mod_callcenter

About

mod_callcenter is an **inbound call queuing application** that can be used for call center needs.

The callcenter dialplan application provides call center functionality by distributing calls to agents using various scenarios and rules. A score-based system is used to distribute inbound calls. A caller's score increases by 1 for every second he waits. You can add a base score to help move a caller to the front of a queue or just remove the wait time from that queue. The callcenter application also has a tiered system for creating different agent 'priorities' as needed.

A simpler alternative means of handling incoming call queues is with mod_fifo, a first-in, first-out (FIFO) queuing system.

Configuration

Settings

odbc-dsn

The callcenter will use the supplied ODBC database instead of the default behavior, which is to use the internal SQLite database.

dbname

This is to specify a different name or path and name of the SQLite database. Useful to put into a ram disk for better performance.

Agent options

type

We currently support 2 types, 'callback' and 'uuid-standby'. callback will try to reach the agent via the contact fields value. uuid-standby will try to bridge the call directly using the agent uuid.

contact

A simple dial string can be put in here, like: user/1000@default. If using verto: ${verto_contact(1000@default)}

status

Define the current status of an agent. Check the Agents Status table for more information.

max-no-answer

If the agent fails to answer calls this number of times, his status is changed to *On Break* automatically.

wrap-up-time

The amount of time to wait before putting the agent back in the available queue to receive another call, to allow her to complete notes or other tasks.

reject-delay-time

If the agent presses the reject button on her phone, wait this defined time amount.

busy-delay-time

If the agent is on *Do Not Disturb*, wait this defined time before trying him again.

no-answer-delay-time

If the agent does not answer the call, wait this defined time before trying him again.

reserve-agents

If defined to true, agent state is changed to Reserved if the old state is Receiving, the call will only be sent to him if the state get's changed.

This is useful if you're manipulating agent state external to mod_callcenter. false by default.

truncate-agents-on-load

If defined to true, we'll delete all the agents when the module is loaded. false by default.

truncate-tiers-on-load

If defined to true, we'll delete all the tiers when the module is loaded. false by default.

Queue options

strategy

The strategy defines how calls are distributed in a queue. A table of different strategies can be found below.

moh-sound

The system will playback whatever you specify to incoming callers. You can use any type of input here that is supported by the FreeSWITCH playback system:

1. A direct path to a .wav file will play in a loop indefinitely.
2. The local stream, e.g. (local_stream://moh) *or* use $${hold_music} as defined in the default configuration.
3. The FreeSWITCH phrase system, e.g., (phrase:my-special-phrase). (I use this to play multiple prompts after each other.)
4. A tone stream as with ringing, e.g., (tone_stream://${us-ring};loops=-1).

(JB - resume editing here)

record-template

Use the record-template to save your recording wherever you would like on the filesystem. It's not uncommon for this setting to start with "$${base_dir}/recordings/". Whatever directory you choose, make sure it already exists and that FreeSWITCH has the required permissions to write to it.

time-base-score

This can be either 'queue' or 'system' (queue is the default). If set to system, it will add the number of seconds since the call was originally answered (or entered the system) to the caller's base score. Raising the caller's score allows them to receive priority over other calls that might have been in the queue longer but *not* in the system as long. If set to queue, you get the default behavior, i.e., nobody's score gets increased upon entering the queue (regardless of the total length of their call).

tier-rules-apply

Can be True or False. This defines if we should apply the following tier rules when a caller advances through a queue's tiers. If False, they will use all tiers with no wait.

tier-rule-wait-second

The time in seconds that a caller is required to wait before advancing to the next tier. This will be multiplied by the tier level if tier-rule-wait-multiply-level is set to True. If tier-rule-wait-multiply-level is set to false, then after tier-rule-wait-second's have passed, all tiers are open for calls in the tier-order and no advancement (in terms of waiting) to another tier is made.

tier-rule-wait-multiply-level

Can be True or False. If False, then once tier-rule-wait-second is passed, the caller is offered to all tiers in order (level/position). If True, the **tier-rule-wait-second** will be multiplied by the tier level and the caller will have to wait on every tier **tier-rule-wait-second'**s before advancing to the next tier.

tier-rule-no-agent-no-wait

Can be True or False. If True, callers will skip tiers that don't have agents available. Otherwise, they are be required to wait before advancing. Agents must be logged off to be considered not available.

