## Contents

1 **pyst2: A Python Interface to Asterisk**  
   1.1 Project Documentation ........................................... 3  
   1.2 Meta ........................................................................... 3  
   1.3 Purpose ....................................................................... 3  
   1.4 Installation ................................................................. 4  
   1.5 Documentation ............................................................. 4  
   1.6 FastAGI ...................................................................... 4  
   1.7 Credits ....................................................................... 4  
   1.8 Things to do for pyst .................................................... 5  

2 **AGI**  
   2.1 Specification ............................................................... 7  

3 **AGITB**  
   3.1 Example .................................................................... 11  
   3.2 Specification ............................................................... 11  

4 **Config**  
   4.1 Example .................................................................... 13  
   4.2 Specification ............................................................... 13  

5 **fastAGI**  
   5.1 Specification ............................................................... 15  

6 **Manager**  
   6.1 Example .................................................................... 17  
   6.2 Specification ............................................................... 18  

7 **Changes** ....................................................................... 19  

8 **Indices and tables** ......................................................... 23  

Python Module Index .......................................................... 25
Contents:
CHAPTER 1

pyst2: A Python Interface to Asterisk

1.1 Project Documentation

http://pyst2.readthedocs.io

1.2 Meta

- Author: Randall Degges
- Email: r@rdegges.com
- Site: http://www.rdegges.com
- Status: looking for maintainer, active

NOTE: This project is in need of an active maintainer. I’m quite busy with other non-asterisk related projects in my personal time, so if you’re a good Python programmer, and working with this library and Asterisk on a daily basis – please consider maintaining the project! Contact me for more information: r@rdegges.com

In the meantime, I’m happy to accept pull requests and cut releases as needed. If you want to contribute to the project, please do!

1.3 Purpose

pyst2 consists of a set of interfaces and libraries to allow programming of Asterisk from python. The library currently supports AGI, AMI, and the parsing of Asterisk configuration files. The library also includes debugging facilities for AGI.
This project has been forked from pyst (http://sf.net/projects/pyst/) because it was impossible for me to contact the project maintainer (after several attempts), and I’d like to bring the project up-to-date, fix bugs, and make it more usable overall.

My immediate plans include adding full documentation, re-writing some of the core routines, adding a test suite, and accepting pull requests.

If you are one of the current maintainers, and would like to take over the fork, please contact me: r@rdegges.com, so we can get that setup!

1.4 Installation

To install pyst2, simply run:

```
$ pip install pyst2
```

This will install the latest version of the library automatically.

1.5 Documentation

Documentation is currently only in python docstrings, you can use python's built-in help facility:

```python
import asterisk
help (asterisk)
import asterisk.agi
help (asterisk.agi)
import asterisk.manager
help (asterisk.manager)
import asterisk.config
help (asterisk.config)
```

Some notes on platforms: We now specify “platforms = ‘Any’” in setup.py. This means, the manager part of the package will probably run on any platform. The agi scripts on the other hand are called directly on the host where Asterisk is running. Since Asterisk doesn’t run on windows platforms (and probably never will) the agi part of the package can only be run on Asterisk platforms.

1.6 FastAGI

FastAGI support is a python based raw SocketServer. To start the server python fastagi.py should start it listening on localhost and the default asterisk FastAGI port. This does require the newest version of pyst2. The FastAGI server runs in as a Forked operation for each request, in an attempt to prevent blocking by a single bad service. As a result the FastAGI server may consume more memory then a single process. If you need to use a single process simply uncomment the appropriate line. Future versions of this will use a config file to set options.

1.7 Credits

Thanks to Karl Putland for writing the original package.

Thanks to Matthew Nicholson for maintaining the package for some years and for handing over maintenance when he was no longer interested.
Thanks to Randall Degges for maintaining this for and accepting pull requests.

## 1.8 Things to do for pyst

This is the original changelog merged into the readme file. I’m not so sure I really want to change all these things (in particular the threaded implementation looks good to me). I will maintain a section summarizing the changes in this README. Detailed changes will be available in the version control tool (currently git).

