1.8 tasking - Module

Running tools and functions within AIPS++
include "tasking.g"

Description

The `tasking` module contains tools that help with the execution of other tools and functions within AIPS++.

`toolmanager` allows execution of tools and functions from the Glish command line and also from Graphical User Interfaces.

`inputsmanager` allows saving and retrieval of inputs to the tools and functions.

`logger` is used for logging messages from tools and functions.

`scripter` allows execution of scripts built up from the toolmanager.

`clipboard` — allows cut-and-paste processing of Glish variables.

`types` defines types of variables in the tasking system.

`plugins` allows commands to be augmented by user-defined plug-in commands.

`asynceval` Allows asynchronous execution of an eval().

Tools

asynceval	Asynchronous eval
clipboard	clipboard object
inputsmanager	Tool for handling inputs for other tools
logger	Tool for logging
plugins	attach code to tools at runtime
scripter	scripter tool
servers	A tool for sending events between glish and precompiled executables
toolmanager	Tool for using AIPS++ tools and functions
types	Tool for defining types known to the tasking system

Functions

is_tool	Is this a tool
toolmanagertest	Test the tool manager
typestest	Test the types definition

1.8.1 toolmanager - Tool

Package utility
Module tasking

Tool for using AIPS++ tools and functions
include "toolmanager.g"

Description

`toolmanager` is the prime interface to the AIPS++ tools and functions. It provides graphical and command line interfaces for using tools and functions.

Using the toolmanager

The use of the `toolmanager` is documented in Getting Started in AIPS++.

Making a tool or function

If you have created an tool that you wish to show up in the toolmanager then the tool must obey the following rules:

- Be a record containing only functions
- Have a function *done* to terminate instances of the tool.
- AND:

  Either have a function `gui` that starts a GUI,

  Or have a meta information file containing a description of the GUI to be built automagically.

If the tool is to be classified correctly, then a help file must be in the system, and built into the documentation. Otherwise the tool will be shown under unclassified (Note: this is considered a serious wart and will be fixed soon). An example of a trival compliant tool is:

```
const foo := function() {
  public := [=];
  public.type := function() {return 'foo'};
  # A do-nothing-much function
  public.gui := function() {
    f:=frame(title='foo.gui');
    b:=button(f, 'Press here to continue');
    await b->press;
    note('foo.gui() finished');
```

```
      f:=F;
      public:=F;
      return T;
    };
    # Another do-nothing-much function
    public.another := function() {
      print "foo.another";
      return T;
    };
    # Another do-nothing-much cli
    public.anothercli := function() {
      print "foo.anothercli";
      return T;
    };
    # Another do-nothing-much gui
    public.anothergui := function() {
      f:=frame(title='foo.anothergui');
      b:=button(f, 'Press here to continue');
      await b->press;
      note('foo.anothergui() finished');
      f:=F;
      public:=F;
      return T;
    }
    return ref public;
  }
  # Add the required meta-information
  types.class('foo').function('ctor_foo');
```

An example of a more detailed meta information file follows:

```
pragma include once; # Include once only

include 'types.g'; # Required

# Name the include file that defines the tool
types.class('imager').includefile('imager.g');

# Constructors are preceeded by ctor_. Then follows the
# list of arguments.  Each argument is defined in this form:
#
# type(arg, default=unset, dir='in', checkeval=T)
#
# where type is defined in type.g and is one of:
#
# file|table|string|integer|float|double|record|boolean|vector_string|complex
```

```
#
# arg is the name of the argument
# default sets default values (may be an executable string)
# dir sets the direction of the argument
# checkeval=T means that a string value is eval'ed and checked,
# otherwise no such check is done.
#
# Define two constructors, one with just a table name, and the other
# with more arguments
types.function('ctor_imager').table('filename').
    function('ctor_imagertester').string('filename', '3C273XC1.ms').
    integer('size', 256).string('cell', '0.7arcsec').string('stokes', 'IV').
    string('coordinates', 'b1950');

# Functions
# Open with just a table name
types.function('open').table('thems');

# Plain close
types.function('close');

# Get the name back
types.function('name').string('return');

# Summary
types.function('summary');

# Set various image parameters. Note the use of the measures
# tool to set the phasecenter.
types.function('setimage').integer('nx',128).integer('ny',128).
    string('cellx','1arcsec').string('celly','1arcsec').
    string('stokes', 'I').
    boolean('doshift', F).
    record('phasecenter', 'dm.direction(\'b1950\', \'0d\', \'0d\')',checkeval=F).
    string('shiftx','0arcsec').string('shifty','0arcsec').
    string('mode', 'mfs').integer('nchan', 1).integer('start', 1).
    integer('step', 1).string('mstart', '0km/s').string('mstep', '0km/s').
    integer('spwid', 1).integer('fieldid', 1);
```

The toolmanager's own constructor is called during startup. Since only one toolmanager can be created, you should never have to call this constructor. The *only* toolmanager that exists has the conventional abbreviation `tm`.

