mod_fifo

About

mod_fifo is a call center app which allows you to make custom call queues with assigned priorities.

FIFO stands for "First In, First Out". As calls enter the queue, they are arranged in order so that the call that has been in the queue for the longest time will be the first call to get answered. Generally FIFO call queues are used in "first come, first served" call scenarios such as customer service call centers.

An alternative to mod_fifo is mod_callcenter which is more of a traditional ACD application and was contributed by a member of the FreeSWITCH™ community.

Usage

App Invocation

```
<fifo name>[!<importance_number>] [in [<announce file>|undef] [<music file>|undef] | out [wait|nowait]
[<announce file>|undef] [<music file>|undef]]
```

Definitions

Keep these terms in mind as they will help you with your mod_fifo endeavors:

- Consumer - An **off-hook agent**, that is, an agent who calls into the FIFO queue and is waiting to 'consume' incoming calls from the queue. The consumer's phone is in use the entire time that he/she is logged in to the queue.
- Member - An **on-hook agent**, that is, an agent who logs into the queue but whose phone is idle until an inbound call hits the queue. The agent's phone rings and he/she takes a call off the queue. Multiple members' phones can ring for each incoming call.
- Caller - A person who has called into the queue. The caller can be "waiting" or can be speaking to an agent.

Configuration

Global configuration parameters for mod_fifo are contained in:

```
$CONFDIR/autoload_configs/fifo.conf.xml
```

Agent Callback

This will call registered agents. Think asterisk queue members.

```
<configuration name="fifo.conf" description="FIFO Configuration">
   <fifos>
      <fifo name="fifo1@domain.name.com" importance="0">
         <member timeout="60" simo="1" lag="20">{fifo_member_wait=nowait}user/1005@$${domain}</member>
      </fifo>
   </fifos>
</configuration>
```

- **member timeout:** how long to ring the extension (in seconds) before moving on.
- **simo:** number of simultaneous registrations to ring. Must be set to at least 1.
    - If you have multiple phones registered to the same extension, this controls how many extensions to ring at once
    - Example: 10 phones registered to one extension, set simo="10" to ring all 10.
- **lag:** seconds to wait before initiating another call (think wrapup-time in Asterisk).
- **fifo_member_wait:** possible values are either *wait* or *nowait*. (more info)

Note: If you wish to specify the caller ID presented when a fifo calls an agent, set the origination_caller_id_name and origination_caller_id_num variables to the values desired. These could be set within the {} of the dialstring, or they could be set using the set application in the dialplan which places the caller into the fifo (before the 'fifo in' executed on the caller).

A real call center example

Dialplan Example

Put a caller into a FIFO queue

```
<action application="fifo" data="myqueue in /tmp/exit-message.wav /tmp/music-on-hold.wav"/>
```

The preceding will put a call into "myqueue", playing "/tmp/music-on-hold.wav" over and over again. "/tmp/exit-message.wav" will be played to the user as the user is being taken out of the queue.

Take a caller out of a FIFO queue

```
<action application="fifo" data="myqueue out nowait /tmp/caller-found.wav /tmp/agent-music-on-hold.wav"/>
```

This pulls the oldest caller out of "myqueue" and connects them with the agent on the current channel. "/tmp/caller-found.wav" will be played to the agent as the call is taken out of the FIFO.

If there are no calls in the FIFO queue and you have the "wait" parameter set, "/tmp/agent-music-on-hold.wav" will be played to the agent over and over again until a new call arrives. Additionally, after taking a call from the queue which ends without the agent hanging up, the agent will be returned to the FIFO queue.

If you have the "nowait" parameter set and there are no calls in the FIFO queue, processing immediately continues past the FIFO queue. If there are one or more calls in the queue, only one call is retrieved and processing continues past the FIFO queue after that call ends.

Taking a specific caller out of a queue

If you want to pick a specific caller out of queue, set the variable fifo_bridge_uuid to be the UUID of the caller in queue.

Setting MOH and announce sounds

Here are some ways that you can use to set the MOH and announce sounds for both fifo in and fifo out

```
<action application="set" data="fifo_music=<sound path>"/>
<action application="set" data="fifo_announce=<sound path>" />
<action application="set" data="fifo_orbit_announce=<sound path>" />
<action application="set" data="fifo_orbit_exten=<exten>:[timeout]" />
<action application="set" data="fifo_override_announce=<sound path>" />
```

Implementing FIFO slots

Each FIFO can have 10 priority slots (default priority is 5). Priority 1 has higher priority than 10. With the FIFO slot, you can put a caller into one of the ten FIFO slots with:

```
<action application="set" data="fifo_priority=1" />
```

Consumers can be assigned to pick up callers with specific priority using fifo_pop_order variable, such as follow:

```
<action application="set" data="fifo_pop_order=1,2" />
```

You can assign multiple priority orders using a comma-delimited list.

