Freedom Fone 2.0

Developers Guide

Dispatcher, Spooler and Application Layer

Louise Berthilson

February 2011

Table of Contents

1 Introduction

The Freedom Fone architecture is designed with modularity in mind, providing clear interfaces between the various components, to facilitate integration of new services and applications.
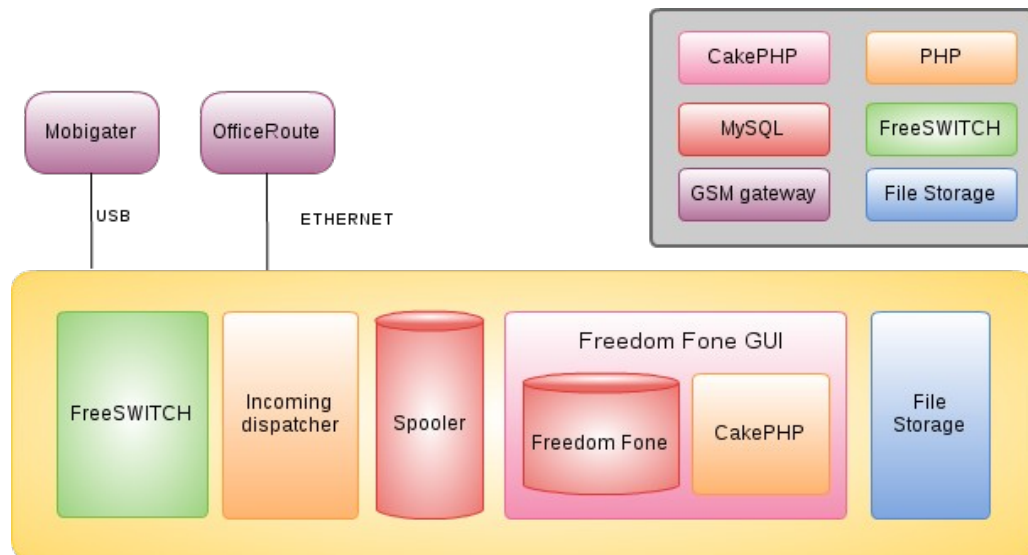
Illustration 1: Architectural design of Freedom Fone 2.0.

Mobigater	A single channel GSM gateway that manages incoming SMS and voice calls.
OfficeRoute	A 4 channel SIP based GSM gateway that manages incoming SMS and voice calls.
FreeSWITCH	PBX that routes incoming/outgoing calls and SMS. Interacts with the incoming dispatcher to feed data to the application layer.
Incoming dispatcher	Analyses FreeSWITCH events, matches data with Freedom Fone applications and stores data into Spooler.
Spooler	A MySQL database with one table per Freedom Fone application. The CakePHP application fetches spooler data regularly by use of a crontab job.
CakePHP	The application layer consisting of CakePHP code and a SQL database. All interaction with the Administrator is done through this interface.
File storage	Uploaded audio files used by Freedom Fone applications. Accessible by both CakePHP and FreeSWITCH.

2 The Incoming dispatcher

2.1 FreeSWITCH events

The incoming dispatcher communicates with FreeSWITCH by subscribing and listening to a set of FreeSWITCH events. In Freedom Fone 2.0, the incoming dispatcher is subscribed to the following events:

Event-Name	Event-Subclass	Application
CUSTOM	leave_a_message	Leave-a-message
CUSTOM	monitor_ivr	IVR monitoring
CUSTOM	gsmopen::dump_event	Monitor Mobigater channels
CUSTOM	officeroute	Incoming SMS from OfficeRoute
MESSAGE		Incoming SMS from Mobigater. "SMS" from Skype
CHANNEL_STATE		Call data records (CDR)
HEARTBEAT		Dispatcher (is FreeSWITCH alive)

Table 1 : FreeSWITCH events for Freedom Fone 2.0

Custom events are events specific to Freedom Fone, such as *leave-a-message and monitor_ivr.*

Message events are standard FreeSWITCH events, such as incoming SMS or Skype chat messages

Channel_State events are CDR details for an incoming call. One single call generators a number of CHANNEL_STATE events.

Heartbeat events are used by FreeSWITCH to announce that FreeSWITCH is alive and running. A heartbeat is sent out every 20 seconds. If the dispatcher does not receive a heartbeat event within 25 seconds, it assumes that FreeSWITCH is not running. The dispatcher keeps running, and tries to re-connect to FreeSWITCH every 5 seconds.

2.2   Dispatcher state machine

This section describes the state machine of the dispatcher.

The illustration below shows the various Rules (orange) and Actions (pink) of the dispatcher logic.
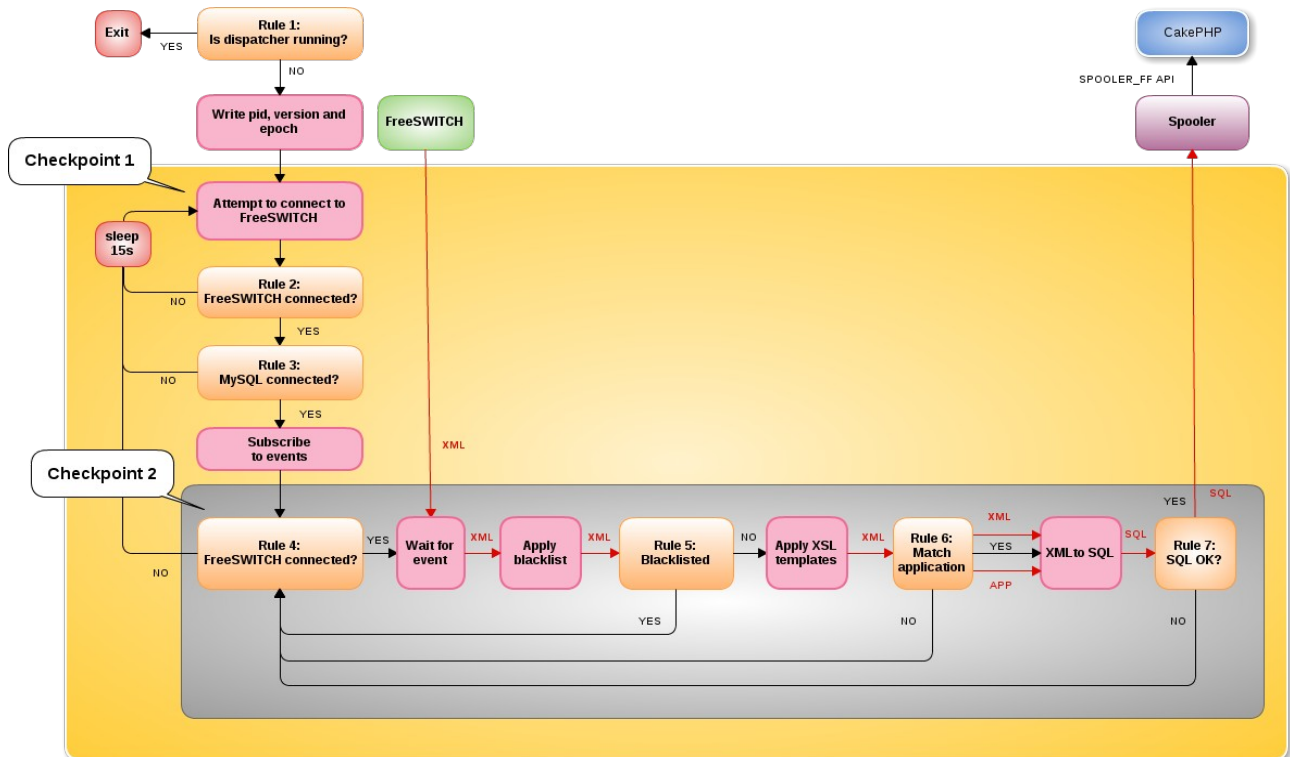
Illustration 2: State machine of incoming dispatcher.

The dispatcher starts by checking if there is any instance of the dispatcher already running. If it finds a running dispatcher, the script exits. If not, it continues to write data (process id, timestamps, version number) to files accessible to the application layer.
Next, the dispatcher tries to connect to FreeSWITCH (Checkpoint 1). If the dispatcher odes not manage to establish a connection to FreeSWITCH (meaning that FreeSWITCH is not running), the scripts waits 15 seconds, and then tries again. If the connection is successfully established, the dispatcher continues and connects to the MySQL database (the spooler database).

Thereafter, the dispatcher subscribes to the events described in Section 2,1. Next, it enters a loop that, as long as a connection to FreeSWITCH is open (Checkpoint 2), waits for incoming events and processes them. If FreeSWITCH is not running (or a connection can not be established), the dispatcher exits the loop and waits 15 seconds until it tries again (Checkpoint 1).

Event arrives from FreeSWITCH to the dispatcher in XML format. The dispatcher applies a filter[1] of blacklisted events to the XML data. If the event is confirmed to be blacklisted, it is ignored, and the dispatcher returns to Checkpoint 2. At the moment, a number of CHANNEL_STATE events[2] are blacklisted.

Next, depending of the Event-Type and Event-Subclass of the event, a XSL template[3] is applied to the XML file, in order to filter out the data needed for the application, and drop the rest of the data.

Next, a set of rules are applied to the XML output in order to match the event with a certain application. If the data can not be matched to an application, the dispatcher returns to Checkpoint 2. This should in theory not happened.

If a matching application is found (default scenario), the event data is parsed to an SQL query and inserted into an application specific table in the spooler (in) database.

The CakePHP application fetches data from the spooler using the spooler API (*spooler_ff.php).* This is done automatically five minutes, by means of a cronjob. The administrator of the system can made a manual refresh from the dispatcher, but using any of the Refresh buttons in the GUI.