discard-abandoned-after

The number of seconds before we completely remove an abandoned member from the queue. When used in conjunction with abandoned-resume-allowed, callers can come back into a queue and resume their previous position.

abandoned-resume-allowed

Can be True or False. If True, a caller who has abandoned the queue can re-enter and resume their previous position in that queue. In order to maintain their position in the queue, they must not abandoned it for longer than the number of seconds defined in 'discard-abandoned-after'.

max-wait-time

Default to 0 to be disabled. Any value are in seconds, and will define the delay before we quit the callcenter application IF the member haven't been answered by an agent. Can be used for sending call in voicemail if wait time is too long.

max-wait-time-with-no-agent

Default to 0 to be disabled. The value is in seconds, and it will define the amount of time the queue has to be empty (without logged agents, on a call or not) before we disconnect all members. This principle protects kicking all members waiting if all agents are logged off by accident.

max-wait-time-with-no-agent-time-reached

Default to 5. Any value are in seconds, and will define the length of time after the max-wait-time-with-no-agent is reached to reject new caller. This allow for kicking caller if no agent are logged in for over 5 seconds, but new caller after that 5 seconds is reached can have a lower limit.

ring-progressively-delay

Default to 10. The value is in seconds, and it will define the delay to wait before starting call to the next agent when using the 'ring-progressively' queue strategy.

callcenter.conf.xml example

```
<configuration name="callcenter.conf" description="CallCenter">

  <settings>
      <!--<param name="odbc-dsn" value="dsn:user:pass"/>-->
      <!--<param name="dbname" value="/dev/shm/callcenter.db"/>-->
  </settings>

  <queues>
    <queue name="sales@default">
      <param name="strategy" value="agent-with-least-talk-time"/>
      <param name="moh-sound" value="$${hold_music}"/>
      <!--<param name="record-template" value="$${base_dir}/recordings/sales/${strftime(%Y-%m-%d-%H-%M-%
S)}.${destination_number}.${caller_id_number}.${uuid}.wav"/>-->
      <param name="time-base-score" value="queue"/>
      <param name="tier-rules-apply" value="false"/>
      <param name="tier-rule-wait-second" value="300"/>
      <param name="tier-rule-wait-multiply-level" value="true"/>
      <param name="tier-rule-no-agent-no-wait" value="false"/>
      <param name="discard-abandoned-after" value="14400"/>
      <param name="max-wait-time" value="0"/>
      <param name="max-wait-time-with-no-agent" value="120"/>

    </queue>
    <queue name="support@default">
      <param name="strategy" value="longest-idle-agent"/>
      <param name="moh-sound" value="$${hold_music}"/>
      <!--<param name="record-template" value="$${base_dir}/recordings/support/${strftime(%Y-%m-%d-%H-%M-%
S)}.${destination_number}.${caller_id_number}.${uuid}.wav"/>-->
      <param name="time-base-score" value="system"/>
      <param name="tier-rules-apply" value="false"/>
      <param name="tier-rule-wait-second" value="300"/>
      <param name="tier-rule-wait-multiply-level" value="true"/>
      <param name="tier-rule-no-agent-no-wait" value="false"/>
      <param name="discard-abandoned-after" value="60"/>
      <param name="abandoned-resume-allowed" value="false"/>
      <param name="max-wait-time" value="0"/>
      <param name="max-wait-time-with-no-agent" value="120"/>
    </queue>
  </queues>

<!-- WARNING: Configuration of XML Agents will be updated into the DB upon restart. -->
<!-- WARNING: Configuration of XML Tiers will reset the level and position if those were supplied. -->
<!-- WARNING: Agents and Tiers XML config shouldn't be used in a multi FS shared DB setup. -->

  <agents>
    <agent name="1000@default" type="callback" contact="[leg_timeout=10]user/1000@default" status="Available"
max-no-answer="3" wrap-up-time="10" reject-delay-time="10" busy-delay-time="60" />
    <!-- If you would like to set the Caller ID name, for whatever reason notice below. -->
    <agent name="1001@default" type="callback" contact="[origination_caller_id_name='Queue Caller',
leg_timeout=10]user/1001@default" status="Available" max-no-answer="3" wrap-up-time="10" reject-delay-time="10"
busy-delay-time="60" />
  </agents>

  <tiers>
    <!-- If no level or position is provided, they will default to 1. You should do this to keep db value on
restart. -->
    <!-- agent 1000 will be in both the sales and support queues -->
    <tier agent="1000@default" queue="sales@default" level="1" position="1"/>
    <tier agent="1000@default" queue="support@default" level="1" position="1"/>
    <!-- agent 1001 will only be in the support queue -->
    <tier agent="1001@default" queue="support@default" level="1" position="1"/>
  </tiers>

</configuration>
```