Agent/Caller Example

This scenario has two extensions: 7010 will be for agents who will hear music till someone calls 7011 will be the customer who will hear hold music until an agent is free.

```
    <extension name="Agent_Wait">
      <condition field="destination_number" expression="^7010$">
        <action application="set" data="fifo_music=$${hold_music}"/>
        <action application="answer"/>
        <action application="fifo" data="myq out wait"/>
      </condition>
    </extension>
    <extension name="Queue_Call_In">
      <condition field="destination_number" expression="^7011$">
        <action application="set" data="fifo_music=$${hold_music}"/>
        <action application="answer"/>
        <action application="fifo" data="myq in"/>
      </condition>
    </extension>
```

The agents can dial in to extension 7010 and wait. Callers can be routed/transferred to extension 7011 where they'll be queued until an agent is available.

Park Time Out Example

This Example will return the parked call to exten 1004 after 20 sec.

```
 <extension name="park" continue="true">
   <condition field="destination_number" expression="^5900$">
     <action application="set" data="fifo_music=$${hold_music}"/>
     <action application="set" data="fifo_orbit_exten=1004:20"/>
     <action application="fifo" data="5900@$${domain} in"/>
   </condition>
 </extension>
```

Park/Un-park Example

See Park & Retrieve

Simple On-hook Agent Login/Logout Example

```
 <include>

  <!--

    This is a simple FIFO agent login/logout dialplan example. Drop it into the conf/dialplan/default directory
in a standard
    FreeSWITCH install or modify it as you see fit.

    These are "on hook" agents, i.e. the phone rings when a call comes in; agents don't sit and wait on the
phone.
    Dial 6* plus the FIFO number to log in.  Example: 6*1 to log into "FIFO1"
    Dial 6# plus the FIFO number to log out. Example: 6#1 to log out of "FIFO1"
```

```
    NOTE: I added 6** as an alternative to logging out b/c some phone dialplans don't like 6#...

    By default there are ten slots. You can mix, match, add, etc. as you see fit.
    NOTE: There isn't any error checking, so the the system will say that the user is logged in or out no
matter what!
    NOTE: I have added an option to use group_confirm so that your agents have to press 1 to accept. This is
useful in the case where an agent is a cell phone or other phone that has voicemail.

    Send a caller to the queue by transferring to 610[0-9]; 6101 corresponds to FIFO1, 6102 to FIFO2, etc.

    To see that your user did indeed login go to the FS CLI and type "fifo list" and it will show you
everything.
    Use "fifo list_verbose" to see LOTS of information about the FIFO queues.

  -->

  <!-- Agent login extension: 6*[0-9] -->
  <extension name="Agent Login">
    <condition field="destination_number" expression="^6\*(\d)">
      <action application="answer"/>
      <action application="set" data="result=${fifo_member(add FIFO$1 {fifo_member_wait=nowait}user
/${user_name} )"/>
      <!-- use the following line instead if you want to have group_confirm for the agent
        <action application="set" data="result=${fifo_member(add FIFO$1 {fifo_member_wait=nowait,
group_confirm_file=ivr/ivr-accept_reject_voicemail.wav,group_confirm_key=1}user/${user_name} )"/>
      -->

      <action application="log" data="INFO Add FIFO agent result: ${result}"/>
      <action application="log" data="INFO User Login: FIFO$1 User: ${user_name}"/>
      <!-- No error checking, just assuming login went well... -->
      <action application="playback" data="ivr/ivr-you_are_now_logged_in.wav"/>
    </condition>
  </extension>

  <!-- Agent logout extension: 6#[0-9] -->
  <extension name="Agent Logout">
    <condition field="destination_number" expression="^6(#|\*\*)(\d)">
      <action application="answer"/>
      <action application="set" data="result=${fifo_member(del FIFO$2 {fifo_member_wait=nowait}user
/${user_name})}"/>
      <!-- Use this line instead if you are using group_group confirm
           <action application="set" data="result=${fifo_member(del FIFO$2 {fifo_member_wait=nowait,
group_confirm_file=ivr/ivr-accept_reject_voicemail.wav,group_confirm_key=1}user/${user_name} )"/>
       -->

      <action application="log" data="INFO Del FIFO agent result: ${result}"/>
      <action application="log" data="INFO User Logout: FIFO$1 User: ${user_name}"/>
      <!-- No error checking, just assuming logout went well... -->
      <action application="playback" data="ivr/ivr-you_are_now_logged_out.wav"/>
    </condition>
  </extension>


  <!-- Send a call to FIFO[0-9] -->
  <extension name="send caller to FIFO">
    <condition field="destination_number" expression="^610(\d)$">
      <action application="answer"/>
      <action application="set" data="fifo_music=$${hold_music}"/>
      <action application="playback" data="ivr/ivr-hold_connect_call.wav"/>
      <action application="fifo" data="FIFO$1 in"/>
    </condition>
  </extension>
</include>
```

