Asterisk - Advanced Configuration

PacNOG 3 VoIP Workshop
June 2007, Cook Islands

Jonny Martin
jonny@jonnynet.net
Variable Expressions

- Variables used to
  - reduce configuration complexity
  - add clarity
  - provide additional dialplan logic

- Basic expressions allow us to perform basic mathematical calculations

  exten => 501,1,Set(Count=1)
  exten => 501,2,Set(Newcount=${Count}+1)
  exten => 501,3,SayNumber(${NewCount})
Substrings

- ${variable:offset:length}
- Returns the substring of ‘variable’ of length ‘length’, starting at offset
- Commonly used to strip access codes
  - exten => 1X.,1,Dial(SIP/${EXTEN:1})
  - Dials the extension minus the initial ‘1’
  - If ‘length’ is omitted, the rest of the string is returned
- To concatenate two strings, simply write them together:
  - ${string1}${string2}
Variable Operators

- Boolean operators (non-zero = true, zero=false)
  - Or - var1 | var2
  - And - var1 & var2
  - Comparisons - var1 {=, >, >=, <, <=, !} var2

- Mathematical operators
  - Addition and subtraction - var1 {+, -} var2
  - Multiplication, integer division, remainder - var1 {* , /, %} var2
Dialplan Functions

• Basic syntax:
  
  • FUNCTION_NAME(argument)

• To reference the value of a function
  
  • ${FUNCTION_NAME(argument)}

• can be nested, i.e. ‘argument’ above replaced with another function reference

• Used for string manipulation
Dialplan Functions

- `exten => 502,1,Set(TEST=example)`
  - `exten => 502,2,SayNumber(${LEN(${TEST})})`)

  - `Len()` returns the length of a string

- Many more...
Functions

*CLI> show functions
Installed Custom Functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URIENCODE</td>
<td>URIENTCODE(&lt;data&gt;) Encodes a string to URI-safe encoding.</td>
</tr>
<tr>
<td>URIDECODE</td>
<td>URIDECODE(&lt;data&gt;) Decodes an URI-encoded string.</td>
</tr>
<tr>
<td>SQL_ESC</td>
<td>SQL_ESC(&lt;string&gt;) Escapes single ticks for use in SQL statements.</td>
</tr>
<tr>
<td>ODBC_PRESENCE</td>
<td>ODBC_PRESENCE(&lt;arg1&gt;[...[,&lt;argN&gt;]]) Runs the referenced query with the specified arguments</td>
</tr>
<tr>
<td>ODBC_ANTIGF</td>
<td>ODBC_ANTIGF(&lt;arg1&gt;[...[,&lt;argN&gt;]]) Runs the referenced query with the specified arguments</td>
</tr>
<tr>
<td>ODBC_SQL</td>
<td>ODBC_SQL(&lt;arg1&gt;[...[,&lt;argN&gt;]]) Runs the referenced query with the specified arguments</td>
</tr>
<tr>
<td>TXTCIDNAME</td>
<td>TXTCIDNAME(&lt;number&gt;) TXTCIDNAME looks up a caller name via DNS.</td>
</tr>
<tr>
<td>ENUMLOOKUP</td>
<td>ENUMLOOKUP(number[,Method-type[,opt ENUMLOOKUP allows for general or specific querying of NAPTR records or counts of NAPTR types for ENUM or ENUM-like DNS pointers.</td>
</tr>
<tr>
<td>CALLERID</td>
<td>CALLERID(datatype) Gets or sets Caller*ID data on the channel.</td>
</tr>
<tr>
<td>ARRAY</td>
<td>ARRAY(var1[,var2[...[,varN]]]) Allows setting multiple</td>
</tr>
</tbody>
</table>
Asterisk Database

