisplaypanel* `tool`

**Synopsis**

*viewerdisplaypanel(parent, viewer, width, height, nx, nx, maptype, newcmap, mincolors, maxcolors, autoregister, holdsdata, show, widgetset)*

**Arguments**

| | |
|---|---|
| *parent* | *parent frame for the panel*<br>*Allowed: agent or F*<br>*Default: F* |
| *viewer* | *the Viewer* `tool` *which will manager this* `tool`<br>*Allowed: Viewer* `tool` |
| *width* | *width in pixels of the drawing area*<br>*Allowed: integer*<br>*Default: 250* |
| *height* | *height in pixels of the drawing area*<br>*Allowed: integer*<br>*Default: 250* |
| *nx* | *number of panels in x*<br>*Allowed: integer*<br>*Default: 1* |
| *nx* | *number of panels in y*<br>*Allowed: integer*<br>*Default: 1* |
| *maptype* | *how colors will be addressed on this Viewerdisplaypanel* `tool`<br>*Allowed: 'index' 'rgb' 'hsv'*<br>*Default: 'index'* |
| *newcmap* | *install a private colormap? (ignored if a parent frame is provided)*<br>*Allowed: boolean*<br>*Default: F* |
| *mincolors* | *minimum number of colors to allocate*<br>*Allowed: Integer or unset*<br>*Default: unset* |
| *maxcolors* | *maximum number of colors to allocate*<br>*Allowed: Integer or unset*<br>*Default: unset* |
| *autoregister* | *accept auto-register data on this Viewerdisplaypanel* `tool`*?*<br>*Allowed: boolean*<br>*Default: F* |
| *holdsdata* | *is this Viewerdisplaypanel* `tool` *going to hold data?*<br>*Allowed: boolean*<br>*Default: T* |
| *show* | *show the Viewerdisplaypanel* `tool`*?*<br>*Allowed: boolean*<br>*Default: T* |
| *widgetset* | *Widgetserver* `tool` *to use to build the GUI*<br>*Allowed: Widgetserver* `tool` *or unset*<br>*Default: unset* |

**Returns**

*Viewerdisplaypanel* `tool` *or fail*

**Description**

*This is the only constructor for Viewerdisplaypanel* `tools`*. It constructs (creates) a Viewerdisplaypanel* `tool` *which can subsequently be used for the display of Viewerdisplaydata* `tools`*. If* `widgetset` *is left* `unset`*, then the Viewerdisplaypanel* `tool` *will use the Widgetserver* `tool` *used by its parent Viewer* `tool`*.*
**This constructor should not be called by users:** *it is called by the newdisplaypanel function of the Viewer* `tool`*, which returns a Viewerdisplaypanel* `tool`*.*

---

**viewerdisplaypanel.addgui - Function**

| | |
|---:|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Add various GUI components to this Viewerdisplaypanel* `tool`

**Synopsis**
*addgui(guihasmenubar, guihascontrolbox, guihasanimator, guihasbuttons, hasdismiss, hasdone, isolationmode)*

**Arguments**

| | |
|---|---|
| *guihasmenubar* | *add a menu bar (top)?*<br>*Allowed: boolean*<br>*Default: T* |
| *guihascontrolbox* | *add a control box (left)?*<br>*Allowed: boolean*<br>*Default: T* |
| *guihasanimator* | *add an animator/tapedeck (right)?*<br>*Allowed: boolean*<br>*Default: T* |
| *guihasbuttons* | *add buttons (bottom)?*<br>*Allowed: boolean*<br>*Default: T* |
| *hasdismiss* | *add a "Dismiss" button to the GUI?*<br>*Allowed: unset or boolean*<br>*Default: unset* |
| *hasdone* | *add a "Done" button to the GUI?*<br>*Allowed: unset or boolean*<br>*Default: unset* |
| *isolationmode* | *prevent user accessing other components of this Viewer* `tool`*?*<br>*Allowed: boolean*<br>*Default: F* |