See also the contrib file: 01_FIFO_example.xml

Another example of On-hook Agent Login/Logout with loopback members

You might also want to mess around with the call that is being generated by the FIFO to the agent. You can achieve that by using mod_loopback to make the call go back to the dialplan as shown:

```
<extension name="Agent Login">
 <condition field="destination_number" expression="^6\*(\d)">
  <action application="answer"/>
  <action application="set" data="result=${fifo_member(add ACD$1 {fifo_member_wait=nowait}loopback/1${user_name}
/default/XML )"/>
  <action application="log" data="INFO Add FIFO agent result: ${result}"/>
  <action application="log" data="INFO User Login: ACD$1 User: ${user_name}"/>
  <action application="playback" data="ivr/ivr-you_are_now_logged_in.wav"/>
 </condition>
</extension>
```

You would also have to change the way you log agents out like this:

```
<extension name="Agent Logout">
 <condition field="destination_number" expression="^6#(\d)">
  <action application="answer"/>
  <action application="set" data="result=${fifo_member(del ACD$1 {fifo_member_wait=nowait}loopback/1${user_name}
/default/XML)}"/>
  <action application="log" data="INFO Del FIFO agent result: ${result}"/>
  <action application="log" data="INFO User Logout: ACD$1 User: ${user_name}"/>
  <action application="playback" data="ivr/ivr-you_are_now_logged_out.wav"/>
 </condition>
</extension>
```

Finally, you would have to setup the piece of the dialplan where the FIFO, using mod_loopback, is going to place the call to. I chose a simple one like this:

```
<extension name="Dial to Agent from Queue">
 <condition field="destination_number" expression="^1(10[01][0-9])$">
  <action application="bridge" data="user/$1@$${domain}"/>
 </condition>
</extension>
```

On the code above, you have to be careful with applications that would make originate return with a successful call such as pre_answer or start_dtmf. If it returns without you making sure agent picked up, both legs are going to be hungup.

See contrib file Based on MCollins dialplan

JavaScript Example

The following are JavaScript versions of the examples above. Here we are adding a call to a queue.

```
session.execute( "fifo", "myqueue in 'sounds/your-being-picked-up.wav' 'sounds/tunes.wav'" );
```

You can replace the sound files with "undef" if you don't want them. Here we are taking a call out of a queue without specifying an announcement file.

```
session.execute( "fifo", "myqueue out wait undef 'sounds/tunes.wav'" );
```

This call is **non blocking** when putting a call into the fifo. You still have access to the session but you won't know, for example, when the call is taken off of the queue. If you are interested in these events, watching the event socket is where you should focus.

Lua Example, queue position announcement trick

The fifo_chime_list variable may point at a fixed symlink (were are in the Linux World), and by changing the symlink pointer during the period, where the customer waits inside the fifo, the sound played back will change. The Lua example below implements this trick. The channel variables: fifo_chime_list and fifo_chime_freq must be set before the customer enters the fifo. In order to make the symlink unique, the uuid is used at the first name of the symlink:

```
-- Scriptname: voiceprompts.lua
-- This script provides queue position announcement
-- using a symlink to the appropiate queue length file

-- Initialization of variables
api = freeswitch.API()
caller_uuid = argv[1]
-- Queue_name not used presently
queue_name = argv[2]
times = 1
-- The 0.wav file is a very short silence sound file
symlinkcommand = "ln -s -f /var/spool/voiceprompts/0.wav /var/spool/voiceprompts/"..caller_uuid..".wav"
os.execute(symlinkcommand)

while (times<1000) do
  freeswitch.msleep(2000)
  position = api:executeString("uuid_getvar "..caller_uuid.." fifo_position")
  freeswitch.consoleLog("notice","Callers fifo position: "..position.."\n");
  if (string.sub(position,1,4)~="-ERR") then
    position = tonumber(position)
    times = times + 1
    -- We only announce the position when the caller is number two, as number one may already be on the way to
be helped.
    position = position - 1
    if (position > 3) then position = 3 end
    symlinkcommand = "ln -s -f /var/spool/voiceprompts/"..position..".wav /var/spool/voiceprompts/"..
caller_uuid..".wav"
    freeswitch.consoleLog("notice","Symlinkcommand: "..symlinkcommand.." result: "..os.execute
(symlinkcommand).."\n");
    os.execute(symlinkcommand)
  else times = 1000 end
  freeswitch.msleep(4000)
end
delsymlink = "rm '/var/spool/voiceprompts/"..caller_uuid..".wav'"
result = os.execute(delsymlink)
```