- astdb - simple database forms part of Asterisk
- Dial plan and CLI can insert and remove data
- Data stored in a file, so is retained across Asterisk reloads and server reboots
- Data stored in groupings of families containing keys
  - `exten => s,1,Set(DB(family/key)=${some_variable})`
  - `exten => s,1,Set(DB(system/nightmode_on)=1)`
  - `exten => s,1,Dial(${DB(exten/${EXTEN}/dial_string)},15)`
Asterisk Database - Example

; start counting and store count progress in astdb

; check if DB key exists, if not, jump to key_no_exist
; function DB_Exists returns 1 if the key exists, 0 if not
exten => 30,1,GotoIf(DB_Exists(test/count)?key_no_exist)

; begin the counting!
exten => 30,n(start),Set(COUNT=${DB(test/count)})
exten => 30,n,SayNumber(${COUNT})
exten => 30,n,Set(COUNT=$[${COUNT} + 1])
; update the DB
exten => 30,n,Set(DB(test/count)=${COUNT})
exten => 30,n,Goto(start)

; if we got here it is because the key didn’t exist in the DB
; create the key
exten => 30,n(key_no_exist),Set(DB(test/count)=1)
; and jump back to the start to begin counting
exten => 30,n,Goto(start)
GotoIf

;   GotoIf(condition?label1[:label2])
;
; Go to label1 if condition is true or to next step (or label2 if defined) if
condition is false, or
;
;   GotoIf(condition?[label1]:label2)
;
; Go to next step (or label1 if defined) if condition is true or to label2 if
condition is false.
Macros

• Avoids repetition in the dial plan

• Akin to building a function in the dial plan

• Useful for building standard phone dialling logic

• Uses extra specific channel variables:

  ${ARGn}: The nth argument passed to the macro
  ${MACRO_CONTEXT}: Context of the extension that triggered this macro
  ${MACRO_EXTEN}: The extension that triggered this macro
  ${MACRO_PRIORITY}: The priority in the extension where this macro was triggered
Macro Example

[macro-stdexten]

; Standard extension macro:
;   ${ARG1} - Extension (we could have used ${MACRO_EXTEN} here as well
;   ${ARG2} - Device(s) to ring
;
; ring the interface for 20sec max
exten => s,1,Dial(${ARG2},20)
; jump based on status (NOANSWER,BUSY,CHANUNAVAIL,CONGESTION,ANSWER)
exten => s,2,Goto(s-{$DIALSTATUS},1)

exten => s-NOANSWER,1,Voicemail(u${ARG1}) ; If unavailable, send to voicemail
exten => s-NOANSWER,2,Goto(default,s,1)   ; If they press #, return to start

exten => s-BUSY,1,Voicemail(b${ARG1})     ; If busy, send to voicemail w/ busy announce
exten => s-BUSY,2,Goto(default,s,1)       ; If they press #, return to start

exten => _s-.,1,Goto(s-NOANSWER,1)        ; Treat anything else as no answer
exten => a,1,VoicemailMain(${ARG1})       ; If they press *, send to VoicemailMain
AGI Scripts

- Asterisk Gateway Interface

- Dial plan can call Perl, Python, PHP scripts

- AGI script reads from STDIN to get information from Asterisk

- AGI script writes data to STDOUT to send information to Asterisk

- AGI script can write to STDERR to send debug information to the console

- Scripts stored in /usr/share/asterisk/agi-bin/ on Debian

- exten => 520,1,AGI(/path/to/agi-script.agi)
AGI Scripts

• Very very powerful

• A2Billing uses them to implement a complete billing system
  • All the relevant call data is sent to the AGI
  • MySQL lookups performed
  • Relevant dial command returned to Asterisk
  • Database updated at end of call
Agents

• Users can log in as an Agent

• Maps current extension to that user’s Agent

• Agent can then be logged into queues

• Agents can log in / out at will, follow-me functionality

• Agents functionality still quite buggy - best not to use for anything complex
agents.conf

/etc/asterisk/agents.conf

[general]
; Define whether callbacklogins should be stored in astdb for persistence
persistentagents=yes