Minimizing use of the toolmanager

If you wish, the toolmanager may be prevented from doing very much by putting the following in your .aipsrc file

```
toolmanager.fullgui: F
```

toolmanager.usegui - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Use GUIs if possible

Synopsis
usegui()

Description
Note that the GUI can only be used if Glish has access to a GUI!
An alternate function of specifying this is via the aipsrc mechanism: add the following to your .aipsrc file

```
toolmanager.gui.auto: T
```

Returns
Bool

---

toolmanager.usecli - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Use CLI

Synopsis
usecli()

Description
The CLI can always be used.

An alternate function of specifying this is via the aipsrc mechanism: add the following to your .aipsrc file

```
toolmanager.gui.auto: F
```

Returns
Bool

---

toolmanager.show - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Show one of: the toolmanager, a tool or function

Synopsis
show(what, prefergui)

Description
`show` can be used in four ways:

- To show the toolmanager:

  ```
  tm.show()
  ```

- Or, to show a GUI or CLI for a specific tool:

  ```
  tm.show('imgr')
  ```

- Or, to show a GUI or CLI for a specific global function:

  ```
  tm.show('imagerwizard')
  ```

- Or, to show a GUI or CLI for a tool constructor:

  ```
  tm.show('imager')
  ```

Arguments

what	Name of thing to be displayed Allowed:String Default: ”	
prefergui	Use a GUI if possible Allowed:Bool Default: T	

Returns
Bool

---

toolmanager.showconstructor - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Show the constructors of a tool

Synopsis
showconstructor(type, prefergui)

Description
Show the constructors of a type of tool. Before a type of tool *e.g.* `imager` can be used, it must be constructed into an actual tool *e.g.* `myimager`. There are usually a number of ways of constructing a tool. `showconstructor` allows one of these to be selected, the arguments set, and the constructor executed.

Arguments

type	Type of the tool to be constructed Allowed:String Default: ”
prefergui	Use a GUI if possible Allowed:Bool Default: T

Returns
Bool

Example

```
- tm.showconstructor('imager')
```

---

toolmanager.showtool - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Show a user interface for a tool

Synopsis
showtool(type, prefergui, forceauto)

Description
Show a user interface (GUI or CLI) for a tool. This interface will allows functions of the tool to be executed.

Arguments

type	Type of the tool to be constructed Allowed:String Default: ''
prefergui	Use a GUI if possible Allowed:Bool Default: T
forceauto	Force use of the auto GUI Allowed:Bool Default: F

Returns
Bool

Example

```
- tm.showtool('myimager')
```

---

toolmanager.showglobalfunction - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Show a user interface for a global function

Synopsis
showglobalfunction(ofn, prefergui, forceauto)

Description
Show a user interface (GUI or CLI) for a global function.

Arguments

ofn	Type of the tool to be constructed Allowed:String Default: ”
prefergui	Use a GUI if possible Allowed:Bool Default: T
forceauto	Force use of the auot GUI Allowed:Bool Default: F

Returns
Bool

Example

```
- tm.showglobalfunction('imagertest')
```

---

toolmanager.showitemmanager - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Show a user interface for a tool function

Synopsis
showitemmanager(ofn, prefergui, forceauto)

Description
Show a user interface (GUI or CLI) for an itemmanager.

Arguments

ofn	Type of the tool to be constructed Allowed:String Default: ”
prefergui	Use a GUI if possible Allowed:Bool Default: T
forceauto	Force use of the auot GUI Allowed:Bool Default: F

Returns
Bool

Example

```
- tm.showitemmanager('defaultmodelmanager')
```

---

toolmanager.where - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Find package, module and description of a type of tool

Synopsis
where(type)

Description
This returns a record containing the package *e.g.* `aips` or `synthesis`, module *e.g.* `images` or `utility`, and description of a type of tool.