- **ChangeLog**: The ChangeLog needs to be updated from the monotone logs.
- **Documentation**: All of pyst’s inline documentation needs to be updated.
- **manager.py**: This should be converted to be single threaded. Also there is a race condition when a user calls `manager.logoff()` followed by `manager.close()`. The `close()` function may still call `logoff` again if the socket thread has not yet cleared the `_connected` flag.

A class should be made for each manager action rather than having a function in a manager class. The manager class should be adapted to have a send method that know the general format of the classes.

Matthew Nicholson writes on the mailinglist (note that I’m not sure I’ll do this, I’m currently satisfied with the threaded implementation):

For pyst 0.3 I am planning to clean up the manager.py. There are several know issues with the code. No one has actually reported these as problems, but I have personally had trouble with these. Currently manager.py runs in several threads, the main program thread, a thread to read from the network, and an event distribution thread. This causes problems with non thread safe code such as the MySQLdb libraries. This design also causes problems when an event handler throws an exception that causes the event processing thread to terminate.

The second problem is with the way actions are sent. Each action has a specific function associated with it in the manager object that takes all possible arguments that may ever be passed to that action. This makes the api somewhat rigid and the Manager object cluttered.

To solve these problems I am basically going to copy the design of my Astxx manager library (written in c++) and make it more python like. Each action will be a different object with certain methods to handle various tasks, with one function in the actual Manager class to send the action. This will make the Manager class much smaller and much more flexible. The current code will be consolidated into a single threaded design with hooks to have the library process events and such. These hooks will be called from the host application’s main loop.
2.1 Specification

class asterisk.agi.AGI (stdin=<open file '<stdin>', mode 'r'>, stdout=<open file '<stdout>', mode 'w'>, stderr=<open file '<stderr>', mode 'w'>)

This class encapsulates communication between Asterisk and a python script. It handles encoding commands to Asterisk and parsing responses from Asterisk.

answer ()
agi.answer() -> None Answer channel if not already in answer state.

appexec (application, options="")
Executes <application> with given <options>. Returns whatever the application returns, or -2 on failure to find application.

channel_status (channel="")
agi.channel_status(channel='') -> int Returns the status of the specified channel. If no channel name is given the returns the status of the current channel.

Return values: 0 Channel is down and available 1 Channel is down, but reserved 2 Channel is off hook 3 Digits (or equivalent) have been dialed 4 Line is ringing 5 Remote end is ringing 6 Line is up 7 Line is busy

control_stream_file (filename, escape_digits="", skipms=3000, fwd="", rew="", pause="")
Send the given file, allowing playback to be interrupted by the given digits, if any. escape_digits is a string ‘12345’ or a list of ints [1,2,3,4,5] or strings ['1','2','3'] or mixed [1,‘2’,3,’4’] If sample offset is provided then the audio will seek to sample offset before play starts. Returns digit if one was pressed. Throws AGIError if the channel was disconnected. Remember, the file extension must not be included in the filename.
**database_del** *(family, key)*

agi.database_del(family, key) -> None Deletes an entry in the Asterisk database for a given family and key.

**database_deltree** *(family, key="")*

agi.database_deltree(family, key="") -> None Deletes a family or specific keytree with in a family in the Asterisk database.

**database_get** *(family, key)*

agi.database_get(family, key) -> str Retrieves an entry in the Asterisk database for a given family and key. Returns 0 if <key> is not set. Returns 1 if <key> is set and returns the variable in parenthesis example return code: 200 result=1 (testvariable)

**database_put** *(family, key, value)*

agi.database_put(family, key, value) -> None Adds or updates an entry in the Asterisk database for a given family, key, and value.

**exec_command** *(command, *args)*

Send an arbitrary asterisk command with args (even not AGI commands)

**get_data** *(filename, timeout=2000, max_digits=255)*

agi.get_data(filename, timeout=DEFAULT_TIMEOUT, max_digits=255) -> digits Stream the given file and receive dialed digits

**get_full_variable** *(name, channel=None)*

Get a channel variable.