The event data is finally inserted in the Freedom Fone database, which is a part of the CakePHP application.

---

[1]   The filter is an XSL template (/usr/local/freedomfone/dispatcher_in/templates/blacklist.xsl)
[2]   Blocked events are: CS_NEW, CS_INIT,CS_EXECUTE, CS_HANGUP,CS_REPORTING
[3]   Located at: /usr/local/freedomfone/dispatcher_in/templates/

3   Poll

The Poll application allows end users to send SMSs to Freedom Fone and participate in polls. The administrator of the system is allowed to Create, Read, Update, and Delete polls.

Several polls can be active at the same time, over the same GSM trunk.

3.1   Event data and dispatcher

1. An incoming SMS reaches a Mobigater/OfficeRoute unit, which transfers the data to FreeSWITCH.

2. The SMS is captured by FreeSWITCH, which creates a **message event** containing the SMS information. If the SMS comes from an Mobigater, a FreeSWITCH native "message" event in created. If the SMS comes from an OfficeRoute unit, a customized message event is created (see Table 1).

3. The event is captured by the incoming dispatcher and is analysed to be matched with an application. Once a matching application is found, the data is parsed to SQL and inserted into the incoming spooler database.
   If the SMS consists of " two separate words", it is matched to the application Poll (table: poll_in). If not, it is classified as an incoming SMS (table: bin).

4. Freedom Fone (GUI) is configured to run a crontab to execute the CakePHP method "refresh". This method can also be executed manually by pointing the browser to http://<your-domain>/freedomfone/polls/refresh

   The refresh method fetches *new entries* from the incoming dispatcher (from poll_in), apply application specific logics, and insert the data in one or more tables (polls, votes) of the Freedom Fone database (freedomfone).

3.2   Poll specific logics

The refresh method for the poll application is responsible for two key actions; register new poll votes and keep track of existing polls status.

The business logics of the refresh method looks for the poll code *(code)* and the poll answer (*chtext*) to determine what to do with the data.
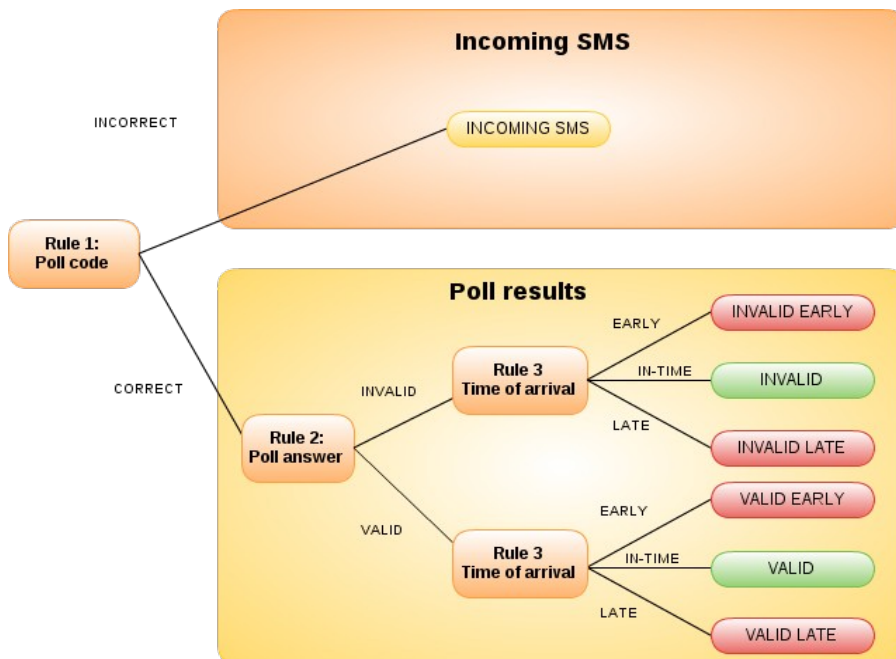
Illustration 3: State machine of incoming SMS and poll votes.

Rule 1: Poll code

1. *No match: Incorrect vote*

If the poll code does not match an existing poll, the vote is classified as Incorrect. The SMS data is stored as "Incoming SMS".

2. *Match: Correct vote*

If the poll code matches an existing poll, it is classified as *Correct vote*.

Rule 2: Poll answer

Rule 2 is only applied to correct votes.

1. No match: Invalid vote

If the poll answer does not match any poll option, the vote is classified as Invalid vote.

2. Match: Valid vote

If the poll code matches an existing poll option for the specific poll, the vote is classified as Valid vote.

Rule 3: Time of arrival

Rule 3 is applied to both valid and invalid votes. A vote is classified as Early, In Time or Late depending on the time of process (when the Freedom Fone application fetches the data from the spooler).

Hence, if the GSM modem has been switched off, or the Freedom Fone server has been down while users have been voting for a poll, their votes will have a timestamp that does not reflect the time when the actual SMS was sent.

1. **Early**

   The vote arrived before the poll was opened. If the vote is valid, it will be registered as "Early vote" for that specific poll option. If the vote is invalid, it will be registered as "Invalid / Early Votes". None of these votes will be a part of the statistics.

2. **In time**

   The vote arrived while the poll was open. Both valid and invalid votes will be a part of the statistics.

3. **Late**

   The vote arrived after the poll was closed. If the vote is valid, it will be registered as "Late vote" for that specific poll option. If the vote is invalid, it will be registered as "Invalid / Late Votes". None of these votes will be a part of the statistics.

The refresh method is also responsible for keeping the status updated for all polls. If the current time is greater (later) than the closing time (*end_time*) for a poll, its status is changed to "closed".

3.3   Database structure

The Poll application uses two database tables, Polls and Votes.

3.3.1   Polls

The **Polls** table handles the individual polls configuration options such as: polls question, SMS code, time of creation, start time, end time and status (active/closed). Each poll is identified with an id.

The Polls table registered invalid votes (correct poll code, incorrect poll option). Invalid votes can be in time (invalid_open), early (invalid_early) or late (invalid_closed).

The *code* must be unique, to be able to match an incoming SMS with the correct poll.

This constraint is managed by PHP, not MySQL.

Field	Type	Key	Null
id	int(10) unsigned	Primary	NO
question	varchar(200)		NO
code	varcher(10)		NO
created	int(10) unsigned		NO
start_time	datetime		YES
end_time	datetime		YES
status	tinyint(4)		YES
invalid_open	int(10) unsigned		YES
invalid_closed	int(10) unsigned		YES
invalid_early	int(10) unsigned		YES

3.3.2 Votes

The **Votes** table matches incoming votes with existing polls, and presents statistics to the user. Each entry in the Votes table represents one poll option (chtext) for a specific poll (poll_id) and the number of votes it has received. Votes are classified as early (votes_early), in-time (chvotes) or late (votes_closed) depending on their time of arrival.

Only valid votes (correct poll code and poll option) are registered in the Votes table. Invalid votes are registered in the Polls table.

Field	Type	Key	Null
id	int(10) unsigned	Primary	NO
poll_id	int(10) unsigned		NO
chtext	varcher(128)		YES
chvotes	int(10) unsigned		YES
votes_closed	int(10) unsigned		YES
votes_early	int(10) unsigned		YES

The *poll_id* (poll) in combination with *chtxt* (poll answer) must be unique. This restriction is implemented in CakePHP (not MySQL).

3.4 Admin functionality

The administrator have access to the following functionality:

1. Create, Read, Update and Delete (CRUD) polls

2. View statistics for each poll

    1. total number of valid and invalid (in-time, early and late) votes

    2. votes per poll option

    3. percentage per poll option

3.5 Logging

All incoming votes to the poll application is logged (poll.log[4]). The votes are logged as follows:

Action	Message
Correct poll code and poll option	Valid {early, open,closed}
Correct poll code, incorrect poll option	Invalid {early, open,closed}
Incorrect poll code	Unclassified

For each action, these parameters are logged:

Field	Value
Message	See table above
Body	SMS body
From	Sender number
Timestamp	Time of arrival to Freeswitch

---

[4] Log are found at: /usr/local/freedomfone/gui/app/tmp/logs/

4   Leave a message

4.1   Event data and dispatcher

1. An incoming voice call enters the system by a GSM gateway (Mobigater/OfficeRoute), and is transferred to the dialplan of Freeswitch.

2. The dialplan executes a Javascript file that manages the Leave-a-Message IVR menu. FreeSwitch stores the recorded message in a file system, which can be local or remote.

3. Freeswitch creates a customized event containing information about the event, such as start time, end time, sender, and file name. The event is captured by the incoming dispatcher, which matches the data with a set of rules, identifies the application, and filter out the necessary data fields. The relevant data fields are inserted into the spooler (table lm_in).

5. Freedom Fone is configured to run a crontab (every 5 min) to execute a refresh method for the Leave-a-message application. The refresh method fetches new data from the spooler and applies application specific logics to the information. Finally is stores the "refined" data in one or more SQL tables (messages) in the CakePHP database (freedomfone).

4.2   Leave-a-message specific logics

The refresh method of Leave-a-Message is responsible for fetching new data from the spooler (lm_in), apply application specific logics to the data, and store the refined data in table *messages.*

Most of the data is a one-to-one matching between the spooler and the CakePHP table. However, the refresh method calculates the length of the messages by calculating the difference between the end time of the call, and the start time of the call.

The refresh method is responsible for logging all incoming messages.

4.3 Admin functionality

4.3.1 Manage messages

A message is defined by a set of attributes. The table below lists the attributes with a description and an indication of what data is editable by the administrator.

