

---

**mod\_python has been rewritten and the documentation below is pending a re-write. See [migrate mod python 8998](#) for changes you will need to make to scripts.**

---

## Contents

- [1 Getting Started](#)
  - ◆ [1.1 Building mod\\_python](#)
    - ◇ [1.1.1 Install python-dev package](#)
    - ◇ [1.1.2 Enable compilation in modules.conf](#)
    - ◇ [1.1.3 Notes regarding Configure Script](#)
  - ◆ [1.2 Enabling mod\\_python](#)
  - ◆ [1.3 PYTHONPATH](#)
    - ◇ [1.3.1 Copy or Symlink to site-packages directory](#)
    - ◇ [1.3.2 Adding to PYTHONPATH environment variable](#)
  - ◆ [1.4 Invoking mod\\_python applications](#)
- [2 Python module specification](#)
- [3 API](#)
- [4 Sample Python Scripts](#)
  - ◆ [4.1 Hello World via call](#)
  - ◆ [4.2 Hello World via cmd line](#)
  - ◆ [4.3 Hello World - Dialplan API](#)
  - ◆ [4.4 Run something in a thread using API](#)
  - ◆ [4.5 More samples](#)
  - ◆ [4.6 Fetch Effective Caller Name from CSV file using Python](#)
- [5 FAQ](#)
  - ◆ [5.1 Does each script spawn a python interpreter?](#)
  - ◆ [5.2 Are there thread safety issues?](#)
  - ◆ [5.3 I changed a module I'm importing, and nothing happened](#)
  - ◆ [5.4 How do I pass arguments to the script?](#)
  - ◆ [5.5 Can I test scripts using python shell?](#)
  - ◆ [5.6 Can it serve configuration \(like Lua\)?](#)
- [6 Gotchas](#)
  - ◆ [6.1 String Substitution in Functions](#)
- [7 Troubleshooting](#)
  - ◆ [7.1 Script fails but I don't see a stack trace](#)
  - ◆ [7.2 Error:TypeError: CoreSession.playAndGetDigits\(\) takes exactly 9 arguments \(10 given\)](#)
  - ◆ [7.3 Cannot import FreeSWITCH](#)
  - ◆ [7.4 Cannot import time](#)
  - ◆ [7.5 CoreSession.setDTMFCallback\(\) takes exactly 4 arguments ..](#)
  - ◆ [7.6 Message: 'module' object has no attribute 'EventConsumer\\_node\\_set'](#)

## Mod\_python

- ◆ [7.7 CoreSession\\_streamfile\(\) takes exactly 3 arguments \(4 given\)](#)
- ◆ [7.8 Error calling DTMF callback - wrong # of arguments](#)
- ◆ [7.9 'console\\_log'. argument 2 of type 'char \\*'](#)
- ◆ [7.10 Channels are not being cleaned up](#)
- ◆ [7.11 Avoid module-level global variables](#)
- ◆ [7.12 Build error: Python.h: No such file or directory](#)
- ◆ [7.13 Build error: /usr/bin/ld: cannot find -lpython2.5](#)
- ◆ [7.14 ImportError: ../datetime.so: undefined symbol: PyExc IOError](#)
- ◆ [7.15 ImportError: /usr/lib/./datetime.so: undefinedsymbol: Py ZeroStruct](#)
- [8 Known Bugs](#)
  - ◆ [8.1 Hangup hook + SQLAlchemy crashes switch](#)
  - ◆ [8.2 Makefile deploys to /usr/lib/python2.4/site-packages even if python 2.5](#)
  - ◆ [8.3 read\(\) function might have problems](#)
  - ◆ [8.4 FIXED: Django database connections are not cleaned up](#)
  - ◆ [8.5 NEEDS RETESTING: python IVR's that execute nested python applications crash](#)
  - ◆ [8.6 NEEDS RETESTING:transferring to an extension that runs a python IVR hangs](#)
  - ◆ [8.7 NEEDS RETESTING:Using originate\(\) and bridge\(\) results in the audio only flowing one direction](#)
  - ◆ [8.8 NEEDS RETESTING:Strange errors on shutdown](#)
  - ◆ [8.9 NEEDS RETESTING:Unloading and reloading the module does not seem to work](#)
- [9 See Also](#)

## Getting Started

### Building mod\_python

#### Install python-dev package

On Debian/Ubuntu:

```
apt-get install python2.5 python2.5-dev
```

Be careful not to have a previously installed version of python (like python2.4). Otherwise it will try to compile with this version, and you will need either to remove python2.4 package, or install the development header for it.