Dialplan Examples

Put a caller into a queue

```
<action application="callcenter" data="support@default"/>
```

Time of Day / Voicemail timeout Example

```
<condition field="destination_number" expression="^3000$" break="on-false"/>
<!-- Mon-Fri, 9am-5pm -->
<condition wday="2-6" time-of-day="09:00-17:00" break="on-true">
  <!-- limit 3 calls to this destination number per 1 second, otherwise give congestion message -->
  <action application="limit" data="hash inbound ${destination_number} 3/1 !NORMAL_CIRCUIT_CONGESTION"/>
  <!-- play a message before entering the queue. -->
  <action application="playback" data="$${sounds_dir}/greeting.wav"/>
  <!-- hangup after successful bridge to agent -->
  <action application="set" data="hangup_after_bridge=true"/>
  <!-- queue caller -->
  <action application="callcenter" data="queue@default"/>
  <!-- if no agent was reached and using max-wait-time - send to general voicemail -->
  <action application="playback" data="$${sounds_dir}/queue_voicemail.wav"/>
  <action application="answer"/>
  <action application="set" data="skip_greeting=true"/>
  <action application="set" data="skip_instructions=true"/>
  <action application="voicemail" data="default voicemail 2001"/>
  <action application="hangup"/>
</condition>
<condition>
  <action application="limit" data="hash inbound ${destination_number} 3/1 !NORMAL_CIRCUIT_CONGESTION"/>
  <!-- outside business hours - play message and send to general voicemail -->
  <action application="playback" data="$${sounds_dir}/tod_voicemail.wav"/>
  <action application="answer"/>
  <action application="set" data="skip_greeting=true"/>
  <action application="set" data="skip_instructions=true"/>
  <action application="voicemail" data="default voicemail 2001"/>
  <action application="hangup"/>
</condition>
```

Queues

Queues can only be configured in the XML configuration. They are only loaded once.

Distribution Strategy

String	Description
ring-all	Rings all agents simultaneously.
longest-idle-agent	Rings the agent who has been idle the longest taking into account tier level.
round-robin	Rings the agent in position but remember last tried agent.
top-down	Rings the agent in order position starting from 1 for every member.
agent-with-least-talk-time	Rings the agent with least talk time.
agent-with-fewest-calls	Rings the agent with fewest calls.
sequentially-by-agent-order	Rings agents sequentially by tier & order.

random	Rings agents in random order.
ring-progressively	Rings agents in the same way as top-down, but keeping the previous members ringing (it basically leads to ring-all in the end).

Time base score

When a caller goes into a queue, we can add to their base score the total number of seconds they have been in the system. This enables the caller to get in front of other callers by the amount of time they have already spent waiting elsewhere.

The time-base-score param in a queue can be set as 'queue' (base score counts only the time the caller is in this queue) or 'system' (base score accounts for the total time of the call).

Agents

Agents have Status *and* States. The Status is the general state of the agent. Statuses are not updated by the system automatically, so they must be set or changed as needed. States are the specific state of an agent with regard to the calls in the queue. States are dynamic and are updated by the system based on the progress of a agent in a call. The reason for separating the two is so that an agent can logout (change Status to 'Logged Out') without affecting his current call State (possibly set to 'In a queue call').

If an agent changes his status to Logged Out, any active callback attempts will be halted and the queue will try to place that caller with another agent.

Status only applies to the next call. So for example, if you change user from Available to Available (On Demand) while they are in a call, they will receive one more call when the current one finishes.

Agent Status and States follow:

Agent Status:

String	**Description**
Logged Out	Cannot receive queue calls.
Available	Ready to receive queue calls.
Available (On Demand)	State will be set to 'Idle' once the call ends (not automatically set to 'Waiting').
On Break	Still Logged in, but will not receive queue calls.