Attribute	Description	Editable
Title	Title (defined by administrator)	Yes
Sender	Caller ID	
Rate	1-5	Yes
Category	Reference to entry in Category table	Yes
Tag	Tag names (one or more options)	Yes
File	File name of audio message	
URL	Absolute path to audio message	
New	Read or unread (boolean)	
Status	Inbox or archive	Yes
Length	Length of file in sec	
Created	Timestamp of incoming message	
Modified	Timestamp of last edit	Yes
Instance	Instance of the Leave-a-message service	
User	Reference to entry in Users table	
Comment	Administrators comment to message	Yes
Quick hangup	Yes / No	

The administrator can

1. Read, Update and Delete any message.

2. Sort messages by Status, Title, Rate, Length, Category, Created and Modified.

3. Listen to any message via a built in flash audio player.

4. Download audio file (.mp3)

5. Move messages between Inbox and Archive

6. Navigate between message with pagination

7. Delete or Archive[5] multiple messages at a time, using check boxes.

8. Create, Read, Update and Delete categories.

9. Create, Read, Update and Delete tags.

---

[5]   Archives messages can in the same way be moved back to the Inbox again.

4.3.2 Leave-a-message instances

Freedom Fone 2.0 allows the administrator to create multiple Leave-a-message instances (maximum 20 instances). Each Leave-a-message instance will have and instance id in the range of 2100-2119. This instance id (aka Service) is important when mapping GSM trunks to a specific service. More about that is Section 18 (OfficeRoute).

All files belonging to an instance of a Leave-a-message are stored under:

`/usr/local/freedomfone/gui/app/webroot/freedomfone/leave_message/{instance_id}`

The folder contains three sub-directories:

1. **audio_menu**

   Contains audio files (mp3/wav) of the Leave-a-message IVR

2. **conf**

   100.conf: Text-to-speech options created by user through the GUI
   100_core.conf: default base directory for application (do not edit)
   default.conf: fallback text for Text-to-speech (may be edited)

3. **messages**

   Contains audio files (mp3/wav) and a meta file of each message left by a user

4.3.3 Leave-a-message structure

A Leave-a-Message IVR consists of eight voice messages. Each voice message (Step $1-8$).

Each voice messages can be either:

1. An audio file uploaded by the administrator (mp3/wav) OR
2. A text message set by the administrator synthesised by Cepstral OR
3. A fallback text message (defined in default.conf) synthesised by Cepstral

The prioritization between audio files, text messages and fallback text is illustrated below.
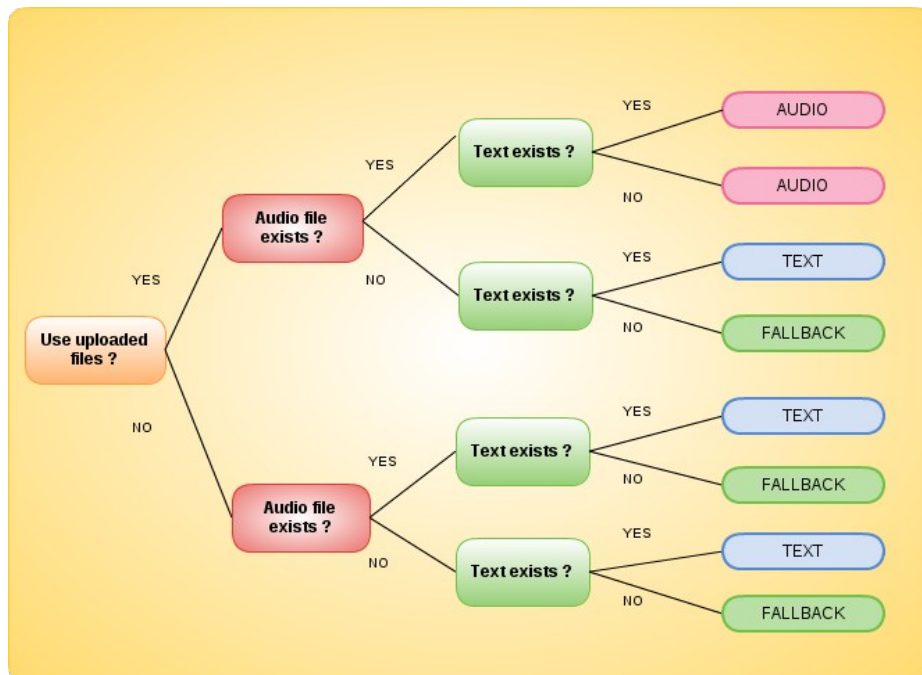
Illustration 4: Prioritization of audio files, text messages and fallback text in Leave-a-message.

The eight audio messages have the following fallback text which can be modified in *default.conf.*

1. Welcome: Welcome to Freedom Fone!

2. Inform: Record your message after the beep. To Finish, Press #

3. Select: To Play... press *. To Delete... press 0. To Save... press 1.

4. Delete: Your message has been deleted.

5. Save: Thank you.

6. Goodbye: Goodbye.

7. Long: You talk to much. Keep it simple.

8. Invalid: Invalid option. Please try again.

4.4   Leave-a-message IVR architecture

The state machine of the IVR menu is designed as according to the illustration below. By using the buttons star (*), hash (#), zero (0) and one (1), the user can manoeuvre between the different stages.

The IVR menu allows the end user to:

1. Record a message

2. Listen to a message

3. Delete a message (that was just recorded)

4. Save a message

The system have a time-out for long messages. The maximum message duration can be set by the administrator for each Leave-a-message instance. The default value is 120 seconds.

Illustration 5: State machine of Leave-a-message IVR.

4.5   Database structure

The Leave-a-message application uses five database tables, messages, categories, tags, messages_tags, and lm_menus.

4.5.1   Table: messages

The table *messages* stores data regarding each message left. It has a one-to-many relationship with *categories*, and a many-to-many relationship with *tags*.

The filename must be unique in order to distinguish one audio file from another. The filename is created by FreeSWITCH.

Field	Type	Key	Null
id	int(11) unsigned	primary	NO
instance_id	int(6)		NO
sender	varchar(200)		NO
title	varchar(200)		NO
rate	smallint(6)		
file	varchar(200)	unique	NO
category_id	int(11) unsigned		
created	int(11) unsigned		NO
modified	int(11) unsigned		
url	varchar(100)		
new	tinyint(1)		
status	tinyint(4)		
length	int(11)		
user_id	int(11)		
comment	varchar(300)		
quick_hangup	varchar(10)		

4.5.2   Table: categories

The categories table contains name and description of categories.

Field	Type	Key	Null
id	int(11) unsigned	primary	NO
name	varchar(100)	unique	NO
longname	varchar(200)		NO

4.5.3   Table: tags

The tags table contains name and description of tags.

Field	Type	Key	Null
id	int(11) unsigned	primary	NO
name	varchar(100)	unique	NO
longname	varchar(200)		NO

4.5.4   Table: messages_tags

This table manages the many-to-many relationship between messages and tags.

Field	Type	Key	Null
id	int(11) unsigned	primary	NO
message_id	int(11) unsigned		
tag_id	int(11) unsigned		

4.5.5   Table: lm_menus

This table stores the user defined options and texts of the Leave-a-message IVR menu for a certain instance.

Field	Type	Key	Null
id	int(11) unsigned	primary	NO
instance_id	smallint(6)	unique	NO
title	varchar(50)		
created	int(11)		NO
modified	int(11)		NO
lmWelcomeMessage	text		
lmInformMessage	text		
lmInvalidMessage	text		
lmLongMessage	text		
lmSelectMessage	text		
lmDeleteMessage	text		
lmSaveMessage	text		
lmGoodbyeMessage	text		
lmForceTTS	tinyint(4)		Default: 0
lmMaxreclen	int(6)		Default: 120
lmOnHangup	varchar(20)		Default: accept

For each instance of Leave-a-message, the administrator can (through the GUI) set:

1. maximum recording length (int lmMaxreclen)
2. Do not use uploaded files (bool lmForceTTS[6])

The variable lmOnHangup refers to the action of a user hanging up after leaving a message without pressing # and follow the instructions how to save the messages.

By default, lmOnHangup is set to 'accept', which implies that the system allows quick hangup (the messages are stored). Even though quick hangup is accepted, the advanced functionality with store, save, delete is still active.

To ignore calls that are terminated with quick hangup, the default value of lmOnHangup in table lm_menus must be changed to "delete".

---

[6]   lmForceTTS(true = ignore uploaded files, force TTS; false = use uploaded files)

5 IVR (Interactive voice menus)

Freedom Fone offers the possibility for the Administrator to build customized IVRs based on personal audio files, or synthesized text messages. IVR's can be of two different kinds:

1. Language Selector OR

2. Voice Menu

The system allows the creation of a total of 20 IVRs. These IVR's can be either Language Selectors or Voice Menus.

5.1 Language selector

A language selector is intended to be used as a top level IVR in a nested IVR. It should provide the caller two or more options depending on its language preferences. Ideally, each language branch should be composed of the same services (Voice Menu, Content, Leave-a-message), but in different languages.

A language selector composes of a title (only used in the GUI to identify a selector), two menu instructions, and a set of Services.

The menu instructions are

1. Instructions (For English... press 1. For Swahili....press 2.)

2. Invalid (Invalid option, please try again

A menu instruction can either be an audio file (mp3/wav) or a text message that is synthesised with Cepstral. If both and audio file and a text message is provided, the audio file is selected. If neither an audio file nor a text message is provided, a default fallback message will be played.

A Language Selector can link to 8 different services. A service can either be a Voice Menu, Content, or a Leave-a-message service.

A Language Selector can be created, edited and deleted by the Administrator.