**Returns**
*T*

**Description**

*This function can be used to add specified graphical user interface components to an existing Viewerdisplaypanel* `tool`*. This function can only be called once per Viewerdisplaypanel* `tool`*, so get it right the first time!*
*If both* `hasdismiss` *and* `hasdone` *are given as* `T`*, then a warning will be emitted, and* `hasdismiss` *will be forcibly set to* `F`*. If both* `hasdismiss` *and* `hasdone` *are left* `unset`*, then* `hasdone` *will be forcibly set to* `T` *and* `hasdismiss` *to* `F`*. If only one of these two arguments is specified, and the other left* `unset`*, then the* `unset` *argument will default to the opposite of the given argument.*
*If* `isolationmode` *is* `T`*, then the GUI added to the Viewerdisplaypanel* `tool`*will not provide the user with the buttons or menu items to create new Viewerdisplaypanel* `tools` *or Viewerdatamanager* `tools`*. Typically this option should be used whenever a non-global Viewer* `tool` *is constructed.*
*See the GUI description on hpw to operate the GUI.*

---

**viewerdisplaypanel.adjust - Function**

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Display adjustment panel for Viewerdisplaydata* `tools`

**Synopsis**
*adjust(displaydata)*

**Arguments**

| | |
|---|---|
| *displaydata* | *name of displaydata*<br>*Allowed: string*<br>*Default: F* |

**Returns**
*T*

**Description**

*In the default mode this function creates and shows a graphical user interface for each Viewerdisplaydata* `tool` *which is registered for display on this Viewerdisplaypanel* `tool`*. If the name of a Viewerdisplaydata* `tool` *is given only it's graphical user interface will be displayed. The nice thing about creating the adjustment panels in this way is that they can be instantly remapped to the screen even if the user has hit the "Dismiss" button. Generally, other methods of obtaining the Viewerdisplaydata* `tool` *GUIs result in GUIs which have "Done" buttons.*

---

### viewerdisplaypanel.animator - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the Vieweranimator* `tool` *used by this* `tool`

**Synopsis**
*animator()*

**Returns**
*Vieweranimator* `tool`

**Description**

*This function simply returns the Vieweranimator* `tool` *which is used by this Viewerdisplaypanel* `tool`*. The return value can be used to manually and automatically step through the frames available to the Viewerdisplaypanel* `tool`*.*

---

### viewerdisplaypanel.annotationdd - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the agent of the annotation layer*

**Synopsis**
*annotationdd()*

**Returns**
*drawingdisplaydata agent*

**Description**
*This function retruns an DDD agent which can be used to put annotations onto the canvas.*

---

### viewerdisplaypanel.autoregister - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Set or query the auto-register state of this Viewerdisplaypanel* `tool`

**Synopsis**
*autoregister(set)*

#### Arguments

| | |
|---|---|
| *set* | *Auto-register on or off*<br>*Allowed: boolean or unset*<br>*Default: unset* |

#### Returns
*boolean*

#### Description

*This function, if called with a boolean value for* `set`*, will set the auto-register state of this Viewerdisplaypanel* `tool` *to the given value, and return the same value. Otherwise, it simply returns a boolean indicating whether auto-register is on (T) or off (F).*
*Any Viewerdisplaydata* `tools` *subsequently created via the loaddata function of the Viewer* `tool`*, with* `autoregister=T`*, will automatically be registered on this Viewerdisplaypanel* `tool` *if auto-register is true for this Viewerdisplaypanel* `tool`.

---

### viewerdisplaypanel.canvasmanager - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the Viewercanvasmanager* `tool` *used by this* `tool`

#### Synopsis
*canvasmanager()*

#### Returns
*Viewercanvasmanager* `tool`

#### Description

*This function simply returns the Viewercanvasmanager* `tool` *which was constructed and is used by this Viewerdisplaypanel* `tool`*. The return value provides functions to modify various settings of the Viewerdisplaypanel* `tool`.

---

### viewerdisplaypanel.canvasprintmanager - Function

**Package** **display**
**Module** **viewer**
**Tool** **viewerdisplaypanel**

*Return the Viewercanvasprintmanager* `tool` *used by this Viewerdisplaypanel* `tool`

**Synopsis**
*canvasprintmanager()*

**Returns**
*Viewercanvasprintmanager* `tool`

**Description**

*This function returns the Viewercanvasprintmanager* `tool` *that was constructed and is used by this Viewerdisplaypanel* `tool` *to provide printing services. The return value provides functions which can be used to control printing options.*

---

### viewerdisplaypanel.viewer - Function

**Package** **display**
**Module** **viewer**
**Tool** **viewerdisplaypanel**

*Return the Viewer* `tool` *which manages this* `tool`

**Synopsis**
*viewer()*

**Returns**
*Viewer* `tool`

**Description**

*This function simply returns the Viewer* `tool` *which manages this Viewerdisplaypanel* `tool`*, and probably constructed it.*

