Returns
Bool

Example

```
include 'scripter.g'
ds.load('imager_scripter.g');
print ds.getcommands()
ds.reset();
```

1.8.7 logger - Tool

Package utility
Module tasking

Tool for logging
include "logger.g"

Description

The `logger` is the tool managing messages from scripts, clients, etc.. In general it writes the messages into a log file and shows them on the screen in a text window or a GUI.
Log messages are produced by binary executable programs and Glish scripts. You may also create log messages yourself through the "note" member function of logger. You will almost always be interacting with `defaultlogger`, which is also known as `dl` to save you some typing.
Note: Users should use the global note function to create log messages. This has sufficient flexibility to do everything that the "log" and "note" member function of logger can do. note checks for the existance of the defaultlogger and sends messages to the screen if the defaultlogger is not present.
The log messages are sent both to your screen (normally to a GUI, but you can have them come out in your terminal window instead if you prefer) and to a file (an AIPS++table). This file will continue to accumulate messages until you purge them.
By default the log file is `~/aips++/aips++.log`, however you may override this default location by setting some variables in your `~/.aipsrc` file as described below.
The first column is the time that the message was produced. The default is to display the GMT times. If you want to see local times instead, you can set the

time zone offset with another `.aipsrc` variable.
The second panel gives the priority of the message. Messages may presently be of `DEBUGGING`, `NORMAL`, `WARN`, and `SEVERE` priority. When a `SEVERE` level message is shown a pop-up window with a dismiss button also comes up to make sure you don't miss it. In the log window `WARN` and `SEVERE` level messages are highlighted and shown in red instead of green to make them stand out.
The third panel contains the (hopefully!) informative messages. Note that messages may be multiple lines, the other column values are only shown at the start of the message, not for every line in it.
The final panel contains the location where the message came from. This may be of little interest to you.
You can control the panels (size, order, and which to see) by `.aipsrc` variables and by the `Options` menu. They can also be used to control whether a few buttons will be displayed at the bottom. It makes it possible to make the log window as big or small as you like.
It is possible to execute a query on the log window; i.e. to select or sort messages. They will be displayed in a separate log window which is similar to the main window, but also shows the select and sort command used in the query. One can execute another query on that new window, etc..

The `File` menu contains various entries to deal with log messages.

- Query makes it possible to query the log messages as described above. It uses taqlwidget to let you form the select and/or sort command.

- Purge removes old messages from the log file. The last `n` messages will be kept which can be set using a slider. Alternatively a TaQL expression can be given to remove the matching messages. You should purge your log file from time to time.
  This entry is only available in the main log window.

- Refresh refreshes a window resulting from a query. New messages are only added to the main log window. It is possible to add them to a sub window too by pressing `refresh`. It simply re-executes the query.
  The refresh button at the bottom-left of the window does the same.

- Script creates a glish script from logged glish commands. It gets all messages starting with the special prefix and stores them in a file after removing the prefix. That file can be saved and/or be executed. A slider lets you specify the maximum number of commands to be written (counted from the end).

- Print writes the messages into a file and optionally prints them. A slider lets you specify the maximum number of messages to be written (counted from the end). It is also possible to tell which columns have to be printed.

- Dismiss to dismiss this window and all its children.
  The dismiss button at the bottom-right of the window does the same.

Note that the commands above are always executed on the log file and not on the messages in the window. For example, a print is done in a window resulting from a query. This may print more messages than there are in the window, because in the mean time new messages may have been added to the log file.

A dismiss of a window also dismisses its child windows resulting from queries. If you dismiss the main log window, log messages will come out on your terminal instead of the GUI. When you get tired of this, you can type `dl.gui()` at the Glish command prompt to restart the GUI.

The only logger functions you might perhaps call are `loginput()` and `logoutput()` to control what is logged from Glish.