[agents]
;autologoff=15 ; time (s) before agent auto logoff if no answer
;ackcall=no
wrapuptime=1000
;musiconhold => default
;updatecdr=no
; Enable recording calls addressed to agents. It's turned off by default.
recordagentcalls=yes
;recordformat=gsm

; agent => agentid,agentpassword,name
agent => 600,1234,Lilly

group=2 ; Senior NOC staff
agent => 610,1234,Steve
Queues

• Reasonably powerful queuing support within Asterisk

• Queues can have static or dynamic members

• Members can be channels, or Agents

• Automatic distribution of calls based on queue strategy
/etc/asterisk/queues.conf

[general]
; Store each dynamic agent in each queue in the astdb for persistence
persistentmembers = yes

; Queue(queueName|[options]|[optionalurl]|[announceoverride]|[timeout])
; example: Queue(dave|t|||45)

[noc]
musiconhold = default
strategy = ringall ; ringall, roundrobin, leastrecent, fewest calls, random, rrmemory

servicelevel = 30 ; SLA setting (s). stats for calls answered in this time
timeout=15 ; How long the phone rings before it's considered a timeout
retry=0 ; How long do we wait before trying all the members again?
; Weight of queue - when compared to other queues, higher weights get preference
weight=2
wrapuptime=5 ; how long before sending agent another call
maxlen = 0 ; of queue, 0 for no maximum

; How often to announce queue position and/or estimated holdtime to caller (0=off)
announce-frequency = 0
;announce-holdtime = yes|no|once
;announce-round-seconds = 10
; How often to make any periodic announcement (see periodic-announce)
;periodic-announce-frequency=60
Queuing Example

; Using Agents
; agent login to helpdesk queue
exten =&gt; *4,1,Answer()
exten =&gt; *4,n,AddQueueMember(noc|Agent/\${CALLERID(NUM)})
exten =&gt; *4,n,AgentCallbackLogin($\{CALLERID(NUM)\}|q$\{CALLERID(NUM)\}@sip)
exten =&gt; *4,n,Hangup()

; agent logout from noc queue
; note # is sent through by as a %23 in some sip headers
; so may need to repeat with exten =&gt; %23
exten =&gt; #4,1,Answer()
; send trigger to flash panel
exten =&gt; #4,n,System(/usr/sbin/asterisk -rx "agent logoff Agent/\${CALLERID(NUM)}")
exten =&gt; #4,n,RemoveQueueMember(noc|Agent/\${CALLERID(NUM)})
exten =&gt; #4,n,Playback(agent-loggedoff)
exten =&gt; #4,n,Hangup

; Or, using dynamic login of channel instead of agents, doesn't send triggers to flash panel
exten =&gt; *4,1,Answer()
exten =&gt; *4,n,AddQueueMember(noc\${CALLERID(NUM)})
exten =&gt; *4,n,Playback(logged-in)
exten =&gt; *4,n,Hangup()

exten =&gt; #4,n,RemoveQueueMember(noc\${CALLERID(NUM)})
exten =&gt; #4,n,Playback(agent-loggedoff)
exten =&gt; #4,n,Hangup
Festival

- Festival - Open sources text to speech engine
  - http://www.cstr.ed.ac.uk/projects/festival/
- Text to speech is a bit rough, but useable
- Easy to use once installed
- Useful for putting together quick IVRs

```plaintext
exten => 1,1,Festival('Record your message now')
exten => 1,n,Record(filename:alaw)
exten => 1,n,Festival('You recorded')
exten => 1,n,Playback(filename)
exten => 1,n,Festival('message saved.')
exten => 1,n,Goto(s,1)
```
Lab 3: Advanced Asterisk Configuration
Asterisk CLI

• Should be quite familiar with it by now

• Can run remote Asterisk CLI commands from server
  
  • asterisk -rx “sip reload”

• Primarily useful for triggering reloads and setting DB keys
Asterisk Manager API

