Elastix queue call-back tutorial with WombatDialer

You are called in to a client site; they seem to have a problem. They run a small (10 agents) inbound call centre, and when you join everybody else in the meeting room, there is a large and colourful graph in the middle of the table. The graph shows the call wait times during the week and boy, it's not a good sight. Their main inbound activity is to offer client support for a company selling sport bikes, and everybody seems to be calling on Monday morning. It looks like people go riding on weekends and whatever problem they have, they call on Monday morning. Wait times peak, abandon rates spike, and nobody is happy. The call center manager is mostly concerned of having to hire and train some temp people in order to handle the load that only happens one day a week. They ask you if you have any better idea on what can be done. And yes, you have some.

You can program an Asterisk queue so that when people tire of waiting, they press a digit and get to a menu where they can leave their number. Then the system queues their call and attempts to call them at a convenient time. This way:

. your customers are happy; they don't have to wait in queue for so long.

. your call center manager is happy twice: the first time because wait times and abandon rates go down, the second one because by placing calls at a convenient time they can smooth out the workload of their agents during the day

This scenario requires some additional "glue" to what is basically supported by Asterisk - exiting a queue and reading a number are easy, but then starts the pain. You'll have to create a database and write a script that reads back from it. You have to handle invalid numbers, busy numbers and the like (if we promised to call back the client, we cannot just try once and forget about it). You'll have to have a GUI of some kind for the manager to start and stop dialing. You'll have to adapt to the number of available agents. You'll have to report on this activities. You'll have to avoid flooding the trunks of your PBX with too many calls. In short, it's the kind of thing that gets more complex the more you think about it. That's what WombatDialer is for.

What we plan to do is to use WombatDialer as the call-back engine. It can be controlled by an external HTTP API, so you can do that from the Asterisk dial-plan. It has a definite topology and call back rules, so you get the number of calls you expect on one or more Asterisk servers. It can work with an existing PBX and does not interfere with calls that are not its own. It keeps track of call completions and knows what to do in case of invalid and busy numbers. It has reports of its own and can work with QueueMetrics for

powerful and detailed reports.

The client uses Elastix as PBX system, so we'll have to integrate it with WombatDialer. No problem!

So what we do is:

. First we create a normal queue, for inbound. We call it "400".

. Then we create a call-back queue. If our main queue is called "400", then let's call this second queue "401". The idea is that WD will monitor this queue - when you have members on this queue, then WD will start placing calls. This way an inbound call-center with multiple queues will find it very natural to have some agents join and leave a call-back queue. When you creet this queue, make sure you set "Ring Strategy: rrmemory", "Event When Called: Yes", "Member Status: Yes", "Autofill: yes" so that WD can use it effectively.

. we create a piece of dialplan that will handle the exits from queue "400" and will gather the telephone number

. we create a new "custom extension" (399) that will jump in the dialplan at "Local/1@queue-leavenumber"

. In Elastix, we create an IVR menu and set it as a destination for queue "400". This menu has only one option (1) that basically jumps to the custom extension "399" that we just created, in order to call our script

. we go back to the queue "400" and set its "Fail Over Destination" as our IVR we just created

. We start by editing the extensions_custom.conf file in our system, adding a new stanza like:

```
[queue-leavenumber]
exten => 1,1,NoOp
exten => 1,n,Set(LANG=en)
exten => 1,n(Start),agi(googletts.agi,"Please enter your number, followed by the hash
       digit.",${LANG})
exten => 1,n,agi(googletts.agi,"We will be happy to call you back!",${LANG})
exten => 1,n,Read(CBNUM,beep,0,,2,10)
exten => 1,n,NoOp( Num ${CBNUM} )
exten => 1,n,GotoIf($["${LEN(${CBNUM})}">="3"]?lenOk)
exten => 1,n,agi(googletts.agi,"The number you entered has the wrong number of
       digits.",${LANG})
```

```
exten => 1,n,GoTo(hang)
exten => 1,n,Wait(1)
exten => 1,n(lenOk),agi(googletts.agi,"You entered the following number",${LANG})
exten => 1,n,SayDigits(${CBNUM})
exten => 1,n,agi(googletts.agi,"Press 1",${LANG})
exten => 1,n,Read(CONF,beep,1,,2,5)
exten => 1,n,GotoIf($["${CONF}"="1"]?Store:Start)
exten => 1,n(Store),NoOp
exten => 1,n,Set(WHEN=${STRFTIME(${EPOCH},,%y%m%d-%H%M%S)})
exten => 1,n,NoOp(When:${WHEN})
exten => 1,n,Set(PARM=number=${CBNUM}&attrs=orgQ:400%2Cwhen:${WHEN})
exten => 1,n,Set(foo=${CURL(http://xxx.xxx.xxx.xxx:8080/wombat/api/calls/?
op=addcall&campaign=callback&${PARM})})
exten => 1,n,agi(googletts.agi,"Thank you! We will get in touch with you as soon as
possible.",${LANG})
exten => 1,n,Hangup
```