Aipsrc Variables

user.aipsdir	AIPS++info directory Allowed: Any writable directory. If it doesn't exist it will be created Default: ~/aips++
tzoffset	Timezone offset in hours Allowed: e.g., Socorro is -7 Default: 0
logger.file	Log file name (no path!) Allowed: any (none means temporary log file is used) Default: aips++.log
logger.default	Where to write log messages Allowed: gui or screen Default: gui
logger.showbuttons	Show buttons at bottom of the GUI? Allowed: Bool Default: T
logger.glish	What to log from Glish Allowed: input, output, both, or none Default: input
logger.guishow	Which panels to show in GUI and in which order Allowed: any of time,priority,message,origin Default: time,priority,message
logger.height	Height of GUI panels Allowed: Integer Default: 8
logger.timewidth	Width of time panel (if shown) Allowed: Integer Default: 20
logger.prioritywidth	Width of priority panel (if shown) Allowed: Integer Default: 6
logger.messagewidth	Width of message panel (if shown) Allowed: Integer Default: 75
logger.originwidth	Width of origin panel (if shown) Allowed: Integer Default: 25
logger.autoscroll	Scroll GUI to the end after appending a message Allowed: Bool Default: T
logger.background	Normal background color of the GUI Allowed: String Default: xing

Constructors

logger Construct logger tool

Functions

attach	Attach to another log table
brief	Write fewer fields in log messages
getformatted	Get a formatted representation of the log file
gui	Use the GUI
log	Add a message to the logging system
loginput	Log input to Glish?
logoutput	Log output from Glish?
nmessages	How many log messages are in the file
note	Add a note to the logging messages
printtofile	Write some or all of the log file to a text file
purge	Delete some or all messages from the log file.
screen	Use the screen for logging messages
verbose	Write more fields in log messages

logger - Constructor

Package	**utility**
Module	**tasking**
Tool	**logger**

Construct logger tool

Synopsis
logger(use_gui)

Description
The GUI will be used, unless you have your aipsrc variable `logger.default` set to something besides "gui", or X-windows is not enabled for your login session (e.g., the DISPLAY environment variable is not set).

Arguments

use_gui	in	Use GUI for logging messages Allowed:Bool Default: global_use_gui

Returns
tool

logger.gui - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Use the GUI

Synopsis
gui()

Description
An alternate method of specifying the display of logging messages is via the aipsrc mechanism: add the following to your .aipsrc file

```
logger.default: gui
```

Returns
Bool

---

logger.screen - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Use the screen for logging messages

Synopsis
screen()

Description
An alternate method of specifying the display of logging messages is via the aipsrc mechanism: add the following to your .aipsrc file

```
logger.default: screen
```

Actually, anything besides “gui” will send log messages to the screen.

Returns
Bool

---

logger.attach - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Attach to another log table

Synopsis
attach(logfile)

Description
Attach the logger to another log table. If the argument is not a string, the logger is attached to the default log table. If the argument is an empty string, it is attached to a temporary log table.

Arguments

logfile	in	Name of new log table Allowed:String Default: unset

Returns
Bool

Example

```
  dl.attach('my.log');
# T
  dl.attach();
# T
```

The defaultlogger is temporarily attached to table my.log. Thereafter it is reattached to the default log table.

Returns
Bool

---

logger.note - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Add a note to the logging messages

Synopsis
note(..., origin, postglobally, postlocally, postcli)

Description
Add a note to the logging messages with optional origin

Arguments

...	in	Strings to be logged Allowed:Any strings Default: Empty string
origin	in	Origin of note Allowed:String Default: 'logger::note()'
postglobally	in	Write into log table? Allowed:Bool Default: T
postlocally	in	Write to screen? Allowed:Bool Default: T
postcli	in	Show message in CLI window too? Allowed:Bool Default: F

Returns
Bool

Example

```
include 'logger.g';
dl.note('Gone to lunch', 'back in about 1 hour')
```

---

logger.log - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Add a message to the logging system

Synopsis
log(timeString, priority, message, origin, postglobally, postlocally, postcli)

Description
Add a message to the logging system. This is the preferred mechanism for Glish scripts to add messages to the logging message file. Note that although priority=SEVERE is allowed, we prefer that in general, the Glish fail mechanism is used for irrecoverable conditions.

Arguments

timeString	in	time Allowed:String Default: Current time
priority	in	Priority of message Allowed:NORMAL—DEBUGGING—WARN—SEVERE
message	in	Message Allowed:String
origin	in	source of the message Allowed:string Default: [=]
postglobally	in	Write into log table? Allowed:Bool Default: T
postlocally	in	Write to screen? Allowed:Bool Default: T
postcli	in	Show message in CLI window too? Allowed:Bool Default: F

Returns
Bool

Example

```
include 'logger.g';
dl.log('', 'SEVERE', 'File does not exist', 'file.open');
```

---

logger.loginput - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Log input to Glish?