---

### viewerdisplaypanel.dying - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Is this Viewerdisplaypanel* `tool` *in the process of destroying itself?*

**Synopsis**
*dying()*

**Returns**
*boolean*

**Description**

*This function is provided so that tools created by (and belonging to) the Viewerdisplaypanel* `tool` *can establish whether a request to destroy them came from the Viewerdisplaypanel* `tool`*itself. There are some tools, for example the Viewercanvasmanager* `tool`*, which will only allow their owning Viewerdisplaypanel* `tool` *to call their* `done` *function.*

---

### viewerdisplaypanel.done - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Close down (destroy) this Viewerdisplaypanel* `tool`

**Synopsis**
*done()*

**Returns**
*T*

**Description**

*If you no longer need to use this Viewerdisplaypanel* `tool`*, calling this function will free up its resources. Any Viewerdisplaydata* `tools` *which are registered on this Viewerdisplaypanel* `tool` *will be unregistered. Any graphical user interfacesassociated with this* `tool` *will be removed from the screen. An event with name* `'done'` *will be emitted when the Viewerdisplaypanel* `tool` *is destroyed.*

---

### viewerdisplaypanel.upperframe - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the upper frame in the Viewerdisplaypanel* `tool` *window*

**Synopsis**
*upperframe()*

**Returns**
*frame*

**Description**

*This function is generally used internally to obtain the frame into which the Viewerdisplaypanel* `tool` *menubar should be placed.*

---

### viewerdisplaypanel.lowerframe - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the lower frame in the Viewerdisplaypanel* `tool` *window*

**Synopsis**
*lowerframe()*

**Returns**
*frame*

### Description

*This function is generally used internally to obtain the frame into which the Viewerdisplaypanel* `tool` *buttonbar should be placed.*

---

### viewerdisplaypanel.leftframe - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the left frame in the Viewerdisplaypanel* `tool` *window*

**Synopsis**
*leftframe()*

**Returns**
*frame*

### Description

*This function is generally used internally to obtain the frame into which the Viewerdisplaypanel* `tool` *controlbox should be placed.*

---

### viewerdisplaypanel.rightframe - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the right frame in the Viewerdisplaypanel* `tool` *window*

**Synopsis**
*rightframe()*

**Returns**
*frame*

### Description

*This function is generally used internally to obtain the frame into which the Viewerdisplaypanel* `tool` *Vieweranimator* `tool` *tapedeck should be placed.*

---

### viewerdisplaypanel.writexpm - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Write an X11 Pixmap of the Viewerdisplaypanel* `tool` *to disk*

### Synopsis
*writexpm(filename)*

### Arguments

| | |
|---|---|
| *filename* | *name of X11 Pixmap file*<br>*Allowed: string* |

### Returns
*T*

### Description

*This function writes an exact replica of the display area part of the Viewerdisplaypanel* `tool` *to disk in X11 Pixmap format. This format can be converted to "portable pixmap format" via the utility* **xpmtoppm**, *and then onwards to many other pixmap formats, for example TIFF, BMP, etc.*

---

### viewerdisplaypanel.annotator - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the viewerannotations created by this Viewerdisplaypanel* `tool` *on construction*

**Synopsis**
*annotator()*

**Returns**
*This Viewerdisplaypanel* `tools` *Viewerannotations* `tool`*tool*

**Description**

*This function returns a reference to this Viewerdisplaypanel* `tools` *Viewerannotations* `tool`*tool.This can be used to draw shapes onto the viewerdisplaypanel, open the annotations GUI, delete shapes etc. For more information see viewerannotations*

---

### viewerdisplaypanel.newcmap - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Query whether this Viewerdisplaypanel* `tool` *is using a new colormap*

**Synopsis**
*newcmap()*

**Returns**
*boolean or unset*

**Description**

*This function returns* `T` *if the Viewerdisplaypanel* `tool` *was constructed using a new (private) X Windows colormap,* `F` *if the Viewerdisplaypanel* `tool` *is using the system X Windows colormap, or* `unset` *if this information is unknown. It will be unknown in cases where the Viewerdisplaypanel* `tool` *was inserted into an existing frame, in which case the Viewerdisplaypanel* `tool` *will use the same X Windows colormap as the parent frame. (Recall that the constructor argument* `newcmap` *is ignored if a parent frame is provided.)*