Agent State:

String	**Description**
Idle	Does nothing, no calls are given.
Waiting	Ready to receive calls.
Receiving	A queue call is currently being offered to the agent.
In a queue call	Currently on a queue call.

Type

Callback

Available

While an agent's State is 'Waiting', calls will be directed to them. Whenever an agent completes one of those calls, their State is set back to 'Waiting'.

Available (On Demand)

This is the same as the regular 'Available' Status, except that when the call is terminated, the agent's State is set to 'Idle'. This means the agent won't receive additional calls until his State is changed to 'Waiting'.

uuid-standby

This is used when agents call into the system and wait to receive a calls.

Sample Dialplan to use this function

```
<extension>
  <condition field="destination_number" expression="^(4099)$">
    <action application="set" data="transfer_after_bridge=4099"/> <!-- Remove this if you just want to get a
single call
    <action application="sleep" data="300"/> <!-- Small delay for safety needs -->
    <action application="set" data="res=${callcenter_config(agent set uuid ${caller_id_number}@${domain_name}
'${uuid}')}" />
    <action application="set" data="res=${callcenter_config(agent set type ${caller_id_number}@${domain_name}
'uuid-standby')}" />
    <action application="set" data="res=${callcenter_config(agent set status ${caller_id_number}@${domain_name}
'Available (On Demand)')}" />
    <action application="set" data="res=${callcenter_config(agent set state ${caller_id_number}@${domain_name}
'Waiting')}" />
    <action application="set" data="cc_warning_tone=tone_stream://%(200,0,500,600,700)"/>
    <action application="answer" />
    <action application="playback" data="$${hold_music}"/>
  </condition>
</extension>
```

No Answer

If you define the max-no-answer for an agent, and that agent fails to answer that many calls, then the agent's Status will changed to 'On Break'.

Rejecting Calls

Rejecting a call does *not* act as a 'no-answer'.

A delay can be added before calling an agent who has just rejected a call from the queue by setting 'reject_delay_time' on an agent.

Do not disturb

An agent who is set to "do not disturb" can have a delay added before he is offered his next call by using the 'busy_delay_time' parameter on the agent.

Example Dialplan

Agent Login/Logout Example

Adapt the following dialplan to meet your needs. For example, you can change the contact info if you use this to login from different workstations, or from a number on the PSTN.

```
<extension name="agent_login">
  <condition field="destination_number" expression="^agent-login$">
    <action application="set" data="res=${callcenter_config(agent set status ${caller_id_number}@${domain_name}
'Available')}" />
    <action application="answer" data=""/>
    <action application="sleep" data="500"/>
    <action application="playback" data="ivr/ivr-you_are_now_logged_in.wav"/>
    <action application="hangup" data=""/>
  </condition>
</extension>

<extension name="agent_logoff">
  <condition field="destination_number" expression="^agent-logoff$">
    <action application="set" data="res=${callcenter_config(agent set status ${caller_id_number}@${domain_name}
'Logged Out')}" />
    <action application="answer" data=""/>
    <action application="sleep" data="500"/>
    <action application="playback" data="ivr/ivr-you_are_now_logged_out.wav"/>
    <action application="hangup" data=""/>
  </condition>
</extension>
```

Zero-out for voicemail

```
<extension name="callcenter">
...
  <action application="bind_digit_action" data="inqueue,0,exec:transfer,1000 XML default,aleg,self"/>
  <action application="digit_action_set_realm" data="inqueue"/>
  <action application="set" data="bridge_pre_execute_aleg_app=clear_digit_action"/>
  <action application="set" data="bridge_pre_execute_aleg_data=all"/>
...
  <action application="callcenter" data="example> />
</extension>
```

API Commands

callcenter_config

agent

Add a new agent into the system

```
callcenter_config agent add [agent name] [type(Callback)]
```

Update Agent value

```
callcenter_config agent set [key
(contact|status|state|type|max_no_answer|wrap_up_time|ready_time|reject_delay_time|busy_delay_time)] [agent name]
[value]
```

