

Mod_voicemail_ivr

mod_voicemail_ivr provide a audio navigation system to a backend voicemail system.

WARNING: Some specs here might change in the next months following discussions (2011-12-24)

```
voicemail_ivr <check> profile domain [id]
```

```
For pre-auth: set voicemail_authorized=true
```

Contents

- [1 Description](#)
- [2 Configuration](#)
 - ◆ [2.1 Example](#)
 - ◇ [2.1.1 apis](#)
 - ◇ [2.1.2 menus](#)
 - [2.1.2.1 phrases](#)
 - [2.1.2.2 keys](#)
- [3 Developer Informations](#)
 - ◆ [3.1 VoiceMail Backend](#)
 - ◇ [3.1.1 mod_voicemail vm_fldb API](#)
 - ◇ [3.1.2 IMAP](#)
 - ◆ [3.2 VoiceMail IVR Engine](#)
- [4 Missing features](#)

Description

The VoiceMail IVR application is a new prototype voicemail ivr engine that allow great flexibility for future improvement. The goal was to make it configuration flexible and yet not be restricted in term of feature we can implement in the different interfaces.

It currently used the exact argument as mod_voicemail. Only the check method is implemented at this time.

Configuration

To get you started, you must have the voicemail_ivr.conf.xml and the voicemail_ivr.xml from the default sample file installed. Compile and load the module. You can call it like you call voicemail. You must have mod_voicemail loaded since it used as the default backend.

Example

You can check the sample configuration file [here](#).

Configuration support multiple profiles. Inside those profile, you can configure apis to be used, and menus.

apis

This is a sample API entry to get the list of message from a mailbox. The `msg_list` and other from the sample configuration file are what the actual code lookup to find the correct API to use. Depending of the features you need, you could add new api here and find it inside the code using the name you defined it.

```
<api name="msg_list" value="vm_fsdb_msg_list" />
```

This is basicly the only way this module do get it informations.

menus

This define an actual menu key map and phrases name to be used inside your IVR. Now that the menu name is a direct reference to the in code menu. so in this example, `std_navigator` actually reference an C function. You cannot just create new menu inside the XML file, you must code the actual scenario inside the module itself. But this method allow for creation of new menu without affecting everyone current implementation. This way, we can add new feature and menu, but if you don't want it, you just never use it.

```
<menu name="std_navigator">
  <phrases>
    <phrase name="msg_count" value="message_count@voicemail_ivr" />
    <phrase name="say_date" value="say_date_event@voicemail_ivr" />
    <phrase name="say_msg_number" value="say_message_number@voicemail_ivr" />
    <phrase name="menu_options" value="listen_file_check@voicemail_ivr" />
    <phrase name="ack" value="ack@voicemail_ivr" />
    <phrase name="play_message" value="play_message@voicemail_ivr" />
  </phrases>
  <keys>
    <key dtmf="1" action="skip_intro" variable="VM-Key-Main-Listen-File" />
    <key dtmf="6" action="next_msg" variable="VM-Key-Main-Next-Msg" />
    <key dtmf="4" action="prev_msg" />
    <key dtmf="7" action="delete_msg" variable="VM-Key-Main-Delete-File" /> <!-- Same
    <key dtmf="8" action="menu:std_forward" variable="VM-Key-Main-Forward" />
    <key dtmf="3" action="save_msg" variable="VM-Key-Main-Save-File" />
    <key dtmf="2" action="callback" variable="VM-Key-Main-Callback" />
    <key dtmf="5" action="menu:std_preference" />
    <key dtmf="#" action="return" /> <!-- TODO Might Conflict with future fast-forward
  </keys>
</menu>
```

phrases

The phrase are defined like apis above. msg_count in this example is what hardcoded inside the code, but the value is used instead for the actual playback. This allow to add more phrases inside the default config file to provide different feature, or to have a private feature set, but at the same time not breaking everyone else setup.

```
<phrase name="msg_count" value="message_count@voicemail_ivr" />
```

keys

The keys are what define the different action to be taken while the playback of an IVR.

- ♦ dtmf - Define the key that an action are associated with.
- ♦ action - Action to execute when a matching dtmf is pressed. This name is actually check in condition inside the module itself. Using name like this allow to create new functionality at will without breaking other people implementation. So all the action are specific to that particular menu except for the standard actions below.
 - ◇ Standard actions:
 - menu:<menu name> - Used to go inside a new menu
 - return - Return to the previous menu
 - ♦ variable - The name you use here will be available inside the phrases and will be populated with the DTMF key.

```
<key dtmf="8" action="menu:std_forward" variable="VM-Key-Main-Forward" />
```

Developer Informations

VoiceMail Backend

mod_voicemail_ivr doesn't have any direct access to any type of backend system. All it does is execute API command inside freeswitch to retrieve voicemail information for a user.

mod_voicemail vm_fsdb API

This is the standard backend used for development of the module. It allow direct access to standard voice mail store by mod_voicemail. This method allow simultaneous usage of mod_voicemail and mod_voicemail_ivr. [mod_voicemail vm_fsdb API reference](#)

IMAP

No imap connector exist at this time, but creating one would be rather trivial. All that is required is to reply to those basic API request.

VoiceMail IVR Engine

mod_voicemail_ivr use custom built ivr interface. This might change in the future depending of technology become available, but for the moment, it what allow the best voicemail experience possible. The reason this interface was written was following the inconstancy of the experience with mod_voicemail.

The C code across all the different menu are pretty similar and done in the following order

1. Initialize the IVR system
2. Run initial code that will be static during the whole IVR process
3. Begin IVR loop allowing retry support
 1. Prepare an Event with the content required for playback and and parsing during this loop
 2. Check if some prompt action need to be executed following an action (This allow playback skipping without discarding the received dtmf key)
 3. Playback the different phrases
 4. Check for status of IVR after playback and perform different action
 1. TIMEOUT : Maximum wait time reached for a DTMF to be entered after the playback of all the phrases
 2. INVALID : The DTMF isn't part of the list that in the menu configuration
 3. FOUND : DTMF key match
 1. We now lookup the action associated to that DTMF and execute what related to it...

This allow for great flexibility but keep full feature set of FS API. Playback can always be interrupted using a DTMF key and the right option will be executed. For comparison with mod_voicemail, you can delete lot of message without having to wait for the full prompt of message deleted to playback.

Missing features

- Seek support
- IVR for leaving voicemail to a user