#### Enable compilation in modules.conf

- edit modules.conf and uncomment languages/mod\_python
- make installall

#### Notes regarding Configure Script

The configure script will try to detect your existing python, if it cannot find it or it does not support multi-threading, it will print a warning. Otherwise, it will write a Makefile that uses the detected version.

## Mod\_python

You can specify arguments to configure to make it use a particular version (you really only need to manually specify them if the default doesn't work for you). Both args need to be given:

- `--with-python` (eg, `--with-python=/usr/bin/python-2.5`)
- `--with-python-config` (eg, `--with-python-config=/usr/bin/python-config-2.5`)

NOTE: if you use custom locations you have to make sure python and python-config are from the **same** version

**NOTE #2: As of svn revision 8455 / 1.0\_rc6 --with-python-config has been removed!**

NOTE #3 - Make current auto fixes the makefile in case it doesn't get generated (correctly, or at all). make `python-reconf` to regenerate it.

## Enabling mod\_python

Open up `conf/autoload_configs/modules.conf.xml` and add an entry

```
<load module="mod_python"/>
```

## PYTHONPATH

There are two different ways to tell the python interpreter how to find python modules. If you don't do either of these, the embedded python interpreter will have *no way* to find your python scripts.

Assuming you have

```
<action application="python" data="foo.bar"/>
```

This is telling python to load the **bar** *module* that lives in the **foo** *package*.

## Copy or Symlink to site-packages directory

If the source file is in `/usr/src/foo/bar.py`:

```
cd /path/to/python/site-packages
ln -s /usr/src/foo .
```

The same can be done via copying. Don't forget, the foo package directory will need an `__init__.py`.

## Adding to PYTHONPATH environment variable

If the file is in `/usr/src/foo/bar.py`, add the following to your system environment startup

```
export PYTHONPATH=$PYTHONPATH:/usr/src
```

Don't forget, the foo package directory will need an `__init__.py`.

## Mod\_python

In the shell where freeswitch is started, this environment variable will need to be defined.

## Invoking mod\_python applications

To call a Python application from dialplan, you should probably be familiar with the [Dialplan](#). You simply call it as an application similar to:

```
<action application="python" data="foo.bar"/>
```

The module is **bar**, in the **foo** package. See the [Finding python modules](#) section to tell the embedded python interpreter how to find this module.

If your module (say, test.py) is not in any package directory, then you would instead use:

```
<action application="python" data="test"/>
```

In both cases, you need to leave off the .py file extension otherwise it will not work. It expects a fully qualified module name only.

You would put this in your 'dialplan' if using the XML dialplan module with freeswitch. Don't forget your 'condition' tags and all that goodness.

Its posible to call python script from CLI with following format:

```
freeswitch> python foo.bar
```

**NOTE:** if you invoke it this way, your python **handler()** function will be called with no arguments.

## Python module specification

Your python module must define a function called handler that takes two arguments:

- session
- args

Eg, *def handler(session, args).*

The session is the main interface to freeswitch, which wraps a freeswitch session, and the args are any args passed to the handler script.

## API

Supports the same API as [Mod\\_lua](#)