5.2 Voice menus

A Voice Menu consists of:

1. four **menu instructions** (Welcome, Instructions, Goodbye and Invalid)

2. one or more (max 8) pointers to **services**

Just like the Leave-a-message IVR, a Voice Menu menu instruction can be either an audio file, a text message or a fallback text. The same prioritization rules applies on Voice

Menus, except that the option of ignoring uploaded files does not exits.

A Service can either be

1. Leave-a-message OR

2. Voice Menu

3. Content

The administrator can at any time edit a Voice Menu (changing menu instructions and services).

The design of a Voice Menu is illustrated below.
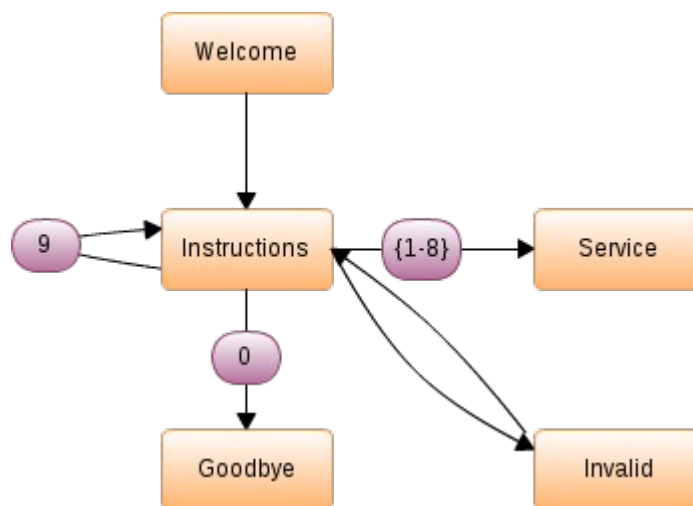
Illustration 6: State machine of a Voice Menu.

Assuming you have created a Voice Menu with 8 services, pressing any number between 1-8 will transfer you to that specific service. Pressing 0, will take you to the Goodbye message and the phone call will be terminated. Pressing 9, will take you back to the instruction message. Pressing any other key, will take you to the Invalid messages, and then back to the Instructions.

Assuming that you have selected a Content Service. Once you have finished listening to the Content, you will be re-directed back to the Instruction message. If you of some reason do not want to finish listening to the Content, you can press any number, and you will be re-directed to the Instruction.

If you select a Leave-a-Message service, the phone call will automatically be terminated after the message has been left. It is not possible to get back in to the Voice Menu, once that you have selected the Leave-a-message Service.

If you select another Voice Menu as a Service, you will be redirected to its Welcome Message.

5.3   Database structure

Field	Type	Null	Comment
instance_id	int(6)	NO	{100-119}
title	varchar(200)	NO	
message_long	Text		TTS message
message_short	Text		TTS message
message_exit	Text		TTS message
message_invalid	Text		TTS message
file_long	Text		Original file name
file_short	Text		Original file name
file_exit	Text		Original file name
file_invalid	Text		Original file name
mode_long	tinyint(1)	Default: 0	1= Ignore uploaded file
mode_short	tinyint(1)	Default: 0	1= Ignore uploaded file
mode_exit	tinyint(1)	Default: 0	1= Ignore uploaded file
mode_invalid	tinyint(1)	Default: 0	1= Ignore uploaded file
switcher_type	varchar(50)	NO	{ivr,lam}
ivr_type	varchar(50)	NO	{ivr,switcher}

5.4   Mapping of services

To map services with digits pressed by the user, the table "Mappings" is used.

The Mappings table requires that ivr_menu_id (unique id for the ivr) in combination with digit (digit pressed by user {1-8}) must be unique.

Each IVR (language selector or Voice Menu) has by default eight entries in the Mappings table. If not all digits are assigned to a Service, the entry (type, node_id, lam_id, ivr_id, instance_id) is empty.

Field	Type	Key	Comment
ivr_menu_id	int(11)	MUL	
type	varchar(10)		{node,ivr,lam}
digit	int(6)	MUL	{1-8}
node_id	int(11)		
lam_id	int(11)		
ivr_id	int(11)		
Instance_id	int(11)		Of node_id or lam_id

5.5   Nested menus

The illustration below shows a four level nested menu that includes all (four) building blocks. Studying the Voice Menus, shows how the commands #0 and #9 affects the IVR depending on where in the tree structure they are executed.

#9 takes the caller back to the top level, while #0 takes her one step up in the tree.
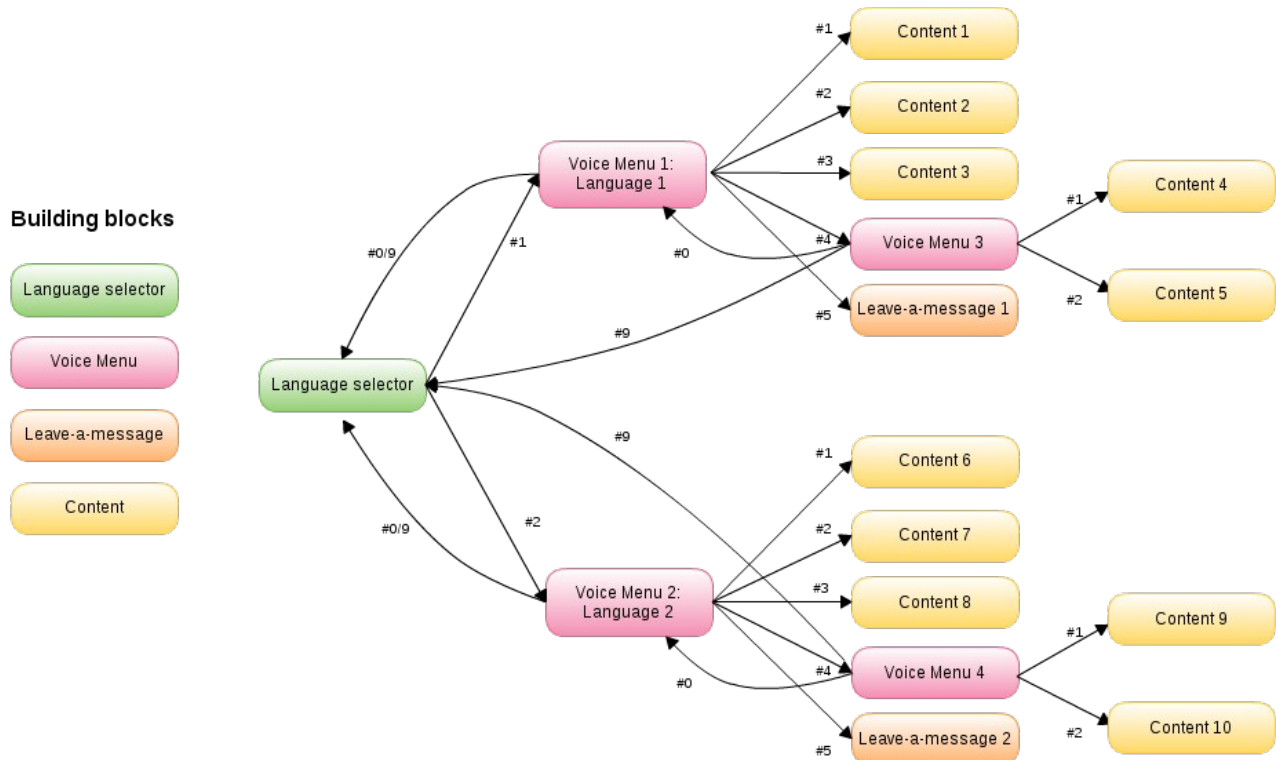
Illustration 7: An example of a nested menu consiting of all types of services.

5.6   Instances

The administrator can create multiple instances of IVR's (Language Selectors and Voice Menus). The system allows up to 20 instances. They will be assigned a instance_id in the range 100 – 119.

All audio files (mp3/wav) that belong to an IVR are found at:

```
/usr/local/freedomfone/gui/app/webroot/freedomfone/ivr/{instance_id}/ivr/
```

Each IVR has its own ivr.xml file, that contains instructions to FreeSWITCH how to route call within that specific IVR. The ivr.xml file is updated every time the IVR is edited. The XML file is located at:

```
/usr/local/freedomfone/gui/app/webroot/freedomfone/ivr/{instance_id}/conf/ivr.xml
```

The system comes pre-equipped with 20 empty folders ready to host IVRs. For example:

```
    root@lupinus:/usr/local/freedomfone/gui/app/webroot/freedomfone/ivr/# tree
.
|-- 100
|    |-- conf
|    `-- ivr
```

When an IVR is deleted, the corresponding conf and ivr folders are emptied from content, but the directories ({instance_id},conf and ivr) are left.

When a new IVR is created, it is being assigned the lowest instance_id available in that range 100-119.

5.7 IVR.XML

The ivr.xml file contains instructions to FreeSWITCH how an IVR should behave as a result of user input.

Freedom Fone 2.0 implements a two-level ivr.xml structure. First, each IVR has its **individual** ivr.xml file, describing only its own behaviour. Secondly, the system has a **main** ivr.xml file that contains all IVR's (language selectors and voice menus) XML code.

The main ivr.xml file is located at:

```
    #/usr/local/freedomfone/gui/app/webroot/xml_curl/ivr.xml
```

The main ivr.xml file is composed by:
1. core.xml[7]
2. all individual ivr.xml files[8]

Whenever an individual IVR is created, added or deleted, the main ivr.xml file is updated.

All ivr.xml files are created by the Vendor :: ivr_xml[9]

---

[7]   Located at: /usr/local/freedomfone/gui/app/webroot/xml_curl/core.xml
[8]   Located at: /usr/local/freedomfone/gui/app/webroot/freedomfone/ivr/{instance_id}/conf/ivr.xml
[9]   Located at: /usr/local/freedomfone/gui/app/vendor/ivr_xml.php

6 Content

A Content item is an audio file associated with a Title. There is no limit in the number of Content items the administrator can create.