This function returns the value of the indicated channel variable. If the variable is not set, an empty string is returned.

**get_option** *(filename, escape_digits="", timeout=0)*

agi.get_option(filename, escape_digits='', timeout=0) -> digit Send the given file, allowing playback to be interrupted by the given digits, if any. escape_digits is a string ‘12345’ or a list of ints [1,2,3,4,5] or strings [‘1’, ‘2’, ‘3’] or mixed [1,’2’,3,’4’] Returns digit if one was pressed. Throws AGIError if the channel was disconnected. Remember, the file extension must not be included in the filename.

**get_result** *(stdin=<open file '<stdin>', mode 'r'>)*

Read the result of a command from Asterisk

**get_variable** *(name)*

Get a channel variable.

This function returns the value of the indicated channel variable. If the variable is not set, an empty string is returned.

**hangup** *(channel="")*

Hangs up the specified channel. If no channel name is given, hangs up the current channel

**noop** ()

agi.noop() -> None Does nothing

**receive_char** *(timeout=2000)*

agi.receive_char(timeout=DEFAULT_TIMEOUT) -> chr Receives a character of text on a channel. Specify timeout to be the maximum time to wait for input in milliseconds, or 0 for infinite. Most channels do not support the reception of text.

**record_file** *(filename, format='gsm', escape_digits='#', timeout=20000, offset=0, beep='beep', silence=0)*

agi.record_file(filename, format, escape_digits, timeout=DEFAULT_TIMEOUT, offset=0, beep='beep', silence=0) -> None Record to a file until a given dtmf digit in the sequence is received. Returns ‘-1’ on hangup or error. The format will specify what kind of file will be recorded. The <timeout> is the maximum record time in milliseconds, or ‘-1’ for no <timeout>. <offset samples> is optional, and, if provided, will
seek to the offset without exceeding the end of the file. <silence> is the number of seconds of silence allowed before the function returns despite the lack of dtmf digits or reaching <timeout>. <silence> value must be preceded by ‘s=’ and is also optional.

**say_alpha**(characters, escape_digits='')

agi.say_alpha(string, escape_digits='') -> digit Say a given character string, returning early if any of the given DTMF digits are received on the channel. Throws AGIError on channel failure

**say_date**(seconds, escape_digits='')

agi.say_date(seconds, escape_digits='') -> digit Say a given date, returning early if any of the given DTMF digits are pressed. The date should be in seconds since the UNIX Epoch (Jan 1, 1970 00:00:00)

**say_datetime**(seconds, escape_digits='', format='', zone='')

agi.say_datetime(seconds, escape_digits='', format='', zone='') -> digit Say a given date in the format specified (see voicemail.conf), returning early if any of the given DTMF digits are pressed. The date should be in seconds since the UNIX Epoch (Jan 1, 1970 00:00:00).

**say_digits**(digits, escape_digits='')

agi.say_digits(digits, escape_digits='') -> digit Say a given digit string, returning early if any of the given DTMF digits are received on the channel. Throws AGIError on channel failure

**say_number**(number, escape_digits='')

agi.say_number(number, escape_digits='') -> digit Say a given digit string, returning early if any of the given DTMF digits are received on the channel. Throws AGIError on channel failure

**say_phonetic**(characters, escape_digits='')

agi.say_phonetic(string, escape_digits='') -> digit Phonetically say a given character string, returning early if any of the given DTMF digits are received on the channel. Throws AGIError on channel failure

**say_time**(seconds, escape_digits='')

agi.say_time(seconds, escape_digits='') -> digit Say a given time, returning early if any of the given DTMF digits are pressed. The time should be in seconds since the UNIX Epoch (Jan 1, 1970 00:00:00)

**send_command**(command, *args)

Send a command to Asterisk

**send_image**(filename)

agi.send_image(filename) -> None Sends the given image on a channel. Most channels do not support the transmission of images. Image names should not include extensions. Throws AGIError on channel failure

**send_text**(text='')

agi.send_text(text='') -> None Sends the given text on a channel. Most channels do not support the transmission of text. Throws AGIError on error/hangup

**set_autohangup**(secs)

agi.set_autohangup(secs) -> None Cause the channel to automatically hangup at <secs> seconds in the future. Of course it can be hungup before then as well. Setting to 0 will cause the autohangup feature to be disabled on this channel.