---

### viewerdisplaypanel.holdsdata - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Returns whether this Viewerdisplaypanel* `tool` *can hold Viewerdisplaydata* `tools`

**Synopsis**
*holdsdata()*

**Returns**
*boolean*

**Description**

*This function simply returns a boolean value indicating whether this Viewerdisplaypanel* `tool` *was constructed with the* `holdsdata` *argument T or F. Viewerdisplaypanel* `tools` *for which* `holdsdata=T` *will have functions for registering and unregistering Viewerdisplaydata* `tools`*. The only use of Viewerdisplaypanel* `tools` without `holdsdata=T` *is for the Viewercolormapmanager* `tool` *graphical user interface as of this writing.*

---

### viewerdisplaypanel.status - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return a record describing various attributes of the Viewerdisplaypanel* `tool`

**Synopsis**
*status()*

**Returns**
*record*

**Description**

*This function returns some fairly low-level information about the Viewerdisplaypanel* `tool`*. The record returned has upto two top-level fields:* `pixelcanvas`*, and* `paneldisplay` *if this Viewerdisplaypanel* `tool` *holds data. Fields which may exist in the* `pixelcanvas` *sub-record include the following (those tagged "dynamic" can change during the lifetime of a Viewerdisplaypanel* `tool`*):*

- `width`*: width of the entire display area ("pixelcanvas") in pixels (dynamic).*
- `height`*: height of the pixelcanvas in pixels (dynamic).*
- `depth`*: bit depth of the pixelcanvas, normally 8-bit.*
- `maptype`*: maptype of the pixelcanvas, normally 'index'.*
- `xdpi`*: approximate resolution in the X direction, in dots per inch (dpi).*
- `ydpi`*: approximate resolution in the Y direction.*
- `colortablesize`*: number of colors currently allocated to the pixelcanvas (dynamic).*
- `maxcolortablesize`*: maximum number of colors that could immediately be allocated to the pixelcanvas (dynamic).*
- `buildincolormaps`*: names of the colormaps that are known to the pixelcanvas.*
- `registeredcolormaps`*: a record having sub-fields* `names`*, which lists the names of the registered colormaps, and* `sizes`*, which lists the length of each of the named, registered colormaps. Normally the values in* `sizes` *should add up to* `colortablesize`*, since the colormaps will dynamically resize to fill the colortable.*

*Fields which may exist in the* `paneldisplay` *sub-record (they are all dynamic) include:*

- `nxpanels`*: The number of panels in x direction on this paneldisplay.*
- `nypanels`*: The number of panels in y direction on this paneldisplay.*
- `xorigin`*: The blcX of this paneldisplay on the pixelcanvas (0.0-1.0).*
- `yorigin`*: The blcY of this paneldisplay on the pixelcanvas (0.0-1.0).*
- `xsize`*: The extent in x of this paneldisplay on the pixelcanvas (0.0-1.0).*
- `xsize`*: The extent in y of this paneldisplay on the pixelcanvas (0.0-1.0).*
- `linearblc`*: linear coordinates* `[x, y]` *of the bottom-left corner of each panel in the paneldisplay.*

- `lineartrc`*: linear coordinates* `[x, y]` *of the top-right corner of each panel in the paneldisplay.*
- `worldblc`*: world coordinates* `[x, y]` *of the bottom-left corner of each panel in the paneldisplay.*
- `worldtrc`*: world coordinates* `[x, y]` *of the top-right corner of each panel in the paneldisplay.*
- *axisunits: the units of the world axes*

---

**viewerdisplaypanel.getoptions - Function**

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return a record describing some modifiable state of this* `tool`

**Synopsis**
*getoptions()*

**Returns**
*record*

**Description**

*This function returns a record which describes some of the internal state of this Viewerdisplaypanel* `tool`*. The record differs from that returned by the* `status tool function` *in both format and content: it is suitable for construction of an Autogui* `tool`*, and contains explicitly modifiable parameters. A brief description of the fields which* may *exist in the returned record follows:*

- `colortablesize`*: the number of colors currently allocated to this Viewerdisplaypanel* `tools` *pixelcanvas.*
- `papercolors`*: a boolean specifying whether screen or paper colors (black/white or white/black background/foreground) should be used.*
- `leftmarginspacepg`*: left margin space in PgPlot character units.*
- `bottommarginspacepg`*: bottom margin space in PgPlot character units.*