# Sample Python Scripts

## Hello World via call

In the case that an extension has been mapped to the python module in the dialplan, here is what bar.py should look like:

```
import os
from freeswitch import *

# HANGUP HOOK
#
# session is a session object
# what is "hangup" or "transfer"
# if you pass an extra arg to setInputCallback then append 'arg' to get that value
#
# WARNING: known bugs with hangup hooks, use with extreme caution
def hangup_hook(session, what):

    consoleLog("info","hangup hook for %s!!\n\n" % what)
    return

# INPUT CALLBACK
#
# session is a session object
# what is "dtmf" or "event"
# obj is a dtmf object or an event object depending on the 'what' var.
# if you pass an extra arg to setInputCallback then append 'arg' to get that value
# def input_callback(session, what, obj, arg):
def input_callback(session, what, obj):

    if (what == "dtmf"):
        consoleLog("info", what + " " + obj.digit + "\n")
    else:
        consoleLog("info", what + " " + obj.serialize() + "\n")
    return "pause"

# APPLICATION
#
# default name for apps is "handler" it can be overridden with <modname>::<function>
# session is a session object
# args is all the args passed after the module name
def handler(session, args):

    session.answer()
    session.setHangupHook(hangup_hook)
    session.setInputCallback(input_callback)
    session.execute("playback", session.getVariable("hold_music"))
```

## Mod\_python

EDIT: Notice the `input_callback` will never be called even if you press digit during the music on hold. Because playback return only at call end. If you want to catch DTMF, use `session.streamFile(session.getVariable("hold_music"))` for example. Took me a few hour to understand that. Hope it help.

## Hello World via cmd line

See

[http://fisheye.freeswitch.org/browse/~raw,r=9052/FreeSWITCH/src/mod/languages/mod\\_python/python\\_example.py](http://fisheye.freeswitch.org/browse/~raw,r=9052/FreeSWITCH/src/mod/languages/mod_python/python_example.py)

## Hello World - Dialplan API

```
<action application="set"
    data="foo=${python(my_script) }"/>
```

So foo channel variable is set to the output of the `my_script` python script.

The script will be called with a magic object called "stream" which has the method `write`, and the anything written to this method will be the script's output. So for example

```
def fsapi(session, stream, env, args):
    stream.write("hello")
```

will cause the foo variable to be set to "hello".

See

[http://fisheye.freeswitch.org/browse/~raw,r=9052/FreeSWITCH/src/mod/languages/mod\\_python/python\\_example.py](http://fisheye.freeswitch.org/browse/~raw,r=9052/FreeSWITCH/src/mod/languages/mod_python/python_example.py)

## Run something in a thread using API

Usage of the API in Python is conceptually identical to usage in other supported languages.

The example scripts below employ the API, along with the 'runtime' function described in `python_example.py` to run a job in a thread.

This approach provides a means of implementing non-blocking code without employing `mod_event_socket` and may be suitable/useful for cleanup or post-processing.

This example includes two modules. The first module is a virtual copy of the default example script with a couple of notable differences.

```
import os
from freeswitch import *

# WARNING: known bugs with hangup hooks, use with extreme caution
```

## Mod\_python

```
def hangup_hook(session, what):

    consoleLog("info","hangup hook for %s!!\n\n" % what)
    return

def input_callback(session, what, obj):

    if (what == "dtmf"):
        consoleLog("info", what + " " + obj.digit + "\n")
    else:
        consoleLog("info", what + " " + obj.serialize() + "\n")
    return "pause"

def handler(session, args):

    session.answer()
    session.setHangupHook(hangup_hook)
    session.setInputCallback(input_callback)
    session.streamFile("/my/test/audio.wav")
    session.hangup() #hangup the call
    #Now run another python script in a thread. If we
    # don't do it this way all subsequent work will block
    # the hangup message from being sent to the client
    new_api_obj = API()
    new_api_obj.executeString( \
        "pyrun foo.postprocessing " + \
        session.getVariable('caller_id_number') )
```

The second module, "postprocessing", handles our post-processing needs and for convenience resides in the same package, "foo":

```
import os, sys, time
from freeswitch import *

# everything after the command (in this case pyrun) and
# the module name (in this case foo.postprocessing) will
# be interpreted as a string and handed to our 'runtime'
# function where it will be accessible via the argument 'arg1'
def runtime(arg1):
    time.sleep(10) #this is just to test that we are actually
                  # running in a separate thread.
    consoleLog( "info", "Caller: %s hung up 10s ago!\n" % arg1 )
```

When running the above example, the client should receive a hangup immediately after streamFile returns. 10 seconds later the "Caller: xxxx hung up 10s ago!" message should be printed to the console.