**set_callerid**(number)

agi.set_callerid(number) -> None Changes the callerid of the current channel.

**set_context**(context)

Sets the context for continuation upon exiting the application. No error appears to be produced. Does not set exten or priority Use at your own risk. Ensure that you specify a valid context.

**set_extension**(extension)

Sets the extension for continuation upon exiting the application. No error appears to be produced. Does not set context or priority Use at your own risk. Ensure that you specify a valid extension.
set_priority \((priority)\)
Sets the priority for continuation upon exiting the application. No error appears to be produced. Does not set extension or context. Use at your own risk. Ensure that you specify a valid priority.

set_variable \((name, value)\)
Set a channel variable.

stream_file \((filename, escape_digits='', sample_offset=0)\)
agi.stream_file(filename, escape_digits='', sample_offset=0) \(\rightarrow\) digit Send the given file, allowing playback to be interrupted by the given digits, if any. escape_digits is a string ‘12345’ or a list of ints [1,2,3,4,5] or strings ['1','2','3'] or mixed [1,'2',3,'4'] If sample offset is provided then the audio will seek to sample offset before play starts. Returns digit if one was pressed. Throws AGIError if the channel was disconnected. Remember, the file extension must not be included in the filename.

tdd_mode \((mode='off')\)
agi.tdd_mode(mode='on'|'off') \(\rightarrow\) None Enable/Disable TDD transmission/reception on a channel. Throws AGIAppError if channel is not TDD-capable.

test_hangup \()\)
This function throws AGIHangup if we have received a SIGHUP.

verbose \((message, level=1)\)
agi.verbose(message=' ', level=1) \(\rightarrow\) None Sends <message> to the console via verbose message system. <level> is the the verbose level (1-4)

wait_for_digit \((timeout=2000)\)
agi.wait_for_digit(timeout=DEFAULT_TIMEOUT) \(\rightarrow\) digit Waits for up to ‘timeout’ milliseconds for a channel to receive a DTMF digit. Returns digit dialed. Throws AGIError on channel failure.

exception asterisk.agi.AGIAppError
exception asterisk.agi.AGIDBError
exception asterisk.agi.AGIIError
exception asterisk.agi.AGIException
exception asterisk.agi.AGIHangup
exception asterisk.agi.AGIInvalidCommand
exception asterisk.agi.AGIResultHangup
exception asterisk.agi.AGISIGHUPHangup
exception asterisk.agi.AGISIGPIPEHangup
exception asterisk.agi.AGIUnknownError
exception asterisk.agi.AGIUsageError
3.1 Example

To enable this module, do:

```python
import asterisk.agitb, asterisk.agi
asterisk.agitb.enable(display = False, logdir = '/var/log/asterisk/')
agi = asterisk.agi.AGI()
asterisk.agitb.enable(agi, False, '/var/log/asterisk')
```

at the top of your script. The optional arguments to enable() are:

- **agi** - the agi handle to write verbose messages to
- **display** - if true, tracebacks are displayed on the asterisk console (used with the agi option)
- **logdir** - if set, tracebacks are written to files in this directory
- **context** - number of lines of source code to show for each stack frame

By default, tracebacks are displayed but not saved, and the context is 5 lines.

You may want to add a logdir if you call agitb.enable() before you have an agi.AGI() handle.

Alternatively, if you have caught an exception and want agitb to display it for you, call agitb.handler(). The optional argument to handler() is a 3-item tuple (etype, evalue, etb) just like the value of sys.exc_info(). If you do not pass anything to handler() it will use sys.exc_info().

This script was adapted from Ka-Ping Yee’s cgitb.

