HEISON CHAK

more Asterisk tricks

Heison Chak is a system and network administrator at SOMA Networks. He focuses on network management and performance analysis of data and voice networks. Heison has been an active member of the Asterisk community since 2003.

■ *heison@chak.ca*

Asterisk provides an integrated platform for many voice applications, ranging from a simple voice gateway or conference server to a full-blown PBX. The increase in Asterisk-compatible hardware being shipped by vendors will enable rapid development and deployment of voice applications to meet various needs.

IN THE FEBRUARY ISSUE OF *;LOGIN:*, a number of interesting Asterisk topics based on the dialplan and AGI (Asterisk Gateway Interface) were revealed. The dialplan (as described in extensions.conf) is the heart of Asterisk; it uses exten => and [context] to route calls, which integrates PBX (private branch exchange), IVR (interactive voice response), and external applications. This article will start off with another dialplan trick for a home office, followed by a number of effective Caller ID applications using Asterisk.

It has always been a challenge to run a PBX system in a home office environment—family members are not used to extensions and may dislike the fact that they cannot pick up just any handset when a phone rings. On the other hand, most do appreciate the privacy they get with individual extensions. Those trying to ring the house number often consider the IVR and the need to remember extension numbers as troublesome and unfriendly. To get around this, we have set up a whitelisting for people who call us frequently. If the telephone number of the caller matches an entry on the whitelist, it will ring all handsets instead of playing the IVR greeting.

```
HEISON=SIP/cisco1
CLARA=SIP/cisco2
KITCHEN=Zap/4
BEDROOM=IAX2/iaxy@iaxy
FAMILYROOM=SIP/grandstream1

HEISON_EXT=${HEISON}&${KITCHEN}&${BEDROOM}
CLARA_EXT=${CLARA}&${FAMILYROOM}&${KITCHEN}&${BEDROOM}
ALL_EXT=${HEISON}&${CLARA}&${KITCHEN}&${BEDROOM}&${FAMILYROOM}
;
; Whitelist—family members
;
exten => s,7,GotoIf($[${CALLERIDNUM} = 4161230123]?50:8) ; Dad's cell
exten => s,8,GotoIf($[${CALLERIDNUM} = 4161230124]?50:60) ; Mom's cell
;
; Calls from family rings all handsets
;
exten => s,50,Dial(${ALL_EXT},20,Tr)
;
; IVR starts here
;
exten => s,60,Background(home-greeting)
```

This scheme significantly reduces the number of complaints, while restoring a number of friendships, since it bypasses the IVR altogether for frequent callers on the whitelist. However, a caller listening to the IVR greeting can easily mistake the voice prompt for an

answering machine, because that is what most people are used to. If the caller simply hangs up, the call info is lost unless someone examines the CDR (call detail record).

Caller ID

Caller Line Identification, CLI, Caller ID, Caller Display, CID—all are different names for the service that allows one to see who is calling. Caller ID carries information about the caller, such as telephone number, name, and the network timestamp. (Note: a caller may opt to not deliver his/her identity on outgoing calls.) People often call themselves to generate CallerID in order to set the time on telephone handsets that have been moved or power-cycled.

With Asterisk allowing logic to extend beyond the dialplan, Caller ID info can easily be made available to TiVo, MythTV, any Windows PC, or Instant Messenger.

Announcement

There are off-the-shelf products that perform simple TTS (text to speech) on the incoming telephone number. They announce the caller's telephone number through built-in speakers after receiving Caller ID information. Some people find it irritating, because the device may take a few seconds to read back the 10 or 11-digit telephone number. Others find it useful, as they can be anywhere in the house and listen to the announcement before answering the call. Using the AGI and a TTS program (e.g., Festival or Cepstral), one can easily report the incoming telephone number via the system's sound card.

MythTV

The Caller ID announcement may be disturbing to people watching TV or a movie. MythTV is a suite of programs that provides a platform for watching video (e.g., TV, DVD), listening to music (e.g., CD, mp3), Web browsing, etc. Thanks to the thoughtful design of MythTV, the OSD (on-screen display) subsystem and predefined XML tags for Caller ID provide hooks in delivering information about your caller onto your television set:

```
exten => s,1,Wait(1)
exten => s,2,System(/usr/bin/mythtvosd
    —template=../programs/mythtvosd/cid.xml
    —caller_name="${CALLERIDNAME}"
    —caller_number="${CALLERIDNUM}"
    —caller_date="${DATETIME}"
exten => s,3,Answer
```

Note that the MythTV OSD can be run from a different host to your MythTV server, where you may have tuner cards installed. The multicast nature of MythTV OSD enables Caller ID to be distributed to all MythTV front ends.

