Park & Retrieve

About

This example implements parking slots; each slot holds one caller.

Note that the feature implemented by this dialplan logic is different from the features that the park and valet_park dialplan tools provide. The park application will place a caller on hold (on the system), and you must bridge to or transfer the caller to effectively retrieve the call. The valet_park application allows you to place a call on hold (on the system) and make the call retrievable by dialing a certain number.

The implementation described below provides the ability to use the same destination number to trigger holding the call in a particular FIFO OR to trigger retrieval of a call from a particular FIFO, depending on the state of the FIFO. Different from simply using park or valet_park, this dialplan controls the use of multiple FIFOs as holding slots for calls, with a one-to-one mapping between calls "parked" and unique FIFOs; only one caller can be on hold in any given FIFO at a time. If a call is on hold in a given FIFO, running this dialplan for that FIFO will result in retrieval of the call. See below for details.

Introduction

If you call this extension:

1. If the slot is empty, you park yourself in the slot.
2. If the slot is holding a caller, the caller will be removed from the slot and delivered to you - you and the caller will be connected.

If you transfer a caller to this extension:

1. If the slot is empty, the transferred caller will be placed in the slot.
2. If the slot is holding a caller already, the transferred caller and parked caller will be connected (the parked caller being removed from the slot).

This behavior is implemented by using FIFOs. The fact that only one caller is allowed in a FIFO at a time is controlled by this dialplan, which uses the FIFO count of callers.

Having this feature in a single extension (and single destination) is great for making a BLF/speed-dial button on a SIP phone which points to this extension. Then you have a single button to control parking and un-parking.

In this example, the regular expression matching the destination for this extension is "slot\d+" so that you may have any number of slots (i.e. you may have slot1, slot2, slot3, ..., slot100, etc.).

You may define FIFOs (e.g. define FIFOs named slot1, slot2, and slot3) in autoload_configs/fifo.conf.xml.

```
<extension name="parking_slots">
   <condition field="destination_number" expression="^(slot\d+)$" break="on-false">
     <!-- Get count of callers in the specified parking slot (FIFO) -->
     <action inline="true" application="set" data="slot_count=${fifo(count $1@$${domain})}"/>
     <action inline="true" application="set" data="slot_count=${slot_count:-9:2}"/>
   </condition>
   <condition field="${slot_count}" expression="^\:0$" break="always">
     <!-- FIFO settings for parking FIFOs -->
     <action application="unset" data="fifo_chime_list"/>
     <action application="set" data="fifo_chime_freq=0"/>
     <!-- FIFO is empty, so park the caller: -->
     <action application="fifo" data="${destination_number}@$${domain} in undef local_stream://moh"/>
     <!-- FIFO has a caller, so un-park the caller: -->
     <anti-action application="fifo" data="${destination_number}@$${domain} out nowait"/>
   </condition>
 </extension>
```

See Also

- mod_fifo
- mod_dptools: park
- mod_valet_parking