3.2 Specification

```python
class asterisk.agitb.Hook(display=1, logdir=None, context=5, file=None, agi=None)
    A hook to replace sys.excepthook that shows tracebacks in HTML.
```
asterisk.agitb.enable(agi=None, display=1, logdir=None, context=5)

Install an exception handler that formats tracebacks as HTML.

The optional argument ‘display’ can be set to 0 to suppress sending the traceback to the browser, and ‘logdir’
can be set to a directory to cause tracebacks to be written to files there.

asterisk.agitb.lookup(name, frame, locals)

Find the value for a given name in the given environment.

asterisk.agitb.scanvars(reader, frame, locals)

Scan one logical line of Python and look up values of variables used.

asterisk.agitb.text(eparams, context=5)

Return a plain text document describing a given traceback.
This module provides parsing functionality for asterisk config files.

4.1 Example

```python
import asterisk.config
import sys

# load and parse the config file
try:
    config = asterisk.config.Config('/etc/asterisk/extensions.conf')
except asterisk.config.ParseError as e:
    print "Parse Error line: %s: %s" % (e.line, e.strerror)
    sys.exit(1)
except IOError as e:
    print "Error opening file: %s" % e.strerror
    sys.exit(1)

# print our parsed output
for category in config.categories:
    print '\[%s\]' % category.name  # print the current category

    for item in category.items:
        print '  %s = %s' % (item.name, item.value)
```

4.2 Specification

`exception asterisk.config.ParseError`
Requires modified pyst2 to support reading stdin/out/err

Copyright 2011 VOICE1, LLC By: Ben Davis <ben@voice1-dot-me>

### 5.1 Specification
This module provides a Python API for interfacing with the asterisk manager.

### 6.1 Example

```python
import asterisk.manager
import sys

def handle_shutdown(event, manager):
    print "Recieved shutdown event"
    manager.close()
    # we could analize the event and reconnect here

def handle_event(event, manager):
    print "Recieved event: %s" % event.name

manager = asterisk.manager.Manager()

try:
    # connect to the manager
    try:
        manager.connect('host')
        manager.login('user', 'secret')

        # register some callbacks
        manager.register_event('Shutdown', handle_shutdown)  # shutdown
        manager.register_event('*', handle_event)  # catch all

        # get a status report
        response = manager.status()

        manager.logoff()
    except asterisk.manager.ManagerSocketException as e:
        print "Error connecting to the manager: %s" % e.strerror
```

(continues on next page)
Remember all header, response, and event names are case sensitive.

Not all manager actions are implemented as of yet, feel free to add them and submit patches.

## 6.2 Specification

```python
class asterisk.manager.Event(message):
    Manager interface Events, __init__ expects a ‘Event’ message
    get_header(hname, defval=None)
        Return the specified header
    has_header(hname)
        Check for a header

exception asterisk.manager.ManagerAuthException

exception asterisk.manager.ManagerException

class asterisk.manager.ManagerMsg(response):
    A manager interface message
    get_header(hname, defval=None)
        Return the specified header
    has_header(hname)
        Check for a header
    parse(response)
        Parse a manager message

exception asterisk.manager.ManagerSocketException
```


2016-11-04 VoiPCS <https://github.com/VoIPCS>
  • Fixing unicode bug in send_action for AMI.

2016-02-16 Randall Degges <https://github.com/rdegges>
  • Fixing issue in AGI class init function. Thanks to @sancho2934489 for the find!

2015-11-16 Scinawa Antani <https://github.com/Scinawa>
  • Fixing indentation errors.

2015-11-14 Ben Davis <https://github.com/tuxpowered>
  • Handling more UTF-8 encoding issues in _quote method.

2015-07-18 Artem Sorokin
  • Fixing UTF-8 encoding issues.

2015-07-15 Artem Sorokin
  • Fix multiline command end marker for OpenVox GSM Gateway.

2015-03-31 Randall Degges
  • Porting packaging to setuptools (modern).
  • Adding six as a dependency (it was missing before).

2015-03-30 Areski Belaid
  • Fixing the MANIFEST.in file I accidentally broke :)

2015-03-29 Timur Tuchkovenko <eill@yandex.ru>
  • UPGRADE: AMI fix for Python 3 compatibility.