## More samples

### Fetch Effective Caller Name from CSV file using Python

The purpose of this script is simply to associate foreign caller-id without caller\_id\_name to a static caller\_id\_name using a csv file, and at the same time it shows how to work with Python and CSV files inside FreeSWITCH

## Mod\_python

First you must add something like that to the generic dialplan for your local extensions (personally i use *Local\_Extension* in **dialplan/default.xml**):

```
<action application="set" data="caller=${caller_id_number}"/>
<action application="python" data="setCallerName"/>
```

It's pretty self-explanatory, it create a variable which will be used in the python script that we call on the second line. **setCallerName** is the name of your script in **\${FS\_ROOT}/scripts/**

Here the code in **\${FS\_ROOT}/scripts/setCallerName.py** :

```
import csv
from freeswitch import *

def handler(session,args):
    caller = session.getVariable("caller") # We use the variable we set previously in our Dia
    csv_reader = csv.reader(open("<filepath>","rb")) # <filepath> must be replace with the pa
    portfolio_list = []
    portfolio_list.extend(csv_reader)
    for data in portfolio_list:
        if (data[0] == caller): # We compare first column to the variable
            session.execute("set","effective_caller_id_name="+data[1]) # If equal, th
```

The CSV should look something like this :

```
"7001","Remote User"
"7002","Remote User2"
```

## FAQ

### Does each script spawn a python interpreter?

No. A single python interpreter is spawned at module startup and used for the lifetime of the freeswitch process.

### Are there thread safety issues?

Each thread swaps in its "thread state" before executing python code and then swaps it out when finished. Also during blocking calls into freeswitch, a thread will swap out its thread state in order to not block other threads, and then swap it in after the blocking call to freeswitch has finished.

### I changed a module I'm importing, and nothing happened

Answer: assume you are importing a module called baz, change your entry point module to:

```
import baz
reload(baz)
```

## How do I pass arguments to the script?

This is possible using channel variables. In the dialplan:

```
<extension name="foo">
  <condition field="destination_number" expression="^123$">
    <action application="set" data="foo=bar"/>
    <action application="python" data="mypackage.myscript"/>
  </condition>
</extension>
```

and in the python script:

```
foo = session.getVariable("foo")
console_log("info", "\n\nfoo: %s\n\n" % foo)
```

## Can I test scripts using python shell?

No, it will fail as follows when you try to import the python module:

```
>>> from freeswitch import *
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/usr/lib/python2.4/site-packages/freeswitch.py", line 7, in ?
    import _freeswitch
ImportError: No module named _freeswitch
```

## Can it serve configuration (like Lua)?

Yes, this has been added but not documented. The default hook for serving a config is `xml_fetch` as stated in the `python_example.py` script. However, the interpreter complains with a 'takes exactly 1 argument (2 given)' message if `xml_fetch` is defined to accept only one param. Altering the definition to accept 2 params solves the problem. However, `consoleLog` invariably shows `param2` to contain nothing...

```
from freeswitch import *

def xml_fetch( param1, param2 ):
    xml = '''
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
  <section name="dialplan" description="RE Dial Plan For FreeSWITCH">
    <context name="default">
      <extension name="generated">
        <condition field="destination_number" expression="^9992$">
          <action application="answer"/>
          <action application="playback" data="{hold_music}"/>
        </condition>
      </extension>
    </context>
  </section>
</document>
'''
    return xml
```

## Mod\_python

and you will need to create/edit a python.conf.xml to be something like:

```
<configuration name="python.conf" description="Python Configuration">
  <settings>
    <param name="xml-handler-script" value="mypackage.mymodule"/>
    <param name="xml-handler-bindings" value="dialplan"/>
  </settings>
</configuration>
```

## Gotchas

### String Substitution in Functions

The session function recordFile cannot contain anything but strings in the <filename> parameter. Attempting to use sql rows or other tuples as part of a string substitution will give you a

```
NotImplementedError: Wrong number of arguments for overloaded function 'CoreSession_recordFile'.
```

For example

```
record_file = row[0] + '-' + row[1] + '.wav'
session.recordFile(record_file,120,500,2)
```

will throw the error mentioned above even though when printing the variable record\_file will result in the expected value. Instead the values row[0] and row[1] need to be wrapped in the str() function when assigning a value to record\_file.

```
first_name = str(row[0])
last_name = str(row[1])
record_file = first_name + '-' + last_name + '.wav'
session.recordFile(record_file,120,500,2)
```

should give proper behavior.