Delete an agent

```
callcenter_config agent del [agent name]
```

List agents

```
callcenter_config agent list [agent_name]
```

Get uuid of the agent who is bridged to a member

```
callcenter_config agent get uuid [agent_name]
```

tier

Add a new tier mapping an agent to a queue. Note that only Ready should be used in case of an problem.

```
callcenter_config tier add [queue name] [agent name] [[level]] [[position]]
```

Update tier value

```
callcenter_config tier set [key(state|level|position)] [queue name] [agent name] [value]
```

Delete a tier

```
callcenter_config tier del [queue name] [agent name]
```

List tiers

```
callcenter_config tier list
```

queue

Load a queue off the in memory xml config file

```
callcenter_config queue load [queue_name]
```

Unload a queue settings

```
callcenter_config queue unload [queue_name]
```

Reload a queue settings

```
callcenter_config queue reload [queue_name]
```

List queues settings

```
callcenter_config queue list
```

List agents with a tier associated to the specified queue. If [status] is specified, only list agents with a given status.

```
callcenter_config queue list agents [queue_name] [status] [state]
```

List callers present in the queue.

```
callcenter_config queue list members [queue_name]
```

List tiers associated to a specific queue.

```
callcenter_config queue list tiers [queue_name]
```

Return the total number of queues.

```
callcenter_config queue count
```

Return the number agents with a tier associated to the specified queue. If [status] is specified, only count agents with a given status.

```
callcenter_config queue count agents [queue_name] [status]
```

Return the number of callers present in the queue (no of callers waiting in a queue + number of callers bridged with an agent).

```
callcenter_config queue count members [queue_name]
```

Return the number of tiers associated to the specified queue.

```
callcenter_config queue count tiers [queue_name]
```

JSON API Commands

JSON API was added in

> ⚠ FS-8799 - Jira project doesn't exist or you don't have permission to view it.

and currently it support the following commands:

Listing agents

```
json {"command": "callcenter_config", "format": "pretty", "data": {"arguments":"agent list"}}
```

Listing queues

```
json {"command": "callcenter_config", "format": "pretty", "data": {"arguments":"queue list"}}
```

Listing agents for a queue

```
json {"command": "callcenter_config", "data": {"arguments":"queue list agents", "queue_name":"support@default"}}
```

Listing members(callers) for a queue

```
json {"command": "callcenter_config", "data": {"arguments":"queue list members", "queue_name":"
support@default"}}
```

Listing tiers for a queue

```
json {"command": "callcenter_config","data": {"arguments":"queue list tiers", "queue_name":"support@default"}}
```

Listing members

```
json {"command": "callcenter_config","data": {"arguments":"member list"}}
```

Listing tiers

```
json {"command": "callcenter_config","data": {"arguments":"tier list"}}
```

Variables

cc_export_vars

Export variables to the b-leg(s) of call center (the agents)

This is necessary because mod_callcenter originates B-leg calls in its own separate thread, therefore it has no access to variables set in the A-leg (as 'bridge' would).

Example usage:

```
<action application="set" data="hold_music=local_stream://example_moh"/>
<action application="set" data="origination_caller_id_name=Call Center"/>
<action application="set" data="origination_caller_id_number=9000"/>
<action application="set" data="cc_export_vars=hold_music,origination_caller_id_name,
origination_caller_id_number"/>
<action application="callcenter" data="9000@callcenter"/>
```

cc_moh_override

Overrides the queue's default Music On Hold.

Example usage from mail list:

```
<action application="set" data="cc_moh_override=/var/sounds/custom_moh.wav"/>
<action application="set" data="cc_moh_override=/var/sounds/custom_moh.mp3"/>
<action application="set" data="cc_moh_override=file_string:///var/sounds/custom_moh.wav!/var/sounds/custom_moh.
mp3"/>
<action application="set" data="cc_moh_override=tone_stream://%(2000,4000,440,480)"/>
```

cc_base_score

Adds the specified amount to the caller's base score, potential putting him in front other callers in the queue.

cc_exit_keys

Caller can exit the queue by pressing this key.

cc_outbound_cid_name_prefix

Adds prefix to the Caller ID Name of the caller.

cc_outbound_announce

Playback specific audio, or an array of audios, to the agent prior to bridging the member.