2014-10-08 Timur Tuchkovenko <eill@yandex.ru>
  • UPGRADE: initial Python 3 support. Now pyst2 requires Python ‘six’ module. Some minor changes in other files.
2014-09-14 Sp1tF1r3 <https://github.com/Sp1tF1r3>
• asterisk/manager.py: added action ‘Reload’ for Asterisk Manager Interface (AMI).

2013-12-03 Ludovic Gasc <gmludo@gmail.com>
• examples/agi_script.py: added example script to explain AGI functionality.
• README: renamed to REAMDE.rst for Github’s Markdown support.
• setup.py: minor changes.

2012-11-12 Arezqui Belaid <areski@gmail.com>
• asterisk/manager.py: minor empty line enhancements.
• examples/show_channels.py: added example script to show information via Asterisk Manager Interface (AMI).

2012-11-11 Arezqui Belaid <areski@gmail.com>
• PEP8 Fixes

2011-05-31 Randall Degges <rdegges@gmail.com>
• BUGFIX: Fixing issue that prevented manager.status command from returning proper output.

2007-01-26 Matthew Nicholson <mnicholson@digium.com>
• asterisk/manager.py: Make get_header() functions work like dict.get().
• UPGRADE: Updated.

2007-01-16 Matthew Nicholson <mnicholson@digium.com>
• asterisk/manager.py: Fix support for Manager.command(). Patch from Karl Putland <karl@klasstek.com>.

2007-01-02 Matthew Nicholson <mnicholson@digium.com>
• asterisk/agi.py (AGI.set_autohangup): Fixed syntax error.

2006-11-28 Matthew Nicholson <mnicholson@digium.com>
• UPGRADE: Tweaked formatting.

2006-10-30 Matthew Nicholson <mnicholson@digium.com>
• ChangeLog: Fixed previous entry.

2006-10-30 Matthew Nicholson <mnicholson@digium.com>
• TODO: Updated.
• asterisk/agi.py (AGI.control_stream_file): Changed default skipms and quoted arguments.

2006-10-24 Matthew Nicholson <mnicholson@digium.com>
• asterisk/agi.py: Added get_variable_full command.

2006-10-18 Matthew Nicholson <mnicholson@digium.com>
• asterisk/agitb.py: Make error output default to sys.stderr instead of sys.stdout.

2006-09-19 Matthew Nicholson <mnicholson@digium.com>
• debian/control: Removed XS-Python-Versions header to make it default to all python versions.

2006-09-19 Matthew Nicholson <mnicholson@digium.com>
• setup.py: Updated version.

2006-09-19 Matthew Nicholson <mnicholson@digium.com>
• debian/rules: Changed to use pysupport.
• debian/control: Changed to use pysupport and changed arch to all.

2006-09-19 Matthew Nicholson <mnicholson@digium.com>
• MANIFEST.in: Added NEWS to manifest.

2006-09-19 Matthew Nicholson <mnicholson@digium.com>
• debian/rules: Updated to reflect new python policy.
• debian/control: Updated to reflect new python policy.
• debian/changelog: Updated.

2006-08-23 Matthew Nicholson <mnicholson@digium.com>
• UPGRADE: Updated.

2006-08-23 Matthew Nicholson <mnicholson@digium.com>
• asterisk/manager.py (unregister_event): Added.

2006-08-23 Matthew Nicholson <mnicholson@digium.com>
• NEWS: Added.

2006-07-14 Matthew Nicholson <mnicholson@digium.com>
• asterisk/agi.py (wait_for_digit): Only catch ValueError, not all exceptions.

2006-07-14 Matthew Nicholson <mnicholson@digium.com>
• TODO: Updated.
• asterisk/agi.py (set_variable): Documentation changes.
• asterisk/agi.py (get_variable): Changed to return and empty string instead of throwing an exception when a channel variable is not set. * UPGRADE: Added.