As the popularity of IM (instant messaging) increases, more and more people run multi-protocol IM clients on their machine. Using AGI and Class.Jabber.php, Asterisk can send messages containing Caller ID information to an IM client supporting the Jabber protocol:

```
exten => s,1,Wait(1)
exten => s,2,AGI(jabber.php)
exten => s,3,Answer
```

jabber.php:

```
#!/usr/bin/php
```

```
<?php
ini_set('display_errors', 0 );
ini_set('include_path', '.:/usr/share/pear');
require_once('class.jabber.php');
$stdin = fopen('php://stdin', 'r');
$stdout = fopen('php://stdout', 'w');
$stdlog = fopen('my_agi.log', 'w');

while (!feof($stdin)) {
    $temp = fgets($stdin);
    $temp = str_replace("\n","",$temp);
    $s = explode(":",$temp);
    $agivar[$s[0]] = trim($s[1]);
    if (($temp == "") || ($temp == "\n")) {
        break;
        }
}

$callerid=$agivar[agi_callerid];

$JABBER = new Jabber;

$JABBER->server = "jabber.org";
$JABBER->port = 5222;
$JABBER->username = "asterisk";
$JABBER->password = "MyPassword";
$JABBER->resource = "ClassJabberPHP";

$JABBER->enable_logging = TRUE;

$JABBER->Connect() or die("Couldn't connect!");
$JABBER->SendAuth() or die("Couldn't authenticate!");

$JABBER->SendPresence();

$JABBER->SendMessage("heison@jabber.org",
    "normal",
    NULL,
    array( // body, thread... whatever
        'body' => 'Incoming call from '.$callerid ),
        NULL
        );

$JABBER->Disconnect();
exit;
?>
```

The biggest drawback to sending Caller ID info to an IM server is lag; Asterisk waits for the AGI script to complete execution before answering the call. The Asterisk dialplan was not designed to run commands in parallel. One possible work around is to use System(jabber.php&) and run the process in the background. Of course, a better solution in this case would be to run a local Jabber server on the same network to reduce the roundtrip time.

Windows PC

In case you haven't noticed, all the above examples make Caller ID available before answering the call. This brings the ability of knowing the caller's identity even before he/she gets to the IVR prompt in Asterisk. However, not everyone uses IM, and only a small percentage of people have a TiVo-like system—and some may dislike the announcement feature. To support Windows users, one might choose to deliver Caller ID information through Samba messaging or with third-party programs, such as YAC (Yet Another Caller ID program).

With the help of System() and netcat, Asterisk can push the values to a Windows machine (e.g., 10.155.1.9) listening on TCP port 10629:

```
exten => s,1,Wait(1)
exten => s,2,System(/bin/echo -n -e "${CALLERIDNAME} ${CALLERID-
NUM}'" | nc -w 1 10.155.1.9 10629)
exten => s,3,Answer
```

As soon as the TCP connection tears down (set by -w timeout_value), YAC will display the information received on the system icon tray.

Fix CIDName

Now that we know how to make Caller ID available by a number of different means, we have a fundamental problem that needs to be discussed. Having access to the caller's telephone number may be useful if you recognize the number. We often examine the Caller ID name sent by the telco, which may not be reliable.

With the PGSQL() command built into Asterisk, one can do several SQLy things, such as querying and modifying the CALLERIDNAME based on the incoming CALLERIDNUM:

```
exten => s,1,Wait,1
exten => s,2,Macro(fixcidname)
exten => s,3,Answer

[macro-fixcidname]
exten => s,1,GotoIF($[ ${LEN(${CALLERIDNUM})} = 0]?8 )
exten => s,2,PGSQL(Connect id host=... port=... password=... user=...
      dbname=...)
exten => s,3,PGSQL(Query rs ${id} SELECT fname\, lname FROM table
      WHERE number =\'${CALLERIDNUM}\')
exten => s,4,PGSQL(Fetch status ${rs} fname lname)
exten => s,5,SetCIDName(${fname} ${lname})
exten => s,6,PGSQL(Clear ${rs})
exten => s,7,PGSQL(Disconnect ${id})
exten => s,8,NoOp(${CALLERIDNAME} ${CALLERIDNUM})
```

Conclusion

The examples above demonstrate how you can use Asterisk to enhance the usefulness of the traditional Caller ID. They are all based on North American on-hook CIDs. If you look into Caller ID for call waiting or international calls, you will find that they are quite different. The Caller ID FAQ contains useful resources on the topic (http://www.ainslie.org.uk/callerid.htm).

OTHER USEFUL URLS

MythTV: http://www.mythtv.org

YAC: http://unflowerhead.com/software/yac