The Lua script must be called by the dialplan before the customer enters the fifo. The dialplan for the queue is:

```
<extension name="Receive queue, fifo customer" >
  <condition field="destination_number" expression="^(\*16\*.*)$">
    <action application="set" data="queue_caller_id_name=Queue-${destination_number:4}
-${effective_caller_id_name}"/>
    <action application="set" data="queue_name=${used_domain}${destination_number:4}"/>
    <action application="set" data="fifo_music=$${hold_music}"/>
    <action application="set" data="fifo_chime_list=/var/spool/voiceprompts/${uuid}.wav"/>
    <action application="set" data="fifo_chime_freq=30"/>
    <action application="set" data="result=${fifo_member(add ${queue_name} {fifo_member_wait=nowait,
used_domain=${used_domain},source=queue,origination_caller_id_name=${queue_caller_id_name}}loopback
/queue${queue_name}/internalpreparation/XML 1 20 5)}"/>
    <action application="set" data="continue_on_fail=false"/>
    <action application="answer" />
    <action application="playback" data="/var/spool/voiceprompts/${queue_name}.wav"/>
    <action application="set" data="result=${luarun(voiceprompts.lua  ${uuid} ${queue_name})}"/>
    <action application="fifo" data="${queue_name} in"/>
  </condition>
</extension>
```

Naturally the choice of prefix: *16*, and the naming of the queue is unimportant for this example.

Queue Position, alternate method

Note that a combination of phrase macros and uuid_getvar can be used to accomplish this as well. Before placing the call in fifo, set the fifo_chime_list value as follows:

```
<action application="set" data="fifo_chime_list=phrase:queue_position_macro:${uuid}" />
```

Inside the phrase macro, use the UUID passed in to get the queue position:

```
<macro name="queue_position_macro">
        <input pattern="(.*)">
                <match>
                        <action function="play-file" data="queue_pos_intro" /> <!-- "You are caller number" -->
                        <action function="play-file" data="cardinals/${uuid_getvar $0 fifo_position}" /> <!--
sound files with names "1", "2", etc. -->
                </match>
        </input>
</macro>
```

Playing the caller number could, of course, be handled with another macro to play large numbers more eloquently.

Details

The agent prompt will play to the agent as the caller hears hold music then the agent will hear nothing as the caller is played their exit message. The agent can end the call with the caller at any time by pressing "*". If the agent hangs up the call while either the agent's prompt or the callers prompt are being played, but before the calls are bridged, the caller will be dropped. (should this be fixed?)

Using bind_meta_app prior to calling 'fifo out x' to refer to, say, *1 to transfer calls will override the '*' hangup functionality. In this case, an additional * is necessary to use this hangup feature. ('**' instead of '*')

The fifo_music channel variable (which can be set globally in freeswitch.xml) will control the music that's played in the FIFO.

Note (this applies to FreeSWITCH 1.0.1 and later): setting the variable "fifo_consumer_exit_key" to something, you can override the default "*" key that exits the call on the consumer side.

API Commands

fifo

list|list_verbose

```
fifo list|list_verbose [ fifo_name ]
```

Both 'fifo list' and 'fifo list_verbose' give tons of information about the FIFO's that are currently defined, including any customer who are waiting or being serviced, the consumers in the FIFO, and the members. If we specify this command with the fifo name then it filters the information of given fifo name alone.

count

```
fifo count [ fifo_name ]
```

Example output:

```
fifo_1:0:0:1:0:0
fifo_2:0:0:0:0:0
```

For each FIFO, a colon ':' separated list of counts in the following format is returned:

```
fifo_name:consumer_count:caller_count:member_count:ring_consumer_count:idle_consumer_count
```

importance

```
fifo importance <fifo_name>
```

Outputs the importance value for the named fifo.

reparse [del_all]

```
fifo reparse [del_all]
```

Re-parses the conf for mod_fifo; if you are using mod_xml_curl for configuration data, this will also re-request the xml.

