| | |
|---|---|
| *short* | *Short help*<br>*Allowed: String* |
| *long* | *Long help*<br>*Allowed: String*<br>*Default: Unset* |
| *width* | *Width of short help*<br>*Allowed: Integer*<br>*Default: 60* |

**Returns**
*T or fail*

**Description**

*This tool function allows you to set the popuphelp that is attached to the rollup. By default, when the rollup is contructed, you get a basic popuphelp. This enables you to write something more informative to the user if appropriate. As with standard popuphelp, you can specify a short (required) and long (optional) help (i.e. when you click the right button the long help is shown in addition to the short). Likewise the standard popuphelp short-help* `width` *argument is also provided.*

**Example**

```
- r := rollup(f, title='My rollup', show=T);
- r.setpopuphelp('This is my rollup', 'You should know what it does')
```

---

---

### 1.4.32 saveregions - Tool

**Package display**
**Module widgets**

*Tk widget to save image regions to a Table*
*include "saveregions.g"*

**Constructors**

| | |
|---|---|
| *saveregions* | *Construct a saveregions widget* |

### Functions

| | |
|---|---|
| *done* | *Destroy ourselves* |
| *gui* | *Restart the GUI* |
| *setregions* | *Set new regions to save* |
| *settable* | *Set a (new) table* |

### Description

*This Tk widget enables you to save image* `regions-of-interest` *(or more simply, regions - see regionmanager) stored to an AIPS++ Table.*
*The widget displays, in listboxes, the regions passed in. If you select specific regions, then the GUI expands to enable you to rename them during the save. The 'Save' button saves only selected regions. The 'Save all' button saves all regions with no renaming option. If you select 'Confirm', you will be prompted for confirmation if you attempt to overwrite a region in the Table. The 'Dismiss' button dismisses the GUI. You can recover it with the gui function.*
*Under the 'File' menu, you can either dismiss the GUI or destroy the widget (same as invoking function done).*
*The Table can be specified in many ways (at construction and via the entry box in the GUI); a Table* `tool`*, an Image)* `tool`*, the name of a Table (a string) which might be an AIPS++ image, or the name of a* Glish *symbol which is either a Table or an Image* `tool`*. If the Table does not exist, it will be created. It will have no useful structure beyond that needed to store the regions.*
*This widget emits an event called* `saved` *(it has no value) when the regions have been restored. It also emits an event called* `dismissed` *(it has no value) when the dismiss button on the GUI is pressed.*

## saveregions - Constructor

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **saveregions** |

*Construct a saveregions widget*

### Synopsis

*saveregions(parent, table, regions, changenames, globalsave, widgetset)*

### Arguments

| | |
|---|---|
| *parent* | *Parent of the widget*<br>*Allowed: Widget*<br>*Default: F* |
| *table* | *The table of interest*<br>*Allowed: Table tool, image tool or string*<br>*Default: ”* |
| *regions* | *The regions to save, or the region names*<br>*Allowed: Record of regions or vector of strings* |
| *changenames* | *Entry boxes to enable region name changes are provided*<br>*Allowed: T or F*<br>*Default: T* |
| *globalsave* | *The regions are saved from global name space or a record*<br>*Allowed: T or F*<br>*Default: F* |
| *widgetset* | *The widget server to use*<br>*Allowed: widgetserver tool*<br>*Default: dws* |

**Returns**
*saveregions tool or fail*

**Description**

*This is the constructor. If* `parent=F`*, then the widget will make its own parent frame. The Table can be specified in many ways; a Table* `tool`*, an Image)* `tool`*, the name of a Table (a string) which might be an AIPS++ image, or the name of a* Glish *symbol which is either a Table or Image* `tool`*. If the Table does not exists, a be one will be created. It will have no useful structure beyond that needed to store the regions.*
*The regions are either provided in a record (*`globalsave=F` *or should exist in global name space, whereupon you just provide their names.*
*If* `changenames=T`*, when you select regions, entry boxes which enable you to change the names of the regions on save are provided.*

**Example**

```
- include 'saveregions.g'
- r1 := drm.box([10,10],[20,30]);
- r2 := drm.quarter();
- sr1 := saveregions(table='hcn', regions="r1 r2", globalsave=T, changenames=F);
- r := [=];
- r[1] := r1;
```

```
- r[2] := r2;
- sr2 := saveregions(table='hcn', regions=r, globalsave=F);
```

*In this example, two widgets are contructed with a Table called 'hcn'. The first shows how to save from global name space. The second saves from a record.*

---

### saveregions.done - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **saveregions** |

*Destroy ourselves*

**Synopsis**
*done()*

**Returns**
*Bool or fail*

**Description**

*This function destroys the widget.*

**Example**

```
- r := saveregions();
- r
ref [*agent*=<agent>, done=<function>, gui=<function>,
regions=<function>, setnewtable=<function>]
- r.done();
T
- r
F
-
```

---

### saveregions.gui - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **saveregions** |

*Restart the GUI*

**Synopsis**
*gui()*

**Returns**
*Bool*

**Description**

*When the widget is first constructed, it brings up a GUI. When you save regions, the GUI is dismissed. This function restarts the GUI; it will have the same state as when it was dismissed.*

### saveregions.setregions - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **saveregions** |

*Set new regions to save*

**Synopsis**
*setregions(regions)*

**Arguments**

| | |
|---|---|
| *regions* | *Regions or region names*<br>*Allowed: Record of regions or vector of strings* |

**Returns**
*Bool or fail*

**Description**

*This function allows you to change the list of regions available for saving. If you constructed with* `globalsave=T`*, then you pass in a vector of region names. Otherwise, you pass in a record of regions.*

### Example

```
- include 'saveregions.g'
- r1 := drm.box([10,10],[20,30]);
- r2 := drm.quarter();
- sr1 := saveregions(globalsave=T);
- sr1.setregions("r1 r2");
- r[1] := r1;
- r[2] := r2;
- sr2 := saveregions(globalsave=F);
- sr2.setregions(r);
```

*We show how to set regions from the global name space and from a record.*

---

## saveregions.settable - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **saveregions** |

*Set a (new) table*

### Synopsis
*settable(table)*

### Arguments

| | |
|---|---|
| *table* | *The table of interest*<br>*Allowed: Table tool, image tool or string*<br>*Default: ''* |

### Returns
*Bool*

### Description

*This function manually (i.e. not via the GUI interface) sets a (new) Table name. It is equivalent to doing it via the GUI. The Table can be specified in many ways; a Table tool, an Image) tool, the name of a Table (a string) which might be an AIPS++ image, or the name of a* Glish *symbol which is either a Table or Image tool.*