- `rightmarginspacepg`: *right margin space in PgPlot character units.*
- `topmarginspacepg`: *top margin space in PgPlot character units.*
- `nxpanels`: *The number of panels in x direction on this paneldisplay.*
- `nypanels`: *The number of panels in y direction on this paneldisplay.*
- `xspacing`: *The space between adjacent panels in x direction.*
- `yspacing`: *The space between adjacent panels in y direction.*

---

### viewerdisplaypanel.setoptions - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Set the modifiable state of this* `tool`

**Synopsis**
*setoptions(options)*

**Arguments**

| | |
|---|---|
| *options* | *options record describing the requested state*<br>*Allowed: record* |

**Returns**
*T*
*This function should be used to set the internal state of a Viewerdisplaypanel* `tool`*, according to the field names of the record returned by the getoptions* `tool function`*. Installing new options on a Viewerdisplaypanel* `tool` *will generally cause the refresh (that is, a total re-drawing) of any Viewerdisplaydata* `tools` *registered on the Viewerdisplaypanel* `tool`*. This is because modifying the options will cause a change in the appearance of the data.*

---

### viewerdisplaypanel.hold - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Prevent refreshes occuring on this Viewerdisplaypanel* `tool`

**Synopsis**
*hold()*

**Returns**
*T*

**Description**

*This function will prevent refreshing occuring on this Viewerdisplaypanel* `tool` *until an equal number of calls to the release function have been made. Typically this should be used when multiple operations need to be carried out before the user is shown the result of the operations.*

---

### viewerdisplaypanel.release - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Allow refreshes to occur on this Viewerdisplaypanel* `tool`

**Synopsis**
*release()*

**Returns**
*T*

**Description**

*This function will re-enable refreshing occuring on this Viewerdisplaypanel* `tool` *provided an equal number of calls have been made to the hold function.*

---

### viewerdisplaypanel.disablecontrols - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Temporarily disable the controls on this Viewerdisplaypanel* `tool`

**Synopsis**
*disablecontrols()*

**Returns**
*T*

**Description**

*This function will simply suspend the actions of controls on this Viewerdisplaypanel* `tool`*. This means, for example, that the zoom and colormap fiddling controls will not work temporarily.*

---

### viewerdisplaypanel.enablecontrols - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Re-enable the controls on this Viewerdisplaypanel* `tool`

**Synopsis**
*enablecontrols()*

**Returns**
*T*

**Description**

*This function re-installs the controls on this Viewerdisplaypanel* `tool` *after they have been disabled with the disablecontrols function.*

---

### viewerdisplaypanel.newcanvasmanagergui - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Create and (probably) show a new graphical user interface for this* `tool`*'s Viewercanvasmanager* `tool`

**Synopsis**

*newcanvasmanagergui(parent, show, hasdismiss, hasdone, widgetset)*

**Arguments**

| | |
|---|---|
| *parent* | *parent frame in which to place the Viewercanvasmanager* `tool`<br>*Allowed: agent or F*<br>*Default: F* |
| *show* | *show the Viewercanvasmanager* `tool`*?*<br>*Allowed: boolean*<br>*Default: T* |
| *hasdismiss* | *add a "Dismiss" button to the GUI?*<br>*Allowed: boolean*<br>*Default: F* |
| *hasdone* | *add a "Done" button to the GUI?*<br>*Allowed: boolean*<br>*Default: F* |
| *widgetset* | *Widgetserver* `tool` *to use to build the GUI*<br>*Allowed: Widgetserver* `tool` *or unset*<br>*Default: unset* |

**Returns**

`tool` *having Viewerstandardgui interface*

**Description**

*This function will create a new graphical user interface which can be used to control this Viewerdisplaypanel* `tool`*'s Viewercanvasmanager* `tool`*. By default, the GUI is placed on the screen, although it can be constructed for later use.*

---

**viewerdisplaypanel.newcanvasprintmanagergui - Function**

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Create and (probably) show a new graphical user interface for this* `tool`*'s Viewercanvasprintmanager* `tool`

**Synopsis**
*newcanvasprintmanagergui(parent, show, hasdismiss, hasdone, widgetset)*

**Arguments**

| | |
|---|---|
| *parent* | *parent frame in which to place the Viewercanvasprintmanager* `tool`<br>*Allowed: agent or F*<br>*Default: F* |
| *show* | *show the Viewercanvasprintmanager* `tool`*?*<br>*Allowed: boolean*<br>*Default: T* |
| *hasdismiss* | *add a "Dismiss" button to the GUI?*<br>*Allowed: boolean*<br>*Default: F* |
| *hasdone* | *add a "Done" button to the GUI?*<br>*Allowed: boolean*<br>*Default: F* |
| *widgetset* | *Widgetserver* `tool` *to use to build the GUI*<br>*Allowed: Widgetserver* `tool` *or unset*<br>*Default: unset* |