## Troubleshooting

### Script fails but I don't see a stack trace

Are you connecting to freeswitch via fs\_cli? If so, that is probably why you don't see the stack trace. The stack trace (unfortunately) only shows up when you are on the actual freeswitch console, not when you are connected via sockets as is done with fs\_cli.

### Error:TypeError: CoreSession\_playAndGetDigits() takes exactly 9 arguments (10 given)

You need to update your scripts for [this change](#)

Can it serve configuration (like Lua)?

## Cannot import FreeSWITCH

Copy freeswitch.py from the python directory installed by freeswitch to /usr/lib/python2.5/site-packages

NOTE: if you are getting this error trying to test on the python shell .. you will never get past it. The only way to test python IVR scripts is to define the script in the dialplan and call the number. There is no way to currently do any testing with mod\_python scripts outside the context of an IVR running in freeswitch.

## Cannot import time

This indicates a problem where lib-dynload is pointing to the wrong place. Try printing out "sys.path" from your python script and look for a situation where you have /opt/freeswitch-trunk/lib/python2.5/site-packages and /usr/lib/python2.5/lib-dynload but **not** /opt/freeswitch-trunk/lib/python2.5/lib-dynload

### Workaround

Change your \$PATH environment variable so it sees the python installed by freeswitch before the system python, eg, PATH=/opt/fs/bin:\$PATH. [More info](#)

### Workaround #2

You can try [modifying the Makefile](#) so that it uses the python already on your system.

## CoreSession\_setDTMFCallback() takes exactly 4 arguments ..

If you see:

```
TypeError: CoreSession_setDTMFCallback() takes exactly 4 arguments (2 given)
```

Indicates you are running an older release and need to update if you want this to work.

## Message: 'module' object has no attribute 'EventConsumer\_node\_set'

If you see:

```
Message: 'module' object has no attribute 'EventConsumer_node_set'
```

You upgraded your freeswitch but obviously you forgot to update freeswitch.py within the /usr/lib/pythonX.Y/site-packages/ (X and Y are the python's version)

## CoreSession\_streamfile() takes exactly 3 arguments (4 given)

If you see

```
TypeError: CoreSession_streamfile() takes exactly 3 arguments (4 given)
```

## Mod\_python

it could mean that you are trying to use a dtmf callback that is a bound method of an object -- don't do that! The dtmf callback function should always be at the module scope and not take ("self") as an argument.

### Error calling DTMF callback - wrong # of arguments

You may be trying to use a bound instance method to a class. (eg, takes self as first argument). This will not work. Instead what you can do is to have a nested method for your dtmf handler that can access instance's **self** name.

### 'console\_log', argument 2 of type 'char \*'

If you see error messages:

```
TypeError: in method 'console_log', argument 2 of type 'char *'
```

You just need to call `str()` on the variable before passing to `console_log`, which cannot deal with Unicode strings at the present time.

### Channels are not being cleaned up

This should not happen, if it does please report a bug with detailed instructions on how to reproduce. This has surfaced and been fixed a few times.

### Avoid module-level global variables

If you find yourself using the **globals** keyword -- redesign your script. Concurrent calls will also be looking at the same variables, and things might not work as you expected. (These aren't "thread safety" issues, per se, since the Python GIL insures only one thread can run python code at any given time, but just be aware that multiple threads can see/access these variables).

### Build error: Python.h: No such file or directory

If you see

```
In file included from freeswitch_python.cpp:3:  
freeswitch_python.h:5:20: error: Python.h: No such file or directory
```

You need to install the `python-dev` package. You should also double-check the `src/mod/languages/mod_python/Makefile` to make sure it's using the version of python you are expecting. If not you can edit the Makefile manually.