Arguments

type	Type of tool Allowed:String Default: ”

Returns
Bool

Example

```
- print tm.where('image')
[package='aips', module='images', description='']
```

---

toolmanager.istool - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Is this a tool?

Synopsis
istool(tool)

Description
Is the named Glish variable a tool? For example, `imager` is a type of tool that can be created, whereas `myimager` is an actual tool.

Arguments

tool	Name of candidate Allowed:String Default: ”

Returns
Bool

Example

```
- tm.istool('imager')
F
- tm.istool('myimager')
T
```

---

toolmanager.istooltype - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Is this a tool type?

Synopsis
istooltype(tool)

Description
Is the named Glish variable a tool? For example, `imager` is a type of tool that can be created, whereas `myimager` is an actual tool.

Arguments

tool	Name of candidate Allowed:String Default: ”

Returns
Bool

Example

```
- tm.istooltype('imager')
T
- tm.istooltype('myimager')
F
```

---

toolmanager.tooltype - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Return tool type

Synopsis
tooltype(tool)

Description
Show the type of the named Glish variable *e.g.* `imager, ms, regionmanager`.

Arguments

tool	Name of candidate Allowed:String Default: "

Returns
Bool

Example

```
- tm.tooltype('myimager')
imager
- tm.tooltype('drm')
regionmanager
```

---

toolmanager.registertool - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Register a tool

Synopsis
registertool(tool, status, description)

Description
Register a tool with the tool manager. This is performed by the toolmanager itself but may also be performed otherwise.

Arguments

tool	Name of tool Allowed:String Default: None
status	Status of tool Allowed:String Default: '-'
description	Description of tool Allowed:String Default: unset

Returns
Bool

Example

```
- tm.registertool('myimager')
T
```

---

toolmanager.settoolstatus - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Set the tool status

Synopsis

settoolstatus(tool, status)

Description
The status of a tool is informative only. Recommended use is the name of an executing function or the string 'Idle'.

Arguments

tool	Name of tool Allowed:String Default: None
status	Status of tool Allowed:String Default: '-'

Returns
Bool

Example

```
- tm.settoolstatus('myimager', 'clean')
T
- tm.settoolstatus('myimager', 'Idle')
T
```

---

toolmanager.deletetool - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Delete a tool

Synopsis
deletetool(tool)

Description
Delete a tool: first unregister the tool with the tool manager, and then delete

the tool. This is performed by the toolmanager itself but may also be performed otherwise.

Arguments

tool	Name of tool Allowed:String Default: None	

Returns
Bool

Example

```
- tm.deletetool('myimager')
T
```

---

toolmanager.killtool - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Kill a tool

Synopsis
killtool(tool)

Description
Kill the server associated with a given tool. Note that multiple tools may be present in one server so that if you have to kill a given tool, you may end up killing its cousins as well. This is a current limitation of the tasking system in AIPS++ that will probably be fixed in version 1.5 or 1.6.

Arguments

tool	Name of tool Allowed:String Default: None	

Returns
Bool

Example

```
- tm.killtool('myimager')
T
```

---

toolmanager.unregistertool - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Unregister a tool

Synopsis
unregistertool(tool)

Description
Unregister a tool with the tool manager. This is performed by the toolmanager itself but may also be performed otherwise.

Arguments

tool	Name of tool Allowed:String Default: None

Returns
Bool

Example

```
- tm.unregistertool('myimager')
T
```

---

toolmanager.isregistered - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Is this tool registered?

Synopsis
isregistered(tool)

Description
Is this tool registered?

Arguments

tool	Name of tool Allowed:String Default: None

Returns
Bool

Example

```
- tm.isregistered('myimager')
T
```

---

toolmanager.toolinfo - Function

Package	**utility**
Module	**tasking**
Tool	**toolmanager**

Show information about a registered tool

Synopsis
toolinfo(tool)

Description
Show information about a registered tool.

Arguments

tool		Name of tool Allowed:String Default: None

Returns
Record

Example

```
- print tm.toolinfo('myimager')
[type='imager', status='Idle', description='Synthesis imaging and deconvolution']
```

1.8.2 is_tool - Function
Package utility
Module tasking

Is this a tool

Synopsis
is_tool(tool)

Description
Check to see if the Glish variable is a valid tool. To be a tool, a variable must be a record, with a field named `type` that is a function returning a string.

Arguments

tool		Variable to be tested Allowed:Glish variable Default: None