**Returns**
`tool` *having Viewerstandardgui interface*

**Description**

*This function will create a new graphical user interface which can be used to control this Viewerdisplaypanel* `tool`*'s Viewercanvasprintmanager* `tool`*. By default, the GUI is placed on the screen, although it can be constructed for later use.*

---

### viewerdisplaypanel.testpattern - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Turn a test pattern on or off on the Viewerdisplaypanel* `tool`

**Synopsis**
*testpattern(onoff)*

**Arguments**

| | |
|---|---|
| *onoff* | *turn pattern on (T) or off (F)*<br>*Allowed: boolean* |

**Returns**
*T*

**Description**

*Calling this function with* `onoff=T` *will place a "test pattern" on the Viewerdisplaypanel* `tool` *which is drawn with each refresh. The test pattern is really available for, surprise, surprise, testing! It is actually the test pattern which is used to display the registered colormaps in the graphical user interface for the Viewercolormapmanager* `tool`*.*

---

### viewerdisplaypanel.type - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return the type of this* `tool`

**Synopsis**
*type()*

**Returns**
*'viewerdisplaypanel'*

**Description**

*This function simply returns the string* `'viewerdisplaypanel'`*, which can be used to identify this* `tool` *as a Viewerdisplaypanel* `tool`.

---

### viewerdisplaypanel.registercolormap - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Explicitly register a colormap on the Viewerdisplaypanel* `tool`

**Synopsis**
*registercolormap(map)*

**Arguments**

| | |
|---|---|
| *map* | *name of colormap to register*<br>*Allowed: string* |

**Returns**
*T*

**Description**

*If the given colormap name is valid, then it will be explicitly registered on this Viewerdisplaypanel* `tool`*. The result of this will be a redistribution of the available color cells to the registered colormaps. A change will only be noticed if previously the named colormap was* not *registered on this Viewerdisplaypanel* `tool`.
This function is generally reserved for internal use; careless use of it could cause the Viewer `tool` or AIPS++ to fail.

---

### viewerdisplaypanel.unregistercolormap - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Explicitly unregister a colormap on the Viewerdisplaypanel* `tool`

**Synopsis**

*unregistercolormap(map)*

**Arguments**

| | |
|---|---|
| *map* | *name of colormap to unregister*<br>*Allowed: string* |

**Returns**

*T*

**Description**

*This function simply explicitly* un*registers the named colormap, provided it is a valid colormap and is already registered on this Viewerdisplaypanel* `tool`.
This function is generally reserved for internal use, and careless use of it could cause the Viewer `tool` or AIPS++ to fail.

---

### viewerdisplaypanel.register - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Register a Viewerdisplaydata* `tool` *for drawing*

**Synopsis**

*register(displaydata)*

**Arguments**

| | |
|---|---|
| *displaydata* | *Viewerdisplaydata* `tool` *to register*<br>*Allowed: Viewerdisplaydata* `tool` |

**Returns**

*T, F or fail*

**Description**

*This function registers the given Viewerdisplaydata* `tool` *on this Viewerdisplaydata* `tool`*. That means that at the next and subsequent refreshes, until it is unregistered, the given Viewerdisplaydata* `tool` *will attempt to draw itself on this Viewerdisplaypanel* `tool`*.*
*A return value of* `T` *indicates that the Viewerdisplaydata* `tool` *was successfully registered on this Viewerdisplaypanel* `tool`*. A return value of* `F` *indicates that the Viewerdisplaydata* `tool` *was already registered, so no action was performed. The return value will be* `fail` *if the Viewerdisplaydata* `tool` *was unknown to, or could not be found by, this Viewerdisplaypanel* `tool`*.*

---

### viewerdisplaypanel.unregister - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Unregister a Viewerdisplaydata* `tool` *from this Viewerdisplaypanel* `tool`

**Synopsis**
*unregister(displaydata)*

**Arguments**

| | |
|---|---|
| *displaydata* | *Viewerdisplaydata* `tool` *to unregister*<br>*Allowed: Viewerdisplaydata* `tool` |