**Build error: /usr/bin/ld: cannot find -lpython2.5**

This can happen due to missing symlinks. Try running "locate libpython", here is some example output:

```
# locate libpython
/usr/lib/libpython2.4.so
/usr/lib/libpython2.4.so.1
/usr/lib/libpython2.4.so.1.0
/usr/lib/libpython2.5.so.1
/usr/lib/libpython2.5.so.1.0
/usr/lib/python2.4/config/libpython2.4.a
/usr/lib/python2.4/config/libpython2.4-pic.a
/usr/lib/python2.4/config/libpython2.4.so
/usr/lib/python2.5/config/libpython2.5.a
```

Note that there is no libpython2.5.so -> libpython2.5.so.1 symlink. in /usr/lib or /usr/lib/python2.4/config/

**ImportError: ../datetime.so: undefined symbol: PyExc\_IOError**

This error and the Py\_ZeroStruct error (below) have been resolved as of subversion revision 11560. A build from the most recent code should resolve the problem.

If you see something like this on your freeswitch console:

```
freeswitch Traceback (most recent call last):
  File "/path/to/myscript.py", line 5, in module
    import datetime
ImportError: /usr/lib/python2.5/lib-dynload/datetime.so: undefined symbol: PyExc_IOError
```

~~FreeSWITCH does not use global name space in their shared objects. This has a side effect on modules who in turn try to load it's own shared objects because they too inherit the non-global namespace param.~~

**Solution:**

Add an attribute to the modues.conf to ask it to load with global name space

```
<load module="mod_foo" global="true"/>
```

**Workaround:**

This will also fix the problem, but is not the recommended approach.

```
export LD_PRELOAD=/usr/lib/libpython2.5.so
./freeswitch
```

**Dev Note:**

mod\_python should be modified to add the param SMODF\_GLOBAL\_SYMBOLS to the SWITCH\_MODULE\_DEFINITION macro (See mod\_cepstral at the top). Also, try changing the module definition to use global symbols the same way it is done in mod\_lua, see if that resolves the issue.

## **ImportError: /usr/lib/./datetime.so: undefinedsymbol: \_Py\_ZeroStruct**

This error, and the PyExec\_IOError have been fixed as of revision 11560 in subversion. A build from the latest code should resolve this problem.

## **Known Bugs**

### **Hangup hook + SQLAlchemy crashes switch**

It's not clear if it only happens in conjunction with SQLAlchemy, but removing the hangup hook definitely fixed the problem. Hangup hooks are buggy, please avoid or use with extreme caution.

### **Makefile deploys to /usr/lib/python2.4/site-packages even if python 2.5**

The configure script detects the latest python version installed on the system and generates the Makefile accordingly. However if it detects 2.5 it still tries to deploy the freeswitch.py to /usr/lib/python2.4/site-packages.

### **read() function might have problems**

See [mailing list thread](#). The bug caused concurrent calls to block eachother when calling the read() function.

Fixed in [svn 12958](#)

Description of fix: the first call to read() called a blocking freeswitch API call without releasing the GIL by calling begin\_allow\_threads(), and so basically everything else in the python VM is just blocked after that since it cannot not get the GIL. The fix was to add a call to begin\_allow\_threads() before calling the freeswitch API call, and its counterpart end\_allow\_threads() after the API call.

### **~~FIXED: Django database connections are not cleaned up~~**

When using Django with mod\_python, database connections are left open and will accumulate.

[UPDATE](#): Fixed in [revision 8193](#)

### **~~NEEDS RETESTING: python IVR's that execute nested python applications crash~~**

See [old mod python](#) for details, not retested after re-write

~~**NEEDS RETESTING:transferring to an extension that runs a python IVR hangs**~~

See [old mod python](#) for details, not retested after re-write

~~**NEEDS RETESTING:Using originate() and bridge() results in the audio only flowing one direction**~~

See [old mod python](#) for details, not retested after re-write

~~**NEEDS RETESTING:Strange errors on shutdown**~~

See [old mod python](#) for details, not retested after re-write

~~**NEEDS RETESTING:Unloading and reloading the module does not seem to work**~~

## See Also

Check [mod python dev](#) for more info.

- [Which scripting language should I use?](#)