

HowTo Write a Perl AGI script for Asterisk*

This is a quick and dirty blueprint for creating IVR (Interactive Voice Response) services using the Asterisk PBX and Perl. We will also demonstrate connecting to a backend MySQL database server.

Get to Know Linux...

All the components run on the Linux Operating System so some familiarity with this OS is definitely a plus.

Asterisk (www.asterisk.org) is a VOIP (Voice Over IP) software phone system.

Perl (www.perl.org) is a flexible, extensible and mature scripting language.

MySQL (www.mysql.com) is an ANSI standard SQL database server.

Linux(s) (www.redhat.com, www.novell.com, www.ubuntu.com, etc.)

Test System Recommendations...

While you can obtain Asterisk itself from Digium, Asterisk based systems and projects from 3rd parties, or roll your own from scratch – I recommend Trixbox (www.trixbox.org) from Fonality. It comes as a downloadable ISO that includes an installer for CentOS linux + Trixbox. All you do is burn a CD, boot from the CD and install. Note this will completely wipe the hard disk during the installation, use a test system!

Trixbox has the advantage of having pre-installed, guaranteed to work, extensions beyond the basic Asterisk load capabilities. Text to speech is there and working, the AGI interface and Perl are already configured for basic operation. You can always build your own system from scratch for deployment (or maybe remove the components you don't need from a stock Trixbox setup?).

If you need to test with actual phone lines you will need some PCI cards to interface with the PSTN (Public Switched Telephone Network). Go hit froogle.com and search for “digium POTS”. Digium is the company that develops Asterisk and makes cards that work with it. There are competitors of course but you can decide their merits. It is about \$100 a port and you can get two kinds of ports:

FXO = Connect me to the Phone Company

FXS = Connect me to a phone

Another reason to use Trixbox is it comes with a pre-configured dial plan. That means you don't have to know a whole bunch about telephony to get started. You do have to configure extensions (add SIP phones) so thank goodness that the dial plan is off your plate – for now.

Setup Your Asterisk Server... Go On Now...

Other than my helpful tips above... You need to have your Asterisk server up and running before continuing. Do it without me. That is not in the scope of this document. Etc.

Flowchart your Application

The method described herein puts the entire application in Perl, which has some portability advantages. You can mix the amount of application that resides in the dial plan and in the AGI script but I won't start you on that road here. Account for everything the caller will hear, all twelve keys at each menu, and the callers patience.

AGI Implementation Notes

AGI interface is via standard IN/OUT, so you can use any language that supports it: perl, php, python, java, C, C++, C#, advanced logo...

At script startup time Asterisk sends various pieces of information to your script and you should read in these items, via standard input, before doing much else.

Each item is sent on a line terminated with a newline and the end of the list is indicated by an empty line. Discard any ARGs you don't need. The list of items received will look something like Table 1 on the right:

```
agi_request: kiosk.pl
agi_channel: Zap/1-1
agi_language: en
agi_type: Zap
agi_callerid:
agi_dnid:
agi_context: default
agi_extension: 3
agi_priority: 1
```

Table 1: Standard ARGs

We will be, as advertised, concentrating on the use of Perl and the **Asterisk::AGI** module. Here is a list of the commands available in the latest version.

(*asterisk-perl-0.09 - 13 Nov 2006*)

[ANSWER](#)
[AUTOHANGUP <time>](#)
[CHANNEL STATUS \[<channelname>\]](#)
[EXEC <application> <options>](#)
[GET DATA <filename> \[<timeout>\] \[<max digits>\]](#)
[GET VARIABLE <variablename>](#)
[HANGUP \[<channelname>\]](#)
[RECEIVE CHAR <timeout>](#)
[RECORD FILE <filename> <format> <escape digits> <timeout> \[BEEP\]](#)
[SAY DIGITS <digit string> <escape digits>](#)
[SAY NUMBER <number> <escape digits>](#)
[SEND IMAGE <image>](#)
[SEND TEXT "<text to send>"](#)
[SET CALLERID <number>](#)
[SET CONTEXT <desired context>](#)
[SET EXTENSION <new extension>](#)
[SET PRIORITY <new priority number>](#)
[SET VARIABLE <variablename> <value>](#)
[STREAM FILE <filename> <escape digits>](#)
[TDD MODE <on|off>](#)
[VERBOSE <level>](#)
[WAIT FOR DIGIT <timeout>](#)

Go Code! And some Links in case you get stuck...

<http://www.asteriskpbx.com/> (For Developers)

<http://www.asterisknow.org/> (For Users)

<http://www.trixbox.org> (For All)

[John Todd's Excellent Asterisk/VOIP Primer](#)

(John Fields') Asterisk AGI App Best Practice

This illustrates the execution path, and the failure mode, of the AGI application.

For example, if the AGI script fails to execute – like if the database connection cannot be established, or you just saved it with a typo – the caller will at the VERY least hear “Goodbye”.

If you are using Trixbox add the following lines to /etc/asterisk/extension-custom.conf.

If you have rolled your own, this needs to be in extension.conf where ever it lives.

Extension 1300 below just happens to fit my installation. Change as needed.

```
exten => 1300,1,Goto(custom-kiosk,s,1)
```

```
[custom-kiosk]
```

```
exten => s,1,Answer
```

```
exten => s,2,AGI(kiosk.pl|${CALLERID(name)}|${CALLERID(number)})
```

```
exten => s,3,Playback(kiosk/gbye)
```

```
exten => s,4,Hangup
```