**Returns**
*T, F or fail*
*This function removes the given Viewerdisplaydata* `tool` *from the list of Viewerdisplaydata* `tools` *which will (attempt to) draw themselves on this Viewerdisplaypanel* `tool` *at each refresh.*
*A return value of* `T` *indicates that the Viewerdisplaydata* `tool` *was successfully unregistered from this Viewerdisplaypanel* `tool`*. A return value of* `F` *indicates that the Viewerdisplaydata* `tool` *was not previously registered, so no action was performed. The return value will be* `fail` *if the Viewerdisplaydata* `tool` *was unknown to, or could not be found by, this Viewerdisplaypanel* `tool`*.*

---

### viewerdisplaypanel.isregistered - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Query whether a Viewerdisplaydata* `tool` *is registered on this Viewerdisplaypanel* `tool`

**Synopsis**
*isregistered(displaydata)*

**Arguments**

| | |
|---|---|
| *displaydata* | *Viewerdisplaydata* `tool` *to query*<br>*Allowed: Viewerdisplaydata* `tool` |

**Returns**
*T, F or fail*
*This function simply returns* `T` *or* `F` *indicating (respectively) that the given Viewerdisplaydata* `tool` *is or is not currently registered for display on this Viewerdisplaypanel* `tool`*. The return value will be* `fail` *if the Viewerdisplaydata* `tool` *was unknown to, or could not be found by, this Viewerdisplaypanel* `tool`*.*

---

### viewerdisplaypanel.registerednames - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return a vector of the names of registered Viewerdisplaydata* `tools`

**Synopsis**
*registerednames()*

**Returns**
*Vector of strings*

**Description**

*This function returns a vector of names of all Viewerdisplaydata* `tools` *registered on this Viewerdisplaypanel* `tool`*.* This function is generally reserved for internal use.

---

### viewerdisplaypanel.registrationflags - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Return a record describing which Viewerdisplaydata* `tools` *are registered on this* `tool`

**Synopsis**
*registrationflags()*

**Returns**
*record*

**Description**

*This function returns a record whose top-level field names are those of the Viewerdisplaydata* `tools` *known to the Viewer* `tool` *which owns this Viewerdisplaypanel* `tool`*. The value of each field is a boolean, which if T indicates that that Viewerdisplaydata* `tool` *is registered for display on this Viewerdisplaypanel* `tool`*.*
This function is generally reserved for internal use.

---

### viewerdisplaypanel.unregisterall - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Unregister all Viewerdisplaydata* `tools` *from this Viewerdisplaypanel* `tool`

**Synopsis**
*unregisterall()*

**Returns**
*T*

**Description**

*This function simply calls the unregister function for every Viewerdisplaydata* `tool` *that is registered on this Viewerdisplaypanel* `tool`*. It is akin to "clearing" the Viewerdisplaypanel* `tool`*. The unregistrations are wrapped in a* `hold`/`release` *scope, so that only one refresh is made at the end of this function.*

---

### viewerdisplaypanel.unzoom - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Unzoom the current view*

**Synopsis**
*unzoom()*

**Returns**
*T*

**Description**

*This function will modify the coordinates on the Viewerdisplaypanel* `tool` *such that the full view (of at least the first registered Viewerdisplaydata* `tool`*) is visible.*

---

### viewerdisplaypanel.setzoom - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Set a specific zoom*

**Synopsis**
*setzoom(blc, trc)*

**Arguments**

| | |
|---|---|
| *blc* | *bottom-left linear coordinates of zoom*<br>*Allowed: 2-vector* |
| *trc* | *top-right linear coordinates of zoom*<br>*Allowed: 2-vector* |

**Returns**
*T*

**Description**

*This function sets the linear coordinate system of the Viewerdisplaypanel* `tool` *to the specified blc and trc. Linear coordinates are generally identical to the pixel coordinates of the first registered Viewerdisplaydata* `tool`*.*

---

### viewerdisplaypanel.zoom - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **viewer** |
| **Tool** | **viewerdisplaypanel** |

*Zoom by a given factor*

**Synopsis**
*zoom(xfac, yfac)*

**Arguments**

| | |
|---|---|
| *xfac* | *scale factor in X direction*<br>*Allowed: float*<br>*Default: 2.0* |
| *yfac* | *scale factor in Y direction*<br>*Allowed: unset or float*<br>*Default: unset* |

**Returns**
*T*

**Description**