cc_bridge_failed_outbound_announce

Playback specific audio to the agent if we can't bridge him because member hangup the call just before the bridge occurs. You can play a busy tone here or a custom audio, for example:

```
<action application="set" data="cc_bridge_failed_outbound_announce=tone_stream://%(250,250,425);loops=3"/>
<action application="callcenter" data="support@default"/>
```

cc_warning_tone

This variable is only valid for agents in 'uuid-standby' mode. It plays the specified tone when a call is sent to the agent.

cc_record_filename

Contains the filename of the call recording *if* a record-template was provided in the queue's configuration. (**read-only**)

cc_side

Contains the leg side of the call. Can be either member or agent. (**read-only**)

cc_member_uuid

Contains the unique callcenter member uuid (Different from the member session uuid) (**read-only**)

cc_member_session_uuid

Contains the member session uuid. (Different from the member_uuid) (**read-only**)

cc_agent

Contains the agent who accepted the call from the queue. (**read-only**)

cc_queue_answered_epoch

Contains the epoch time that the agent answered the call. (**read only**)

cc_queue_terminated_epoch

Contains the epoch time that the bridge with the agent was terminated. (**read-only**)

cc_queue_joined_epoch

Contains the epoch time that the caller joined the queue and started waiting. (**read-only**)

cc_queue_canceled_epoch

Contains the epoch time when a caller leaves the queue and aborts the call. (**read-only**)

cc_agent_bridged

Contains a boolean value indicating if this call was successfully bridged or not. We may call the agent and the member can hangup just before bridging them. (**read-only**)

Events

Actions

Here are a few example events generated by mod_callcenter.

agent-status-change

When an agent's Status changes, this event is generated with the agent's new Status.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Agent: 1000@default
CC-Action: agent-status-change
CC-Agent-Status: Available
```

agent-state-change

Every time an agent's State changes, this event is generated.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Agent: 1000@default
CC-Action: agent-state-change
CC-Agent-State: Receiving
```

agent-offering

Every time a caller is presented to an agent (before he answers), this event is generated.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Agent: AgentNameHere
CC-Action: agent-offering
CC-Agent-System: single_box
CC-Member-UUID: 453324f8-3424-4322-4242362fd23d
CC-Member-Session-UUID: 600165a4-f748-11df-afdd-b386769690cd
CC-Member-CID-Name: CHOUINARD MO
CC-Member-CID-Number: 4385551212
```

bridge-agent-start

When an agent is connected, this event is generated. NOTE: the channel variables, including, for example, Event-Date-Timestamp are present as well as the callcenter values.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Action: bridge-agent-start
CC-Agent: AgentNameHere
CC-Agent-System: single_box
CC-Agent-UUID: 7acfecd3-ab50-470b-8875-d37aba0429ba
CC-Agent-Called-Time: 10000
CC-Agent-Answered-Time: 10009
CC-Member-Joined-Time: 9000
CC-Member-UUID: 453324f8-3424-4322-4242362fd23d
CC-Member-Session-UUID: c6360976-231c-43c6-bda7-7ac4c7d1c125
CC-Member-CID-Name: Their Name
CC-Member-CID-Number: 555-555-5555
```

bridge-agent-end

When an agent is disconnected, this event is generated. NOTE: the channel variables, including, for example, Event-Date-Timestamp are present as well as the callcenter values.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Action: bridge-agent-end
CC-Agent: AgentNameHere
CC-Agent-System: single_box
CC-Agent-UUID: 7acfecd3-ab50-470b-8875-d37aba0429ba
CC-Agent-Called-Time: 10000
CC-Agent-Answered-Time: 10009
CC-Bridge-Terminated-Time: 10500
CC-Member-Joined-Time: 9000
CC-Member-UUID: 453324f8-3424-4322-4242362fd23d
CC-Member-Session-UUID: c6360976-231c-43c6-bda7-7ac4c7d1c125
CC-Member-CID-Name: Their Name
CC-Member-CID-Number: 555-555-5555
```

bridge-agent-fail

When an agent originate fail, this event is generated. NOTE: the channel variables, including, for example, Event-Date-Timestamp are present as well as the callcenter values.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Action: bridge-agent-fail
CC-Hangup-Cause: CHECK FS HANGUP CAUSE
CC-Agent: AgentNameHere
CC-Agent-System: single_box
CC-Member-UUID: 453324f8-3424-4322-4242362fd23d
CC-Member-Session-UUID: c6360976-231c-43c6-bda7-7ac4c7d1c125
CC-Member-CID-Name: Their Name
CC-Member-CID-Number: 555-555-5555
```