2006-07-14 Matthew Nicholson <mnicholson@digium.com>
• ChangeLog: Added.
• TODO: Added.
• MANIFEST.in: Added ChangeLog and TODO.
CHAPTER 8

Indices and tables

- genindex
- modindex
- search
Python Module Index

a
agi, 7
agitb, 11
asterisk.agi, 7
asterisk.agitb, 11
asterisk.config, 13
asterisk.fastagi, 15
asterisk.manager, 17

c
config, 13

f
fastagi, 15

m
manager, 17
Index

A
AGI (class in asterisk.agi), 7
agi (module), 7
AGIAppError, 10
AGIDBError, 10
AGIError, 10
AGIXception, 10
AGIHangup, 10
AGIInvalidCommand, 10
AGIResultHangup, 10
AGISIGHUPHangup, 10
AGISIGPIPEHangup, 10
agitb (module), 11
AGIUnknownError, 10
AGIUsageError, 10
answer() (asterisk.agi.AGI method), 7
appexec() (asterisk.agi.AGI method), 7
asterisk.agi (module), 7
asterisk.agitb (module), 11
asterisk.config (module), 13
asterisk.fastagi (module), 15
asterisk.manager (module), 17

B
C
channel_status() (asterisk.agi.AGI method), 7
config (module), 13
control_stream_file() (asterisk.agi.AGI method), 7

D
database_del() (asterisk.agi.AGI method), 7
database_deltree() (asterisk.agi.AGI method), 8
database_get() (asterisk.agi.AGI method), 8
database_put() (asterisk.agi.AGI method), 8

E
event (class in asterisk.manager), 18
exec_command() (asterisk.agi.AGI method), 8

F
fastagi (module), 15

G
gdata() (asterisk.agi.AGI method), 8
g_full_variable() (asterisk.agi.AGI method), 8
g_header() (asterisk.manager.Event method), 18
g_header() (asterisk.manager.ManagerMsg method), 18
g_option() (asterisk.agi.AGI method), 8
g_result() (asterisk.agi.AGI method), 8
g_variable() (asterisk.agi.AGI method), 8

H
hangup() (asterisk.agi.AGI method), 8
has_header() (asterisk.manager.Event method), 18
has_header() (asterisk.manager.ManagerMsg method), 18
Hook (class in asterisk.agitb), 11

L
lookup() (in module asterisk.agitb), 12

M
manager (module), 17
ManagerAuthException, 18
ManagerException, 18
ManagerMsg (class in asterisk.manager), 18
ManagerSocketException, 18

N
noop() (asterisk.agi.AGI method), 8

P
parse() (asterisk.manager.ManagerMsg method), 18
ParseException, 13

R
receive_char() (asterisk.agi.AGI method), 8
record_file() (asterisk.agi.AGI method), 8
S
say_alpha() (asterisk.agi.AGI method), 9
say_date() (asterisk.agi.AGI method), 9
say_datetime() (asterisk.agi.AGI method), 9
say_digits() (asterisk.agi.AGI method), 9
say_number() (asterisk.agi.AGI method), 9
say_phonetic() (asterisk.agi.AGI method), 9
say_time() (asterisk.agi.AGI method), 9
scanvars() (in module asterisk.agitb), 12
send_command() (asterisk.agi.AGI method), 9
send_image() (asterisk.agi.AGI method), 9
send_text() (asterisk.agi.AGI method), 9
set_autohangup() (asterisk.agi.AGI method), 9
set_callerid() (asterisk.agi.AGI method), 9
set_context() (asterisk.agi.AGI method), 9
set_extension() (asterisk.agi.AGI method), 9
set_priority() (asterisk.agi.AGI method), 9
set_variable() (asterisk.agi.AGI method), 10
stream_file() (asterisk.agi.AGI method), 10

T
tdd_mode() (asterisk.agi.AGI method), 10
test_hangup() (asterisk.agi.AGI method), 10
text() (in module asterisk.agitb), 12

V
verbose() (asterisk.agi.AGI method), 10

W
wait_for_digit() (asterisk.agi.AGI method), 10