Where xxx.xxx.xxx.xxx is the IP address of your WombatDialer server.
It's of paramount importance that all the above instruction lines are not broken.
Every line must start with the “exten” notation. For example the line:

```
exten => 1,n,Set(foo=${CURL(http://xxx.xxx.xxx.xxx:8080/wombat/api/calls/?
op=addcall&campaign=callback&${PARM})})
```

must not be divided in two lines as in this document. Please make sure that in your dial-plan each “exten” instruction lies on one line only.

We use Google TTS as a voice synthesizer - you could use a different one or you could have the messages custom-recorded for you. What our dialplan does is first to collect a number composed by at least three digits (followed by the hash digit), then read it back asking for confirmation and when confirmed, it sends it over to WombatDialer on a campaign called “callback”. Together with the number, we also store the code of the queue that the call was on and the date and time this number was gathered. (Please note that in order to send multiple comma-separated parameters in the HTTP request, we have to use ’%2C’ instead of the comma ’,’).

In order to configure WombatDialer:

. We create a trunk called “Trunk” with a dial-string of Local/9${num}@from-internal and a capacity of 10 lines. This basically replies all numbers as if they were entered on a local extension prefixed by 9.

. We create an End-Point of type Queue for monitoring queue 401; set extension to “401” and context to “from-internal”; max number of lines to 10; boost factor as 1 and max waiting calls to 2. This means that the number of calls placed will match the number of available agents on

queue 401.

. We create a campaign called “callback”; set it to Idle on termination and turn on QM_COMPATILE logging. We add the trunk and the EP we just created. We create a set of reschedule rules in order to handle REJECTED, BUSY, INVALID and NOANSWER calls, e.g. by retrying up to 5 times each waiting 10 minutes between each attempt. Note that we create no lists for this campaign.

. We start the new campaign; having no numbers, it should immediately turn yellow on the Live page to tell you it’s idling.

If we start sending calls to the queue and we try and leave any numbers, we will see that a new list will be created on WombatDialer under the name “callback/AUTO” and that will contain the numbers and attributes like:

```
Number:     5551235
Attributes: orgQ:400   when:121115-153402
```

Those numbers are NOT immediately called, but WD will wait for some agent to be present and active on queue “401” so that they can be called back. This way, the call-center manager can monitor the current call backlog and decide who and when it is to join the callback queue.

How to install googletts on CentOS 6

open the terminal and run

```
yum -y install perl perl-libwww-perl sox cpan bzip2

rpm -Uhv http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.3-1.el6.rf.x86_64.rpm
yum -y install mpg123

wget https://raw.githubusercontent.com/zaf/asterisk-googletts/master/googletts.agi
mv googletts.agi /var/lib/asterisk/agi-bin/
chmod 755 /var/lib/asterisk/agi-bin/googletts.agi
```

Extra handling

You might want to add some logic to the handling of the callback by adding another stanza in the extensions_custom.conf.

This could be useful if, for example, you wanted to save some attributes from the original call (as for the “when” attribute in the previous example) and use it when you handle the callback call.

In the following example we'll see how to pass an "originalCaller" attribute to WD in order to show who actually reserved the callback,
regardless of the actual number that was scheduled for callback.
We will also set the caller name to RECALL so that the agent knows that he is actually answering a callback call.

. First we have to modify our end-point's context and extension. We'll set the extension to 1 and the context to queue-handling

. In the queue-leavenumber context we add the attributes we want to pass to WD editing the following line

```
exten => 1,n,Set(PARM=number=${CBNUM}&attrs=orgQ:400%2Cwhen:${WHEN})
```

changing it into

```
exten => 1,n,Set(PARM=number=${CBNUM}&attrs=orgQ:400%2Cwhen:${WHEN}
%2CoriginalCaller:${CALLERID(number)})
```

this way we mark this call with the attribute originalCaller which holds the information on the original caller.

. Then we add a new stanza in extensions_custom.conf like the following:

```
[queue-handling]
exten => 1,1,NoOp
exten => 1,n,Set(CALLERID(name)=RECALL)
extem => 1,n,NoOp(originalCaller:${originalCaller})
exten => 1,n,Goto(“from-internal”,401,1)
```

as we can see we succesfully handled the callback calls, being able to save date from the original call and to pass it to the callback call.
The last instruction redirects the call to the “from-internal” context at extension 401 after handling the callback, so that it can be answered by a free agent. You can redirect the call to wherever you want from here.