- Allows client programs to connect to Asterisk
  - Issues commands and reads events
  - Used by Flash Operator Panel to keep track of Asterisk’s state
- Telnet to the listening TCP/IP port (5038 by default)
  - Login checked against credentials in manager.conf
  - Specific message types subscribed to in manager.conf
## Asterisk Manager API Commands

<table>
<thead>
<tr>
<th>Action</th>
<th>Privilege</th>
<th>Synopsis</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsoluteTimeout</td>
<td>call,all</td>
<td>Set Absolute Timeout</td>
</tr>
<tr>
<td>AgentCallbackLo</td>
<td>agent,all</td>
<td>Sets an agent as logged in by callback</td>
</tr>
<tr>
<td>AgentLogoff</td>
<td>agent,all</td>
<td>Sets an agent as no longer logged in</td>
</tr>
<tr>
<td>Agents</td>
<td>agent,all</td>
<td>Lists agents and their status</td>
</tr>
<tr>
<td>ChangeMonitor</td>
<td>call,all</td>
<td>Change monitoring filename of a channel</td>
</tr>
<tr>
<td>Command</td>
<td>command,all</td>
<td>Execute Asterisk CLI Command</td>
</tr>
<tr>
<td>DBGet</td>
<td>system,all</td>
<td>Get DB Entry</td>
</tr>
<tr>
<td>DBPut</td>
<td>system,all</td>
<td>Put DB Entry</td>
</tr>
<tr>
<td>Events</td>
<td>&lt;none&gt;</td>
<td>Control Event Flow</td>
</tr>
<tr>
<td>ExtensionState</td>
<td>call,all</td>
<td>Check Extension Status</td>
</tr>
<tr>
<td>Getvar</td>
<td>call,all</td>
<td>Gets a Channel Variable</td>
</tr>
<tr>
<td>Hangup</td>
<td>call,all</td>
<td>Hangup Channel</td>
</tr>
<tr>
<td>IAXnetstats</td>
<td>&lt;none&gt;</td>
<td>Show IAX Netstats</td>
</tr>
<tr>
<td>IAXpeers</td>
<td>&lt;none&gt;</td>
<td>List IAX Peers</td>
</tr>
<tr>
<td>ListCommands</td>
<td>&lt;none&gt;</td>
<td>List available manager commands</td>
</tr>
<tr>
<td>Logoff</td>
<td>&lt;none&gt;</td>
<td>Logoff Manager</td>
</tr>
<tr>
<td>MailboxCount</td>
<td>call,all</td>
<td>Check Mailbox Message Count</td>
</tr>
<tr>
<td>MailboxStatus</td>
<td>call,all</td>
<td>Check Mailbox</td>
</tr>
<tr>
<td>Monitor</td>
<td>call,all</td>
<td>Monitor a channel</td>
</tr>
<tr>
<td>Originate</td>
<td>call,all</td>
<td>Originate Call</td>
</tr>
<tr>
<td>ParkedCalls</td>
<td>&lt;none&gt;</td>
<td>List parked calls</td>
</tr>
</tbody>
</table>
Asterisk Performance

• Performance heavily dependant on what your Asterisk server is doing

• ‘Switching’ calls - can easily get up to ~200 calls/sec

• Terminating media streams - around 30 simultaneous calls on a fast server

• Codecs - low bitrate codecs typically require a lot of CPU

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Number of channels</th>
<th>Minimum recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hobby system</td>
<td>No more than 5</td>
<td>400-MHz x86, 256 MB RAM</td>
</tr>
<tr>
<td>SOHO system</td>
<td>5 to 10</td>
<td>1-GHz x86, 512 MB RAM</td>
</tr>
<tr>
<td>Small business system</td>
<td>Up to 15</td>
<td>3-GHz x86, 1 GB RAM</td>
</tr>
<tr>
<td>Medium to large system</td>
<td>More than 15</td>
<td>Dual CPUs, possibly also multiple servers in a distributed architecture</td>
</tr>
</tbody>
</table>