and the tool you are about to return to the user (that is, you pass it "public" if you are using the usual closure-tool idiom).
If necessary, it loads all the plugin (`.gp`) files for the requested types, and runs any plugin `init()` functions it finds. It then calls all the `attach()` functions on the loaded plugins. Those attach functions add functions to the tool.

Arguments

type	type (class) for which we want to load plugins Allowed:string
objpublic	the tool the constructor will return to the caller Allowed:record

Returns
reference to objpublic (so you can return it directly)

1.6.26 timer - Tool
Package utility
Module misc

Execute functions periodically, wait for a specified time
include "timer.g"

Constructors

Functions

execute	execute a function periodically
remove	remove a callback function from the timer
wait	wait a specified amount of time

Description
`timer` is an tool which is used to execute a function periodically (or once after a timeout), or to wait for a specified period of time.
`timer` is a single tool, only one will be created in a given glish session. The timer is created the first time `timer.g` is included in a glish session.

Example

We can implement a simple “reminder” as follows:

```
- reminder := function(interval, name) { print 'Do not forget your meeting!'; }
-                                                # Run reminder every 5 minutes
- id := timer.execute(reminder, interval=300, oneshot=F)
- ... time passes ... the following message comes up every 5 minutes
Do not forget your meeting!
- ...
- timer.remove(id);   # Turn off reminder
```

timer.execute - Function

Package	**utility**
Module	**misc**
Tool	**timer**

execute a function periodically

Synopsis
execute(callback, interval, oneshot)

Description
`execute` takes a function which you supply (a “callback”), and executes it one or more times after a time that you specify has passed. The function returns a string, which you can use in the `remove` function to deactivate execution of the function.

The callback function you supply must take two arguments, which will be passed to the function when it is called by `timer`.

interval The time interval that was supplied to `execute()`.

name The name of the timing event, *i.e.*, the value returned by `execute()`.

Often the callback will just ignore these arguments

You can specify that the function is to be executed only once (“oneshot”), or “forever” (until deactivated with `remove`).

Arguments

callback	function to be executed Allowed:two-argument function
interval	how long to wait before executing function (seconds) Allowed:float Default: 60
oneshot	execute function once only? Allowed:boolean Default: T

Returns
string identifier, for remove, or fail

---

timer.remove - Function

Package	**utility**
Module	**misc**
Tool	**timer**

remove a callback function from the timer

Synopsis
remove(name)

Description
`remove` removes a callback function from `timer`. You use the value which returned from the `execute` function to determine which callback to remove. It returns `T` on success, `F` if there is no such callback (for example, it might have already been removed), or "fail" on error.

Arguments

name	identifier returned from execute Allowed:string

Returns
T if callback removed, F if no callback, or fail on error

---

timer.wait - Function

Package	**utility**
Module	**misc**
Tool	**timer**

wait a specified amount of time

Synopsis
wait(interval)

Description
`wait` merely waits the specified period of time, and then returns. C and C++ programmers might think of it as "sleep."

Arguments

interval	how long to wait (seconds) Allowed:float

Returns
T

1.6.27 serverexists - Function

Package	**utility**
Module	**misc**

Does the specified server exist

Synopsis
serverexists(symbol, name, server)
include "serverexists.g"

Arguments

symbol	Symbol of interest Allowed:String
name	The name of the type of server Allowed:String
server	The server tool Allowed:Glish closure tool

Returns
T or F

Description

This function can be used to see whether an AIPS++ server tool exists or not. It relies upon the Glish tool having the function `type` which returns the `name` of the type of server. For example, "quanta" for a `quanta` server, or "regionmanager" for a `regionmanager` server. All of the AIPS++ servers (or tools) should have this function.

Example

```
- include 'quanta,g'
- include 'serverexists.g'
- serverexists('dq', 'quanta', dq);
T
- symbol_delete('dq');
- serverexists('dq', 'quanta', dq);
F
```

1.6.28 aips2loc - Function

Package utility
Module misc

Plot lines of code in AIPS++

Synopsis
aips2loc(file)
include "aips2loc.g"

Constructors