

* &#!perl

Using Perl with Asterisk

What is Asterisk?

- An open source PBX / IVR in software
- Supports major VoIP protocols
- Interoperates with standards-based telephony eq.
- Connects disparate technologies together
- Runs on Linux using commodity x86 hardware

Major Components

Media Channels

- SIP, H.323, Skinny, IAX, etc.

Media Channel Configuration Files

- Contains all channel specific options
- Defines source and destination for media
- Defines real names for use in dialplan
- Declares default dialplan context for channel

Major Components

Dialplan

- Comprised of labels, referred to as a "context"
- Contexts contain a list of extension definitions
- Each extension specifies a priority, and a command to run; similar to BASIC

SIP Media Channel

```
[simon]           ; Dialplan channel name
context=local-phones ; Dialplan Context (outbound)
secret=abc123      ; Password
type=friend        ; Will send and recv. calls
host=dynamic       ; Has dynamic IP
dtmfmode=info      ; Touchtone digits sent oob
disallow=all       ;
allow=ulaw          ; ulaw audio codec is okay
canreinvite=no     ; audio must pass thru server
```

Simple Dialplan

[example]

```
exten => <extension>,<priority>,<application>(options)
exten => <extension>,<priority>,<application>(options)
exten => <extension>,<priority>,<application>(options)
```

[local-phones]

```
exten => s,1,Wait(1)
exten => s,2,Answer
exten => s,3,Background(simon-hello_please_dial)
exten => s,4,WaitExten()
exten => s,5,Goto(local-phones,s,3)

exten => 123,1,Dial(SIP/simon)
exten => 123,2,Voicemail(u123)
```

Slight Improvement

[example]

```
exten => <extension>,1,<application>(options)
```

```
exten => <extension>,n,<application>(options)
```

```
exten => <extension>,n,<application>(options)
```

[local-phones]

```
exten => s,1,Wait(1)
```

```
exten => s,2,Answer
```

```
exten => s,3,Background(simon-hello_please_dial)
```

```
exten => s,n,WaitExten()
```

```
exten => s,n,Goto(local-phones,s,3)
```

```
exten => 123,1,Dial(SIP/simon)
```

```
exten => 123,n,Voicemail(u123)
```

Similar to Basic

```
<priority> <command> <arguments>  
<priority> <command> <arguments>  
<priority> <command> <arguments>
```

```
10 PRINT "Enter a number, zero to stop:";  
20 INPUT A  
30 IF A = 0 THEN GOTO 70  
40 LET A = A + 10  
50 PRINT "The number plus 10 is "; A  
60 GOTO 1  
70 STOP
```

Adding some Magick

[example]

```
exten => <extension>,1,<application>(options)
exten => <extension>,n,<application>(options)
exten => <extension>,n,<application>(options)
```

[local-phones]

```
exten => s,1,Wait(1)
exten => s,n,Answer
exten => s,n,AGI(onstar-demo.agi)
exten => s,3,Background(onstar-greeting)
exten => s,n,WaitExten()
exten => s,n,Goto(local-phones,s,3)

exten => 123,1,Dial(SIP/simon)
exten => 123,n,Voicemail(u123)
```

A simple AGI

```
#!/usr/bin/perl -w
```

```
use Asterisk::AGI;  
$AGI = new Asterisk::AGI;
```

```
my %input = $AGI->ReadParse();
```

```
my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday) =  
localtime(time);
```

```
$AGI->say_number( $hour );  
$AGI->say_number( $min );
```

AGI Methods

Channel Control

```
$AGI->answer()  
$AGI->hangup($channel)
```

Communicating with the User

```
$AGI->say_number($number);  
$AGI->say_number($number, $escape_digits)  
$AGI->say_digits($number, $escape_digits)  
$AGI->stream_file($filename, $received_digit)  
$AGI->get_data($filename, $timeout, $maxdigits)
```

AGI Methods

Dialplan Interaction

```
$AGI->exec($app, $options)  
$AGI->set_context($context)  
$AGI->set_extension($extension)  
$AGI->set_priority($priority)  
$AGI->set_callerid($number)
```

Getting/Setting data in the dialplan

```
$AGI->set_variable($variable, $value)  
$AGI->get_variable($variable)
```

An interactive AGI

```
#!/usr/bin/perl
```

```
use Asterisk::AGI;  
$AGI = new Asterisk::AGI;
```

```
my %input = $AGI->ReadParse();
```

```
$digits = $AGI->get_data( "onstar-demo2", 3000, 20 );  
$AGI->stream_file( "onstar-youpressed", undef );  
$AGI->say_digits( $digits );
```

AGI Hacks

```
#!/usr/bin/perl
```

```
use Asterisk::AGI;  
$AGI = new Asterisk::AGI;
```

```
my %input = $AGI->ReadParse();
```

```
`lynx -source http://weather.uc.org > forecast.txt`;  
`text2ave -F 800 -o forecast.wav forecast.txt`;  
$AGI->stream_file('forecast.wav');
```

Scaleability

Problems with the CGI model similar to CGI

- Every request requires own perl interpreter
- Script startup time is high for realtime applications

Scaleability

Solutions:

- Imbed Perl in Asterisk with res_perl

```
exten => 1,1,Perl(some_func:arg1:arg2)
```

- Connect to an AGI server over the network using FastAGI

```
exten => 1,1,AGI(agi://192.168.1.1)
```