The parameter del_all causes cleanup of used memory (fifos are not deleted automatically when they become empty). Memory would not be an issue in most installations, as each fifo consumes very little of it, but it might eventually happen if you are dealing with multi-tenancy.

fifo_member

add

```
add <fifo_name> <originate_string> [<simo_count>] [<timeout>] [<lag>] [expires] [taking-calls]
```

Adds a member to the named fifo, with the same values as described in the Agent Callback section. If the named FIFO does not already exist, it is created.

del

```
del <fifo_name> <originate_string>
```

Deletes the member from the named fifo. Deleting the last member from the FIFO does **not** remove the FIFO.

Additional variables

- fifo_destroy_after_use: FreeSWITCH automatically create a new FIFO when the first time use it, and keep in the memory hash. This var tell FreeSWITCH destroy it to save memory. Using for a one time FIFO.

Caller leg channel variables used or set by mod_fifo

- **fifo_chime_list**: a list of files to broadcast while the customer is waiting for someone to answer
- **fifo_chime_freq:** playing sound files to caller every fifo_chime_freq seconds

- **fifo_orbit_exten:** (exten:timeout), transfer the caller to exten on timeout (use "_continue_" to move on in current extension)
- **fifo_orbit_dialplan:** use this dialplan for the transfer, made on timeout. if not present use the current dialplan
- **fifo_orbit_context:** use this context for the transfer, made on timeout. if not present use the current context
- **fifo_orbit_announce:** play to caller
- **fifo_caller_exit_key:** caller hangup by default
- **fifo_caller_exit_to_orbit:** use with fifo_caller_exit_key, transfer the caller to exten other than hangup
- **fifo_serviced_uuid:** ?
- **fifo_status:** WAITING, TIMEOUT, ABORTED, DONE
- **fifo_timestamp:** timestamp of when the caller was bridged to the consumer
- **fifo_serviced_by:** the uuid of the consumer leg that picked-up of the call
- **fifo_serviced_uuid:** the uuid of the consumer leg that triggered the pick-up of the call, matches fifo_serviced_by if fifo_consumer_id is not set

Consumer leg channel variables set by mod_fifo

These channel variables will be set by mod_fifo according to what is specified for these variables in the configuration of a particular consumer.

If a consumer is statically defined as a member, the values for these variables may be set within {} of the dial string used to call the consumer. To set multiple variables within the {}, separate each variable=value by commas.

If a consumer executes dialplan to receive a call from a fifo, the values for these variables may be set (with the set application) in the dialplan the consumer executes.

- **fifo_strategy:** the strategy to get the caller out of the fifo. Can be: "more_ppl" or "waiting_longer" (default);
- **fifo_consumer_id:** if set, this indicates which uuid should the fifo consumer call be transferred to, useful when the consumer is another uuid;
- **fifo_record_template:** if set, this is the file where the session will record to, expanded on the caller channel;
- **fifo_status:** WAITING or TALKING
- **fifo_target:** the uuid of the call the consumer is talking to
- **fifo_override_announce:** overrides the consumer announce of the fifo
- **fifo_consumer_wrapup_sound:** played at the end of a serviced caller
- **fifo_consumer_wrapup_key:** to
- **fifo_pop_order:** comma separated list of the priorities this consumer calls for
- **fifo_member_wait:** This variable can be set to either 'wait' or 'nowait' on the consumer leg.
    - If it's set to 'wait', then the consumer's leg of the call will not hangup when the caller hangs up [default].
    - If it's set to 'nowait' then the consumer's leg of the call will hangup when the caller hangs up.

Examples

nowait setting and recording file name

- For on hook consumer (statically defined); in this example 12345 will ring when a caller is in the fifo named MyFifo@$${domain}

```
<configuration name="fifo.conf" description="FIFO Configuration">
   <fifos>
     <fifo name="MyFifo@$${domain}" importance="0">
       <member timeout="20" simo="3" lag="0">{fifo_member_wait=nowait,fifo_record_template=$${base_dir}
/recordings/myfifo_call.wav}user/12345@$${domain}</member>
     </fifo>
   </fifos>
 </configuration>
```

- For consumer that dials in to consume; in this example 56789 must be dialed to retrieve a caller from the fifo named MyFifo@$${domain}

```
 <extension name="Get_Fifo_Call">
   <condition field="destination_number" expression="^56789$"?
     <action application="set" data="fifo_record_template=$${base_dir}/recordings/myfifo_call.wav"/>
     <action application="answer"/>
     <action application="fifo" data="MyFifo@$${domain} out nowait"/>
   </condition>
 </extension>
```

See Also

- mod_callcenter