Synopsis

loginput(dolog)

Description

This function turns on (or off) input logging from Glish. That is, if it is called with `T` things that you type into Glish will appear as log messages.

Arguments

dolog	in	Do logging? Allowed:Bool Default: T

Returns

Bool - the current state

---

logger.logoutput - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Log output from Glish?

Synopsis
logoutput(dolog)

Description
This function turns on (or off) input output from Glish. That is, if it is called with `T` things that Glish prints on the screen will also be entered as log messages (with comment signs in front so that they can be rerun).

Arguments

dolog	in	Do logging? Allowed:Bool Default: T

Returns
Bool - the current state

---

logger.nmessages - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

How many log messages are in the file

Synopsis
nmessages()

Description
This function (no arguments) returns the number of messages stored in the log file. If you are using the GUI you probably don't need to know this, however if you are calling any of `purge()`, `printtofile()`, or `getformatted()` by hand for some reason you might want to call this function

Returns
Integer

---

logger.purge - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Delete some or all messages from the log file.

Synopsis
purge(keeplast, expr)

Description
While you will normally want to delete log messages using the GUI, you can do it "by hand" from this function, or you might want to use this function to manage messages in your own scripts.
If an expression is given all matching messages will be removed. Otherwise the last N messages will be kept.

Arguments

keeplast	in	How many messages to keep (from end) Allowed:Int Default: 500
expr	in	Delete messages matching this TaQL expression Allowed:String Default: "

Returns
Bool - function ok

---

logger.printtofile - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Write some or all of the log file to a text file

Synopsis

printtofile(num, filename, colwidth, expr, ascommand)

Description

Again, while you will normally invoke printing from the GUI, you can make a text representation of the log file using this function. It is your responsiblity to actually print the file and delete it when done. The printer tool can do this for you.
The `filename`argument can be given to specify the name of the file to be created. A blank name means that a file `/tmp/aips_NNN` will be created, where NNN is a number to make the name unique.
The `colwidth` argument lets you define if and how wide each column has to be printed. 0 means do not print. -1 means use default width. The order of the columns in this 4-element vector is `time, priority, message, origin`.
It is possible to filter and/or sort the log messages by giving the `WHERE` and `ORDERBY` parts of a TaQL expression in the `expr` argument. This has to be a vector of 0, 1, or 2 strings. If the first string is present, it has to be the `WHERE` expression. If the second string is present, it has to be the `ORDERBY` sort list.
The `ascommand` argument can be used to tell that only logged glish commands should be written (after removing the special prefix). This is, for example, useful to create a glish script from interactively given commands.

Arguments

num	in	How many messages to write (from end) Allowed:Int Default: -1 (all)
filename	in	Name of the output file Allowed:String Default: ”
colwidth	in	Width of each column (0 = do not print; -1 = use default) Allowed:Int vector Default: [-1,-1,-1,-1]
expr	in	select and/or sort command Allowed:String vector Default: ”
ascommand	in	Only write logged glish commands Allowed:Int Default: F)

Returns

String - file name

logger.getformatted - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Get a formatted representation of the log file

Synopsis
getformatted(time, priority, messages, origin, howmany, expr, concat)

Description
This is principally intended to be used by the GUI to refresh itself from the log file (e.g. at startup).

Arguments

time	out	Allowed:String vector
priority	out	Allowed:String vector
messages	out	Allowed:String vector
origin	out	Allowed:String vector
howmany	in	How many to get (from end) Allowed:Int Default: -1 (all)
expr	in	select and/or sort command (see printtofile) Allowed:String vector Default: ”
concat	in	Concatenate consecutive messages with the same priority into one String? Allowed:Bool Default: F

Returns
Bool - success

---

logger.verbose - Function

Package **utility**

Module	**tasking**
Tool	**logger**

Write more fields in log messages

Synopsis
verbose(show_time, show_priority, show_origin)

Description
Verbose by default prints all fields in log messages (time, priority, message, origin).

Arguments

show_time	in	Allowed:boolean Default: T
show_priority	in	Allowed:boolean Default: T
show_origin	in	Allowed:boolean Default: T

Returns
Bool

---

logger.brief - Function

Package	**utility**
Module	**tasking**
Tool	**logger**

Write fewer fields in log messages

Synopsis
brief(show_time, show_priority, show_origin)