members-count

This event is generated every time the queue count api is called *and* anytime a caller enters or leaves the queue.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Action: members-count
CC-Count: 1
CC-Selection: Single-Queue
```

member-queue-start

Joining the queue triggers this event, allowing you to track when callers enter the queue.

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Action: member-queue-start
CC-Member-UUID: 453324f8-3424-4322-4242362fd23d
CC-Member-Session-UUID: b77c49c2-a732-11df-9438-e7d9456f8886
CC-Member-CID-Name: CHOUINARD MO
CC-Member-CID-Number: 4385551212
```

member-queue-end

This is generated when a caller leaves the queue. Different lengths of queue-specific times are reported in seconds. There are two values for CC-Cause: 'Terminated' and 'Cancel'.

'Terminated' means the call ended *after* talking to an agent. Here is an example:

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Action: member-queue-end
CC-Hangup-Cause: CHECK FS HANGUP CAUSE
CC-Cause: Terminated
CC-Agent-Called-Time: 10000
CC-Agent-Answered-Time: 10009
CC-Member-Joined-Time: 9000
CC-Member-Leaving-Time: 10100
CC-Member-UUID: 453324f8-3424-4322-4242362fd23d
CC-Member-Session-UUID: b77c49c2-a732-11df-9438-e7d9456f8886
CC-Member-CID-Name: CHOUINARD MO
CC-Member-CID-Number: 4385551212
```

If we get a hangup from the caller *before* talking to an agent, the CC-Cause will be 'Cancel'. CC-Cause-Reason will contain the reason of the drop between NONE (no specific reason), TIMEOUT (caller has been waiting more than the timeout), NO_AGENT_TIMEOUT (caller has been waiting more than the no_agent_timeout), and BREAK_OUT (caller abandoned). Here's an example:

```
Event-Subclass: callcenter::info
Event-Name: CUSTOM
CC-Queue: support@default
CC-Action: member-queue-end
CC-Member-Joined-Time: 9000
CC-Member-Leaving-Time: 10050
CC-Cause: Cancel
CC-Cancel-Reason: TIMEOUT
CC-Member-UUID: 453324f8-3424-4322-4242362fd23d
CC-Member-Session-UUID: e260ffd0-a731-11df-9341-e7d9456f8886
CC-Member-CID-Name: Marc O Robinson
CC-Member-CID-Number: 5145551212
```

Sample Scripts

Python

This a very simple script that you can use to monitor all callcenter events.

```
#!/usr/bin/env python

import sys

import ESL
con = ESL.ESLconnection("127.0.0.1", "8021", "ClueCon")

if not con.connected():
    print("ERROR: connection failed!")
    sys.exit(1)

con.events("PLAIN", "ALL")

while con.connected():
    e = con.recvEventTimed(1000)
    if e:
        name = e.getHeader("Event-Name")
        if name == 'CUSTOM':
            subclass = e.getHeader("Event-Subclass")
            if subclass == 'callcenter::info':
                print('->>' + name + '|' + subclass + '\n')
                print(e.serialize())
```

Lua

Lua Script to announce members position

This is the script to place in *$PREFIX/scripts*:

```
-- callcenter-announce-position.lua
-- Announce queue position to a member in a given mod_callcenter queue.
-- Arguments are, in order: caller uuid, queue_name, interval (in milliseconds).
api = freeswitch.API()
caller_uuid = argv[1]
queue_name = argv[2]
mseconds = argv[3]
if caller_uuid == nil or queue_name == nil or mseconds == nil then
    return
end
while (true) do
    -- Pause between announcements
    freeswitch.msleep(mseconds)
    members = api:executeString("callcenter_config queue list members "..queue_name)
    pos = 1
    exists = false
    for line in members:gmatch("[^\r\n]+") do
        if (string.find(line, "Trying") ~= nil or string.find(line, "Waiting") ~= nil) then
            -- Members have a position when their state is Waiting or Trying
            if string.find(line, caller_uuid, 1, true) ~= nil then
                -- Member still in queue, so script must continue
                exists = true
                api:executeString("uuid_broadcast "..caller_uuid.." ivr/ivr-you_are_number.wav aleg")
                api:executeString("uuid_broadcast "..caller_uuid.." digits/"..pos..".wav aleg")
            end
            pos = pos+1
        end
    end
    -- If member was not found in queue, or it's status is Aborted - terminate script
    if exists == false then
        return
    end
end
```

Name this script for example *callcenter-announce-position.lua*. This script accepts three arguments:

1. UUID of the call.
2. Callcenter's queue name.
3. How periodically to announce in milliseconds.

First, script checks for empty arguments and terminates if one is not defined. However no type checking is done. Then entering an endless loop, at start we wait before first announce. In *members* we get the output of *callcenter_config queue list members*.

Next we initialize two variables. As the resulting listing from callcenter_config does not contain any autonumbering, we will count the lines. And this is better, because *members* will contain members which have *Aborted* or *Bridged* state, which must not be counted.

The second variable is for checking if the member with this *uuid* is present in list and need announces again.

Iterating thru each line from *members* we first check if the member's call state is in *Waiting* or *Trying* state. The *Trying* state means that this member is the next which will get an agent to bridge. In queue only one member can be in *Trying* state. As an agent can have a long time to answer (may be more time than period we specified), member must be assured it is first in line.

Next we check that member with given uuid is present in current line and therefore we found it's position and can announce the position. Notice forth parameter of the *string.find* function. By default this option is false, meaning, the second parameter is interpreted as regex pattern, which in case of *uuid* will interfere with the "-" symbol. Setting forth parameter two true, we will get a complete character by character compare.

If *uuid* is found, play the position. Using uuid_broadcast we can pause moh sound and play the audio file after which moh sound will continue to play. uuid_broadcast has a forth parameter which indicate to what leg we want to play the position. Position must hear only member which initiated the call, and therefore is considered as *aleg*.

As the member with this uuid were found, the *exists* variable is set to true, for allowing this script to continue saying position.

Then we increase *pos* variable by one to prepare for next position. After which the *exists* variable is checked. If *exists* variable is true, meaning that member is still in queue wating for an agent, the loop will start over. If *exists* is false, this means that member was bridget to an agent, or disconnected from queue, or is not in queue. In this case the script must end.

We have to call this script before entering callcenter module. For example in the dialplan:

```
<extension name="callcenter-example">
        <condition field="destination_number" expression="^callcenter$" break="on-false">
                <!-- limit 3 calls to this destination number per 1 second, otherwise give congestion message --
>
                <action application="limit" data="hash inbound ${destination_number} 3/1 !
NORMAL_CIRCUIT_CONGESTION"/>
        </condition>
        <condition>
                <!-- play a message before entering the queue. -->
                <action application="playback" data="ivr/ivr-welcome.wav"/>
                <action application="playback" data="ivr/ivr-one_moment_please.wav"/>
                <!-- hangup after successful bridge to agent -->
                <action application="set" data="hangup_after_bridge=true"/>
                <!-- queue caller -->
                <action application="set" data="result=${luarun(callcenter-announce-position.lua ${uuid}
example@default 10000)}"/>
                <action application="callcenter" data="example@default"/>
                <!-- if no agent was reached and using max-wait-time - send to general voicemail -->
                <action application="playback" data="ivr/ivr-please_state_your_name_and_reason_for_calling2.wav"
/>
                <action application="answer"/>
                <action application="set" data="skip_greeting=true"/>
                <action application="set" data="skip_instructions=true"/>
                <action application="voicemail" data="default ${domain_name} callcenter"/>
                <action application="hangup"/>
        </condition>
</extension>
```

A note about *uuid_broadcast*. This command was corrected in FreeSWITCH Version 1.2.0-rc2+git~20120807T123541Z~c0626e6801. Before this git version, after playing file, the members call was pulled out from callcenter.

This script is started for each calling party with corresponding uuid of the call. In this case we know to whom which position must be played. So, even moh sound is the same for all members in queue, each one hears its position only.

This script can be easily modified to announce other messages to members with different interval. Using this method is not quite accurate. The first problem occurs when members are being disconnected and come back to its position. In this case other members can hear its queue position increased, but this is rarely. This script does not use score to sort members.

See Also

- mod_dptools: callcenter
- mod_fifo
- https://github.com/gonicus/fsqueuemon - Web status monitor for mod_callcenter queues and agents