5. **readonly** *toggles the readonly flag enabling editing if false, rbw– >readonly() or rbw– >readonly(F).*

6. **reset** *resets the record to it's original state (only valid if readonly=F), rbw– >reset().*

7. **vectorsize** *sets the limits of arrays that are displayed, rbw– >vectorsize([60,60] (The examples will let you look at arrays up to 60 by 60 in the recordbrowser, larger arrays will be viewed in in the array browser).*

8. **showproperties** *causes a properties dialog for the the recordbrowser widget to be displayed, rbw– >showproperties(). The dialog window allows the interactive setting of vectorsize and font.*

9. **newfield** *if readonly is F, then you may added field to the record, rb– >newfield(fieldName='field1.field2', fieldValue='evalable string'). Note the fieldValue must be an evalable string otherwise bad things happen.*

10. **deletefield** *if readonly is F, then you may delete fields from the record, rb– >deletefield('field1.field2'). Just specify the fields in the record you want to delete.*

The recordbrowser widget does not signal any events.

Example

```
include 'recordbrowserwidget.g'
rb := dws.recordbrowserwidget(therecord=system);
```

Will produce a generic record-browser dialog.

1.4.30 restoreregions - Tool

Package display
Module widgets

Tk widget to restore image regions of interest from a table
include "restoreregions.g"

Constructors

restoreregions	*Construct a restoreregions widget*

Functions

done	*Destroy ourselves*
gui	*Restart the GUI*
refresh	*Refresh the display*
regions	*Recover the selected regions from the Table into a record*
settable	*Set a (new) table*

Description

This widget enables you to restore the image `regions-of-interest` *(or more simply, regions - see Regionmanager) stored in an AIPS++ Table. You can restore the regions into a record, or into global name space (i.e. into* Glish*).*
The widget displays, in listboxes, the regions stored in the Table. If you select specific regions, then the GUI expands to enable you to rename them during the restore. The 'Restore' button restores only selected regions. The 'Restore all' button restores all regions with no renaming option. If you select 'Confirm', you will be prompted for confirmation if you attempt to overwrite a pre-existing Glish *symbol. 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).
You can change the Table you are accessing at any time by entering the new Table name in the appropriate entry box and entering a carriage return.
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`*.*
This widget emits an event called `restored` *(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.*

restoreregions - Constructor

Package	**display**
Module	**widgets**
Tool	**restoreregions**

Construct a restoreregions widget

Synopsis

restoreregions(parent, table, changenames, globalrestore, widgetset)

Arguments

parent	*Parent of the widget* *Allowed: Widget* *Default: F*
table	*The table of interest* *Allowed: Table tool, image tool or string* *Default: ''*
changenames	*Entry boxes to enable region name changes are provided* *Allowed: T or F* *Default: T*
globalrestore	*Restore to global name space or a record* *Allowed: T or F* *Default: F*
widgetset	*The widget server to use* *Allowed: widgetserver tool* *Default: dws*

Returns

Agent

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 an Image* `tool`*.*
If `changenames=T`*, when you select regions, entry boxes which enable you to change the names of the regions on restore are provided.*
If `globalrestore=T`*, the regions are restored to global name space (i.e. to* Glish*). Otherwise they are restored to a record which you recover with function regions.*

Example

```
- include 'restoreregions.g'
- r := restoreregions(table='hcn')
- whenever rr->restored do {
   regions := rr.regions();
  }
```

In this example, the widget is contructed with a Table called `hcn`*. When the user has selected the regions, they are recovered into a record (because* globalrestore=F*). Each field of the record contains one region* tool *(see Regionmanager). The fields of that record are named according to the names of the regions in the table.*

---

restoreregions.done - Function

Package	**display**
Module	**widgets**
Tool	**restoreregions**

Destroy ourselves

Synopsis
done()

Returns
Bool

Description

This tool function destroys the widget.

Example

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

---

restoreregions.gui - Function

Package	**display**
Module	**widgets**
Tool	**restoreregions**

Restart the GUI

Synopsis
gui()

Returns
Bool

Description

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

---

restoreregions.refresh - Function

Package	**display**
Module	**widgets**
Tool	**restoreregions**

Refresh the display

Synopsis
refresh()

Returns
Bool

Description

This function reaccesses the table to find the regions and redisplays them in the GUI.

---

restoreregions.regions - Function

Package	**display**
Module	**widgets**
Tool	**restoreregions**

Recover the selected regions from the Table into a record

Synopsis
regions()

Returns
Record of regions

Description

If you constructed with `globalrestore=T`*, this function returns an empty record. Otherwise, it recovers all the selected regions into a record. Each field of the record contains one region* `tool` *(see Regionmanager). The fields are named according to the names of the regions in the Table.*

Example

```
- include 'restoreregions.g'
- r := restoreregions();
- whenever rr->restored do {
   regions := rr.regions();
  }
```

---

restoreregions.settable - Function

Package	**display**
Module	**widgets**
Tool	**restoreregions**

Set a (new) table

Synopsis
settable(table)

Arguments