Content can be used as a Menu Option in a Language selector or a Voice Menu. A Content item does not belong to a certain IVR, but can be present in multiple IVR.s

Audio files that belongs to Content items are found at:

```
#/usr/local/freedomfone/gui/app/webroot/freedomfone/ivr/nodes
```

A Content item that is present in one or more IVR's, can not be deleted. To delete a Content item, the administrator must first make sure that the Content is not a menu option in any IVR.

Field	Type	Key	Null
id	int(11) unsigned	PRIMARY	NO
title	varchar(200)		NO
file	varchar(100)	UNIQUE	NO
duration	int(11) unsigned		Default: 0
created	int(11) unsigned		NO
modified	int(11) unsigned		Default: 0

6.1 Interaction with FreeSWITCH

The Voice Menu application interacts with FreeSWITCH in two ways;

1. it provides an XML file representing all existing voice menus for that instance

2. it provides audio files and/or text messages to be used in the Voice Menu

23

7 The IVR XML dialplan

7.1 Dialplan actions

Each entry in the XML dialplan has an action defined. The actions that Freedom Fone are taking advantage of, are the following:

- menu-exec-app: execute a FreeSWITCH dialplan application
- menu-sub: load a sub-menu
- menu-back: return to the calling menu level

- menu-top: return to the top level menu

Please see FreeSWITCH Wiki for a full documentation of its IVR functionality[10].

7.2 Freedom Fone example code

This XML file shows the instructions for Voice Menu 4102 (light blue). The Voice menu has five (1-5) menu options and two (0,9) default options common to all IVRs (orange).

Option 1-3 are Content items that are executed by using action "menu-exec-app" together with the parameter "play_and_get_digits" (yellow).

Option 4 is a re-direction to another IVR menu (103), which is is reached by the action "menu-sub" and the name of the IVR (freedomfone_ivr_{instance_id}) as parameter (turquoise).

Option 5 is a transfer to a Leave-a-Message (instance 100) service (green).

For each menu option there is a command that executes a Javascript (pink)[11] that monitors the options made by the caller.

---

[10] FreeSWITCH IVR: http://wiki.freeswitch.org/wiki/Misc._Dialplan_Tools_ivr

[11] For simplicity, this code has only been highlighted for menu option 1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<document type="freeswitch/xml">
 <section name="configuration">
  <configuration name="ivr.conf" description="IVR menus">
   <menus>
    <menu name="freedomfone_ivr_102" tts-engine="cepstral" tts-voice="allison" greet-long="say: Welcome to Swedish." greet-
        short="say: For Cholera, press 1, 2 and 3. For another Voice Menu, press 4. For Leave a message, press 5." inv alid-
        sound="say: Invalid option. Please try again." exit-sound="say: Thank you and good bye." timeout="3000" inter-digit-
        timeout="2000" max-failures="10" max-timeouts="4" digit-len="1">
     <entry action="menu-exec-app" digits="1" param="play_and_get_digits 1 1 1 1000 # $$
        {base_dir}/scripts/freedomfone/ivr/nodes/1292926356_Press_1._cholera_menu_ch_1_symptoms_and_prevention.wav "/>
     <entry action="menu-exec-app" digits="1" param="javascript $${base_dir}/scripts/freedomfone/monitor_ivr/main.js ${uuid}
        'Swedish' '1' '1' 'Node' '${caller_id_number}' '${destination_number}'"/>
     <entry action="menu-exec-app" digits="2" param="play_and_get_digits 1 1 1 1000 # $$
        {base_dir}/scripts/freedomfone/ivr/nodes/1292926376_Press_2._cholera_menu_ch_2_treatment.wav "/>
     <entry action="menu-exec-app" digits="2" param="javascript $${base_dir}/scripts/freedomfone/monitor_ivr/main.js ${uuid}
        'Swedish' '2' '2' 'Node' '${caller_id_number}' '${destination_number}'"/>
     <entry action="menu-exec-app" digits="3" param="play_and_get_digits 1 1 1 1000 # $$
        {base_dir}/scripts/freedomfone/ivr/nodes/1292926427_Press_3._cholera_menu_ch_3_where_to_go.wav "/>
     <entry action="menu-exec-app" digits="3" param="javascript $${base_dir}/scripts/freedomfone/monitor_ivr/main.js ${uuid}
        'Swedish' '3' '3' 'Node' '${caller_id_number}' '${destination_number}'"/>
     <entry action="menu-sub" digits="4" param="freedomfone_ivr_103"/>
     <entry action="menu-exec-app" digits="4" param="javascript $${base_dir}/scripts/freedomfone/monitor_ivr/main.js ${uuid}
        'Swedish' '4' '4' 'IvrMenu' '${caller_id_number}' '${destination_number}'"/>
     <entry action="menu-exec-app" digits="5" param="transfer 2100 XML default"/>
     <entry action="menu-exec-app" digits="5" param="javascript $${base_dir}/scripts/freedomfone/monitor_ivr/main.js ${uuid}
        'Swedish' '5' '1' 'LmMenu' '${caller_id_number}' '${destination_number}'"/>
     <entry action="menu-top" digits="9"/>
     <entry action="menu-back" digits="0"/>
    </menu>
   </menus>
  </configuration>
 </section>
</document>
```

7.3   Content

The FreeSWITCH action "menu-exec-app", together with the parameter "play_and_get_digits" is used to execute a transfer to a Content item (node).

```
<entry action="menu-exec-app" digits="1" param="play_and_get_digits 1 1 1 1000 #
$${base_dir}/scripts/freedomfone/ivr/nodes/1292926356_file1.wav "/>
```

The parameters of the command "play_and_get_digits" are the following:

1. Min: minimum number of digits to fetch (default 1)
2. Max: maximum number of digits to fetch (default 1)
3. Tries: numbers of tries for the sound to play (default 1
4. Timeout: number of milliseconds to wait once the file is done playing before you type a digit (default 1000)
5. Terminators: digit(s) used to end input if less than <min> digits have been pressed (default #)
6. File: sound file to play while digits are fetched
7. Invalid File: sound file to play when digits don't match the regular expression (not in use)

7.4   Voice Menu (IVR)

The FreeSWITCH action "menu_sub" is used to transfer from one IVR to another. The action takes only one parameter, which is the name of the IVR to be transferred to.

```
<entry action="menu-sub" digits="4" param="freedomfone_ivr_103"/>
```

The name of an IVR can be found in the "menu" tag (attribute = "name") of its individual ivr.xml file.

```
<menu name="freedomfone_ivr_103" >
```

7.5   Leave-a-Message

The FreeSWITCH action "menu-exec-app" together with the command "transfer" is used to reach a Leave-a-Message service.

```
<entry action="menu-exec-app" digits="5" param="transfer 2100 XML default"/>
```

In this example, pressing digit 5 will transfer the caller to instance 100 of Leave-a-Message.

The command transfer takes the following parameters:
1. Extension (2 + instance id)
2. Dialplan ("XML")
3. Context ("default")

7.6   Default IVR options

By default, every IVR offers the caller to:

1. Go to the root IVR (#9)

2. Go to the instructions of the current IVR (#0)

These transfers are executed by the actions "menu-top" and "menu-back"

```
<entry action="menu-top" digits="9"/>
<entry action="menu-back" digits="0"/>
```

7.7   Monitoring of Voice Menus

For every selection made by a user that transfers them to a Service (not pressing 0 or 9 or any invalid option), a customized Java-based script[12] is executed.

```
<entry action="menu-exec-app" digits="1" param="javascript
    $${base_dir}/scripts/freedomfone/monitor_ivr/main.js ${uuid} 'Swedish' '1' '1' 'Node'
    '${caller_id_number}' '${destination_number}'"/>
```

---

[12] Located at: /usr/local/freedomfone/freeswitch/scripts/freedomfone/monitor_ivr/main.js

The Javascript is taking the following parameters:

Parameter	Comment	Event
IVR ID	Id of current IVR	FF-IVR-Unique-ID
IVR Name	Title of current IVR	FF-IVR-IVR-Name
IVR Digit	Digit pressed	FF-IVR-IVR-Node-Digit
Service ID	Id of service transferred to	FF-IVR-IVR-Node-Unique-ID
Service type	{Node/Lam/IvrMenu}	FF-IVR-IVR-Node-Service-ID
Caller	Caller's Caller ID	FF-IVR-Caller-ID-Number
Destination number	Number/extension called by user	FF-IVR-Destination-Number

The scripts generates a custom FreeSWITCH event called "monitor_ivr", that the dispatcher processes and makes available to the Freedom Fone application (CakePHP).

Please see Appendix 1 for event details.

8 Audio tricks

8.1 On-the-fly audio conversion (iWatch)

By means of iWatch, the system is monitoring audio folders and converts mp3-files to wav-files, and vice-versa as they are uploaded.

The iWatch component is located at:

```
#/usr/local/freedomfone/audio_bot/iwatch
```

If the number of instances allowed for IVR and Leave-a-Message is increased, the iwatch.xml file must be modified accordingly.

8.2 File duration

After any audio file is uploaded, but before the data is saved to the database, a script called wavDuration($file)[13] is executed, which calculates the duration of the file. The script takes a file in wav-format, and returns the length of the file in seconds.

The script is third party code[14], but has been slightly modified to fit our needs.

8.2.1 Known issues

This scripts assumes that the file is encoding in pure WAV format (which all WAV files should be). The length of a file is calculated based on information stored in the file's header. A non-pure WAV files does not store its header information in the same format as a pure WAV. Hence, the length of the file can not be calculated. Since there are a large number of non-pure WAV formats, we can not support all scenarios.

If the administrator is experience that uploaded audio files are presented as 0seconds long, they should make sure that the audio file are encoded in pure WAV.

---

[13] Located in: usr/local/freedomfone/gui/app/app_controller.php

[14] Source: http://snipplr.com/view/285/read-wav-header-and-calculate-duration/

9 Call data records (CDR)

CDR are used in the application layer for reporting of incoming calls.

9.1 CDR event types

For each incoming call to the system, FreeSWITCH generates a number of CDRs. Each call generators *at least* one event of every type:

- CS_NEW
- CS_INIT
- **CS_ROUTING**
- CS_EXECUTE
- CS_HANGUP
- CS_REPORTING
- **CS_DESTROY**

The dispatcher is programmed to fetch *only* the CS_ROUTING and CS_DESTROY events, and store them in the spooler. All other CHANNEL-STETE events are dropped.

The structure of CS_ROUTING and CS_DESTROY events an incoming call generates, differs depending the channel in use (SIP, GSM, Skype, SkypeOut). Some channels generators multiple CS_ROUTING event, that contains different caller unique information. Hence, when the application layer fetches new CDR from the spooler, some channel unique logic must be applied to the data, before it can be saved in the database.

The CakePHP file /models/cdr.php manages all CDR logic.

The tables below shows what action should be taken for CS_ROUTING and CS_DESTROY events for each channel.

- The action "insert" implies insertion of the event data to the freedomfone database.
- The action "ignore" implies to drop the event data.

If should be noted that calling in via GSM using OfficeRoute generates SIP CDR since the device is a GSM-SIP gateway. Calling in via a Mobigater generates CDR of type GSMopen.

9.1.1 CS_ROUTING

x= service extension {2 = Leave-a-message, 4 = IVR}

iid = instance id {100 - 119}

Channel	Channel-Name	Answer-State	Caller-Destination-Number	Action
GSMOpen	gsmopen/interface{x}	Ringing	5{iid}	Ignore
GSMOpen	gsmopen/interface{x}	Ringing	{x}{iid}	Ignore
GSMOpen	**gsmopen/interface{x}**	**Answered**	**{x}{iid}**	**Insert**
SkypeOut	gsmopen/interface{x}	Ringing	5{iid}	Ignore
SkypeOut	gsmopen/interface{x}	Ringing	{x}{iid}	Ignore
SkypeOut	**gsmopen/interface{x}**	**Answered**	**{x}{iid}**	**Insert**
Skype	**skypeopen/interface{x}**	**Answered**	**{x}{iid}**	**Insert**
SIP	**sofia/internal/{a-z0-9}**	**Ringing**	**{x}{iid}**	**Insert**

9.1.2 CS_DESTROY

Only *one* CS_DESTROY message is created per incoming call and channel type. Hence, all CS_DESTROY messages from the spooler is saved to the freedomfone database.

Channel	Channel-Name	Answer-State	Caller-Destination-Number	Action
GSMOpen	gsmopen/interface{x}	Answered	NULL	Insert
SkypeOut	gsmopen/interface{x}	Answered	NULL	Insert
Skype	skypeopen/interface{x}	Answered	NULL	Insert
SIP	sofia/internal/{a-z0-9}	Answered	NULL	Insert

9.2   Statistics

The statistics page shows the number of incoming calls to each type of service. The statistics are based on Call Data Records. If a CDR for an incoming call is deleted, it will not show up in the statistics.

Each CDR is mapped to an application based on the "application" field in the Cdr table.

The field "application" can have the following (4) values:

1. lam
2. ivr
3. poll
4. bin (incoming SMS)

9.3   Reporting

The Reporting view displayed the activity for each instance of Leave-a-message and IVR (Voice Menu and Language Selector).

Just like the Statistics, the Reporting data is based on CDR entries.

10 Users and phone books

Freedom Fone 2.0 supports *automatic and manual* management of callers (users).

The system automatically creates new users in the system based on incoming calls and SMS. Users can also be manually added, edited and deleted through the GUI.

A users is defined by the following parameters:

Field	Type	Default
name	varchar(50)	Unknown user
surname	varchar(50)	
email	varchar(50)	
skype	varchar(50)	
organization	varchar(50)	
count_poll	int(11) unsigned	
count_ivr	int(11) unsigned	
count_lam	int(11) unsigned	
count_bin	int(11) unsigned	
first_app	varchar(10)	
first_epoch	int(11) unsigned	
last_app	varchar(10)	
last_epoch	int(11) unsigned	
acl_id	int(11) unsigned	0
new	tinyint(4)	1 (true)

Additionally, a user can have {0:n} phone numbers that are stored in the table phone_numbers:

Field	Type	Null
user_id	int(11)	NO
number	varchar(200)	NO

10.1   Validation of data

User data are validated both for automatic and manual insertion. The following validation rules are applied:

Field	Rule	Regex	Allow empty
Name	Only letters, spaces and hyphens	`^[a-zA-Z0-9 -.\']+$`	No
Surname	Only letters, spaces and hyphens	`^[a-zA-Z0-9 -.\']+$`	Yes
Email	Valid email	CakePHP native rule	Yes
Skype	Start a with letter. 6-32 characters long. Allow {0-9,A-Z_.-}	`^[a-zA-Z]{1,1}` `[0-9a-zA-Z.-_\\,]{5,31}$`	Yes
Phone number	May/may not start with a plus sign. 5-33 characters long Allow {+0-9}	`^[+]{0,1}[0-9]{4,25}$`	Yes

10.2   User activity

The Users table monitors users activity, and stores the number of times a user have accesses a certain type of service.

It also stored how (through which type of service) and when a user entered the system. Finally, it keeps track of its last activity (what type of service and when).

The user activity is managed automatically, and can not be edited by the administrator.

10.3   Phone books

The administrator can create/edit/delete phone books. A phone book can contain {0:n} users. A user can be assigned to a phone book via the Edit User view, or through the Edit Phone Book view.

User details can be listed and exported (to CSV) by phone book.

10.4   Access Control Level (ACL)

A user can be given an Access Control Level (White, Black, None). By default, the ACL "None" is given to new users.

In Freedomfone 2.0, the ACL value is not in use. ACL in intended to be used in combination with the Dialer (v 2.5) to allow/disallow users to access the callback service.

11 Processes

When the system boots, both FreeSWITCH and the Dispatcher is started automatically.

From the Health view (GUI), the administrator can monitor the status of FreeSWITCH and the incoming Dispatcher. The dispatcher can be stopped and started manually by the administrator.

Both of these core components can easily be stopped and started from command line whenever needed:

```
#/etc/init.d/freeswitch {start|stop|restart|force-reload}
#/etc/init.d/dispatcher_in {start|stop|restart|reload|force-reload|status}
```

11.1 Dispatcher

When the dispatcher starts, it writes three files that are accessible to CakePHP[15].

1) Process id of dispatcher (pid)
2) Dispatcher version
3) Timestamp (when the dispatcher was started)

11.1.1 Process id (pid)

When the dispatcher starts, it checks if the file PidFile exists (Rule 1). If the file exists, the dispatcher reads it. The file contains the process id (pid) of the previous dispatcher process. Then, the dispatcher checks if there is a process with this pid running (Rule 2).

The presence of this process id in the system, shows that an instance of the dispatcher already is running. Hence, if a process with the same pid is found, the dispatcher exits.

If there no such file (PidFile) is found, it means that the dispatcher was stopped manually (from GUI) and the file has been deleted by the application. In this case, the dispatcher writes its own pid to the PidFile and starts.

If a PidFile exists, but no process with its pid is found, the dispatcher writes its own pid to the PidFile (overwrites whatever that was in the file before) and starts.

---

[15] All files located under: /usr/local/freedomfone/gui/app/webroot/system/{epoch|pid|version}
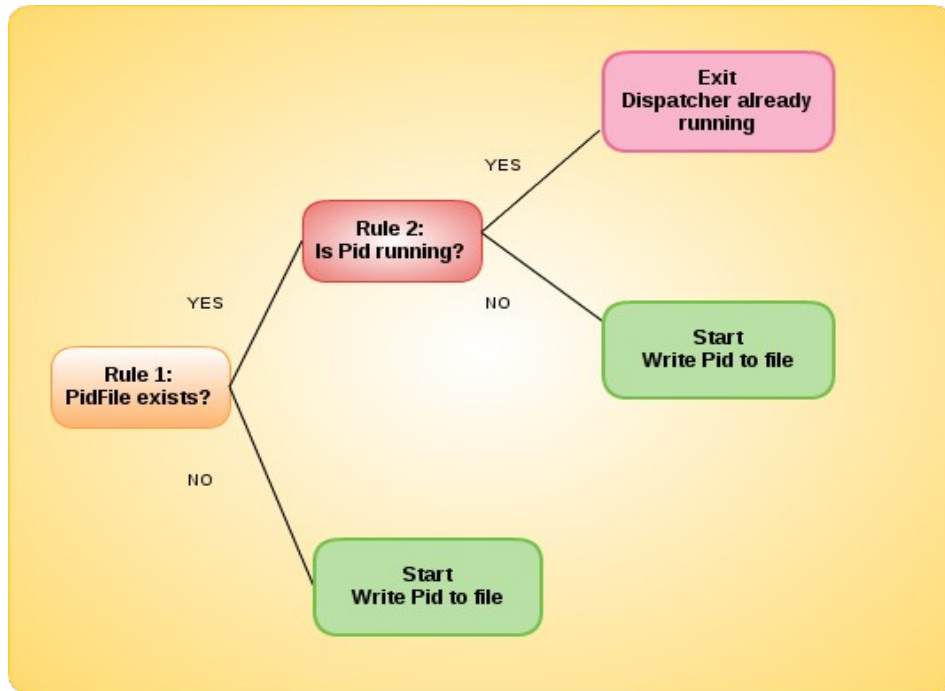
Illustration 8: Dispatcher management of process id's to determine whether an instance of a Dispatcher is running or not.

11.2   Dispatcher timestamps

There are two timestamps that are of importance to the Dispatcher:

- Start_time: time when the dispatcher was started (visible when dispatcher is running)
- Last_seen: last time the dispatcher was seen running (visible when dispatcher is off).

When the dispatcher starts, it writes the current timestamp to the EpochFile.

When the Processes page is reloaded (using the refresh() method), the EpochFile (created by the dispatcher) is read and the application check if there is a process with this pid running (Rule 1).

If a process with this pid is running, the dispatcher is on (green frame). If not, the dispatcher is off (red frame).
Next, the application checks the dispatcher's "status" value in the Processes table.
If the dispatcher is running AND the status is set to "false", the dispatcher has gone through a "unclean" shut-down. The same applies if the dispatcher is off, but the status is set to "true".

An "unclean shut-down" occurs either through server shut-down/reboot, or a manual shut-down from the command line. A clean shut-down of the dispatcher can only be done through the GUI.

To clean up after an "unclean" shut-down, the application takes the following measures (see table below):
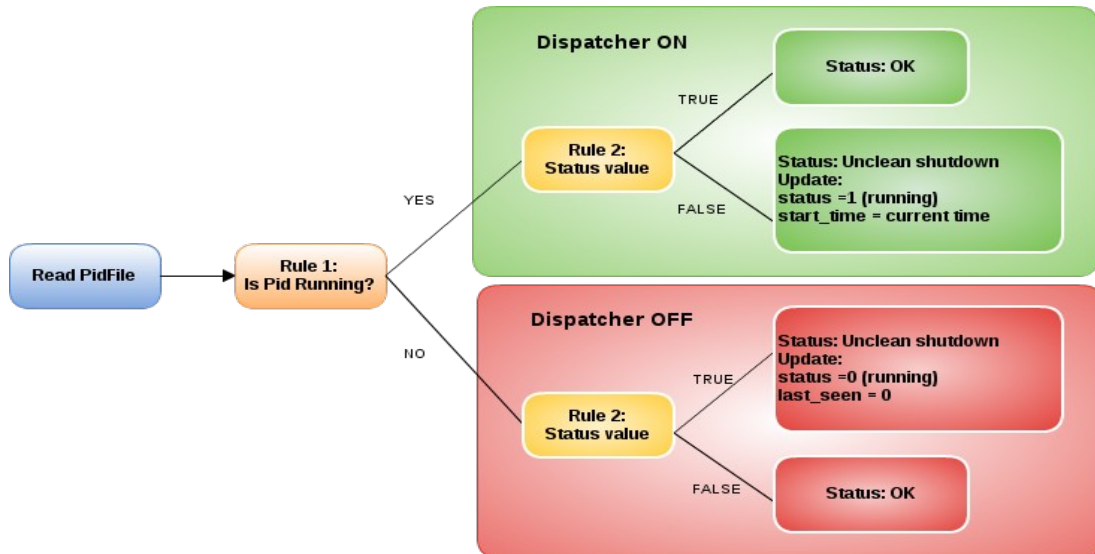

Illustration 9: Dispatcher management of timestamp to determine start- and last seen timestamps.

For a "clean" shutdown, please see Section 11.3-11.4 that explains how to Stop and Start the dispatcher from the GUI.

11.3 Start

When the Administrator press Start process ($id) from the GUI, Cake fetches the corresponding entry in the database (id = $id).

It reads the dispatcher_in.pid file from the system (pid of last known dispatcher), and checks if there is a process running with this pid.

If no process with this pid is running, or the file does not exist, the dispatcher is started using the command found in database as "start_cmd". The start method returns the pid of the new process that runs the dispatcher.

The freedomfone database entry for the incoming dispatcher is updated with start_time, last_seen, pid, status (1 = running), interrupt = false).

Cake sends out a flash message saying either "Incoming dispatcher started", or "The incoming dispatcher is already running", if that is the case.

11.4 **Stop**

When the Administrator press Stop process ($id) from the GUI, Cake fetches the corresponding entry in the database (id = $id).

It reads the dispatcher_in.pid file from the system (pid of last known dispatcher), and checks if there is a process running with this pid. If no such process exists, a flash message is sent out "Incoming dispatcher is not running".

If no pid file is found, a flash message is sent out "No pid found, Contact system admin". This should never happen.

If a process is found, it is killed (kill -9 {pid}), and the dispatcher_in.pid file is deleted (unlink).

The freedomfone database entry for the incoming dispatcher is updated with start_time, last_seen, pid (0), status (0 = not running), interupt = Manual).

The incoming dispatcher is described as follows in the table Processes (freedomfone database).

Field	Value
name	dispatcher_in
status	1
start_cmd	dispatcher_in/dispatcherESL.php —log= /var/tmp/dispatcher_in.log > /dev/null 2>&1 & echo $!
instance_id	100
title	Incoming dispatcher
start_time	{epoch}
last_seen	{epoch}
interupt	{Unmanaged/Manual}
type	run
script	/usr/bin/php

12   Localization

Freedom Fone 2.0 is available in three languages (English, Spanish, and French). It is using CakePHP native l10n support that is PO based.

To localize a string the following PHP code is required:

```
echo __("Hello world!",true);
```

To use the CakePHP built in I18n shell, issue the following command:

```
cd /usr/local/freedomfone/gui/cake/console/
bash cake i18n
```

To extract all string into one single PO file (freedomfone_2.0.pot), do the following (bold text represents user input):

```
Welcome to CakePHP v1.2.4.8284 Console
-----------------------------------------------------------------
App : console
Path: /usr/local/freedomfone/gui/cake/console
-----------------------------------------------------------------
I18n Shell
-----------------------------------------------------------------
[E]xtract POT file from sources
[I]nitialize i18n database table
[H]elp
[Q]uit
What would you like to do? (E/I/H/Q)
> E
What is the full path you would like to extract?
Example: /usr/local/freedomfone/gui/cake/myapp
[Q]uit
[/usr/local/freedomfone/gui/cake/console] > /usr/local/freedomfone/gui/app/
What is the full path you would like to output?
Example: /usr/local/freedomfone/gui/app//locale
[Q]uit
[/usr/local/freedomfone/gui/app//locale] > /usr/local/freedomfone/gui/app/locale

Extracting...
----------------------------------------------------------------
Path: /usr/local/freedomfone/gui/app/
Output Directory: /usr/local/freedomfone/gui/app//locale/
----------------------------------------------------------------
Would you like to merge all translations into one file? (y/n)
[y] > y
What should we name this file?
[default] > freedomfone_2-0
Processing /usr/local/freedomfone/gui/app//app_controller.php...
Processing /usr/local/freedomfone/gui/app//webroot/xml_curl/index.php...
...
...
...
Done.
```

```
----------------------------------------------------------------
I18n Shell
----------------------------------------------------------------
[E]xtract POT file from sources
[I]nitialize i18n database table
[H]elp
[Q]uit
What would you like to do? (E/I/H/Q)
> Q
```

The newly created POT file is located at:

```
/usr/local/freedomfone/gui/app/locale/freedomfone_2-0.pot
```

12.1 Add new language

To add a new language, edit the POT file with your favorite PO editor[16].

Create a few folder with the three letter ISO code (ISO_639-3)[17] of the new language under :

```
/usr/local/freedomfone/gui/app/locale/{xxx}
```

Next, create a few folder named "LC_MESSAGES" :

```
/usr/local/freedomfone/gui/app/locale/{xxx}/LC_MESSAGES
```

Finally, rename your localized po-file to default.po and place it in the LC_MESSAGES folder:

```
/usr/local/freedomfone/gui/app/locale/{xxx}/LC_MESSAGES/default.po
```

Next, we need to add the new language to the Settings GUI of the application. Open the CakePHP configuration file config.php[18] and edit the $config['LANGUAGES'] variable:

```
$config['LANGUAGES']= array(
                    'eng' => __('English',true),
                    'swa' => __('Swahili',true),
                    'esp' => __('Spanish',true)
                                    );
```

In the background, CakePHP applies the selected locale in the AppController (beforeFilter).

---

[16] Gedit is a simple and good tool for editing PO files

[17] List of ISO 639-2 codes: http://www.sil.org/iso639-3/codes.asp

[18] Located at: /usr/local/freedomfone/gui/app/config/config.php

13   Time zones and timestamps

All timestamps are managed in epoch format, which is expressed in GMT. To allow the administrator to see timestamps in local time, we allow her to set the time zone of the Freedom Fone deployment. The time zone can be set under Dashboard → Settings.

This settings does not affect the actual time stamp (epoch) stored for each entry in the database, but only how the administrator sees the time from the application GUI.

14   IP address for streaming media

The Flash-player needs to know the IP address of the deployment to successfully stream audio content. To allow streaming media outside of the local network, a public IP address is required.

By default (upon first installation) the IP address is set to '127.0.0.1' (localhost). Hence the flash-player will only work if the administrator is sitting by the computer that is hosting the Freedom Fone installation. If you are trying to access Freedom Fone from another computer, the Flash player will not appear.

To allow streaming over a Local Area Network (LAN), the administrator must change ip address to one that is reachable over the LAN. The application GUI is facilitating this process, and presents an option to select "Local IP address".

If the Freedom Fone server is connected to the internet and is reachable via a Public (external) IP address, the administrator needs to select "Public IP address" to enable streaming of audio files over the Internet.

15   Refresh methods / cronjob

The application needs to fetch data from the spooler every now and then. That is done automatically by a cronjob that is running every 5 minutes.

The cronjob (crontab.freedomfone) is locate at:

```
/usr/local/freedomfone/cron
```

The cronjob runs the refresh method of the following controllers:
1. office_route
2. polls

3. messages
4. cdr
5. monitor_ivr
6. processes
7. channels
8. bin

```
################################################################################
# Freedom Fone 2010, refresh/cron methods for spoolers
# App: OfficeRoute pop3 daemon
*/5 * * * * root /usr/bin/php /usr/local/freedomfone/office_route/pop3_daemon.php  >/dev/null 2>&1
# App: SMS/Skype polls
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/freedomfone/polls/refresh/cron  >/dev/null 2>&1
# App: Leave a message
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/freedomfone/messages/refresh/cron  >/dev/null 2>&1
# App: Call/Event data records
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/freedomfone/cdr/refresh/cron  >/dev/null 2>&1
# App: Datamining of IVR options
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/freedomfone/monitor_ivr/refresh/cron  >/dev/null 2>&1
# App: System Health
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/freedomfone/processes/refresh/cron  >/dev/null 2>&1
# App: GSMOPEN
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/freedomfone/channels/refresh/cron  >/dev/null 2>&1
# App: Other SMS - The Bin
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/freedomfone/bin/refresh/cron  >/dev/null 2>&1
################################################################################
```

The administrator can also execute the refresh methods manually. Each of the above listed view provides a manual Refresh method:

Main Menu	Page Title	Controller	View
Home	System Overview	cdr	overview
Polls	Polls/(poll results)	polls	index/view
Message Centre	Audio Messages	messages	index
Message Centre	Incoming SMS	bin	index
User Management	Users/Edit User	users	index/edit
System Data	Call data records	cdr	index
System Data	CDR Statistics	cdr	statistics
System Data	Monitoring of Voice Menus	monitor_ivr	index
System Data	Health	processes	index
System Data	GSM Channels	channels/officeroute	index

The (10) instances of the Refresh method calls a system wide function (app_controller.php) that executes the same commands as the cronjob does.

```
$this->requestAction('/polls/refresh/manual');
$this->requestAction('/messages/refresh');
$this->requestAction('/cdr/refresh/manual');
$this->requestAction('/monitor_ivr/refresh/manual');
$this->requestAction('/processes/refresh/manual');
$this->requestAction('/channels/refresh/manual');
$this->requestAction('/bin/refresh/manual');
```

16   Channels

Freedom Fone 2.0 supports the following channels by default:

1. GSMopen (Mobigater)

2. SIP (OfficeRoute)

Additionally, Freedom Fone 2.0 supports Skype and SkypeIn. However, these channels need to be added manually, which is not trivial. The dispatcher and the application layer does however already support Skype and SkypeIn, without any modifications needed.

17   OfficeRoute configuration

To facility the configuration of the OfficeRoute device, we have taken the following measures:

1. Provisioning of a configuration file (backup file) to OfficeRoute (firmware 2.3.11)

2. Configuration of Freedom Fone 2.0 with a default IP address

The backup file is provided with the ISO, but can also be download from the SVN:

https://dev.freedomfone.org/browser/trunk/installer/sekowei/extras/backup_officeroute_2.3.11.tgz

The backup file provides the following configuration:

1. SIP trunk to Freedom Fone

OfficeRoute GUI: Telephony services → Devices → SIP lines

A SIP line pointing to 192.168.1.250 (eth0:0 of Freedom Fone 2.0) at port 5080, with username/password = 1001. This SIP line is associated with a Line ID (24) that will be used later to create "operators".

2. Freedom Fone Services

OfficeRoute GUI: Telephony services → GSM routing → Operator

Two "operators" are created by default (2100 and 4100). The service name and the Description can be named anything, but the Operator number must be the Freedom Fone Service ID ({2,4}{100-119}). The VoIP line must be set to "SIP 24", which represents the Freedom Fone SIP trunk.

The administrator can create more Operators (by pressing the "Plus" icon), and select any operator number (service id) she wish to reach.

Service name		Operator number	VoIP line
IVR 100	☐	4100	SIP - 24
IVR 101	☐	4101	SIP - 24
IVR 102	☐	4102	SIP - 24
IVR 103	☐	4103	SIP - 24
LAM 100	☐	2100	SIP - 24
LAM 101	☐	2101	SIP - 24
LAM 102	☐	2102	SIP - 24

3. Channel mapping

OfficeRoute GUI: Telephony services → GSM routing → GSM

This view maps "Operators" with GSM modules. The OfficeRoute device has four GSM modules (1-4).

By default, GSM module 1 points to the operator "2100" which is the Freedom Fone Leave-a-message service with service id 2100.

GSM module 2 – 4 points to the operator "4100", which is the Freedom Fone IVR with service id 4100.

GSM routing - GSM

GSM device	Service name
GSM module - 1	2100
GSM module - 2	4100
GSM module - 3	4100
GSM module - 4	4100

17.1 Load configuration file

To load the default configuration file of Freedom Fone 2.0, follow the steps below.

1. Go to Administration → Configuration backup → Upload config

2. Select the file to upload (backup_officeroute_2.3.11.tgz), and make sure that the "Don't extract licence" option is checked.

3. Press Save (the bottom-right "floppy drive" icon).

4. Wait a few minutes for the device to reboot and apply the new configuration file.

   The OfficeRoute is rebooted when the Power led is lit Blue and the occupied channels led's are lit Red.

5. Navigate back to the login page of OfficeRoute, and log in again.

Freedom Fone 2.0 comes by default with the IP address 192.168.1.250. This ip address is set as an alias to interface eth0. In this way, the default configuration script of OfficeRoute can be set to point at .250 and is independent of what local IP address Freedom Fone is assigned by its DHCP.

18   Appendix 1: Event data

18.1   Custom events

Event-Name	Event-Subclass	Application
CUSTOM	leave_a_message	Leave-a-message
CUSTOM	monitor_ivr	IVR monitoring
CUSTOM	gsmopen::dump_event	Monitor Mobigater (gsmopen) channels
CUSTOM	officeroute	Incoming SMS from OfficeRoute
MESSAGE		Incoming SMS from Mobigater. Incoming SMS from Skype
CHANNEL_STATE		Call data records (CDR)
HEARTBEAT		Dispatcher (is FreeSWITCH alive)

18.1.1   Leave-a-message

Event-Subclass: leave_a_message
Event-Name: CUSTOM
Core-UUID: 0fb12e62-2d4a-11e0-a17f-b7d076781c4b
FreeSWITCH-Hostname: c10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-03%2012%3A39%3A09
Event-Date-GMT: Thu,%2003%20Feb%202011%2011%3A39%3A09%20GMT
Event-Date-Timestamp: 1296733149199747
Event-Calling-File: mod_spidermonkey.c
Event-Calling-Function: event_construct
Event-Calling-Line-Number: 502
FF-InstanceID: 101
FF-URI: http%3A//192.168.1.10/freedomfone/freedomfone/leave_message/101/messages/
FF-FileID: 1ea2fb34-2f8a-11e0-a29a-b7d076781c4b
FF-CallerID: 1008
FF-CallerName: OPENVZ10
FF-StartTimeEpoch: 1296733145280
FF-FinishTimeEpoch: 1296733149200
FF-OnQuickHangup: true

18.1.2   Monitor IVR

Event-Subclass: monitor_ivr
Event-Name: CUSTOM
Core-UUID: 0fb12e62-2d4a-11e0-a17f-b7d076781c4b
FreeSWITCH-Hostname: c10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-03%2012%3A26%3A22
Event-Date-GMT: Thu,%2003%20Feb%202011%2011%3A26%3A22%20GMT
Event-Date-Timestamp: 1296732382209547
Event-Calling-File: mod_spidermonkey.c

```
Event-Calling-Function: event_construct
Event-Calling-Line-Number: 502
```
FF-IVR-Unique-ID: 6a542e88-2f88-11e0-a292-b7d076781c4b
FF-IVR-IVR-Name: Selector%201
FF-IVR-IVR-Node-Digit: 1
FF-IVR-IVR-Node-Unique-ID: 2
FF-IVR-IVR-Node-Service-ID: IvrMenu
FF-IVR-Caller-ID-Number: 1008
FF-IVR-Destination-Number: 4104

Fields marked in bold are coming from the ivr.xml file. The other fields are generated by FreeSWITCH.

18.1.3 GSMopen::dump_event

Event-Subclass: gsmopen%3A%3Adump_event
Event-Name: CUSTOM
```
Core-UUID: f62c06de-3298-11e0-8322-ffb9470851ea
FreeSWITCH-Hostname: sekowei
```
FreeSWITCH-IPv4: 192.168.1.103
```
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-07%2010%3A04%3A07
Event-Date-GMT: Mon,%2007%20Feb%202011%2009%3A04%3A07%20GMT
```
Event-Date-Timestamp: 1297069447921142
```
Event-Calling-File: mod_gsmopen.cpp
Event-Calling-Function: dump_event_full
Event-Calling-Line-Number: 3150
```
interface_name: interface1
interface_id: 1
active: 1
not_registered: 0
home_network_registered: 1
roaming_registered: 0
got_signal: 0
running: 1
imei: 353579016738243
imsi: 240084703497964
controldev_dead: 0
controldevice_name: /dev/ttyACM0
no_sound: 0
```
alsacname: plughw%3A1
alsapname: plughw%3A1
```
playback_boost: 1619.086162
capture_boost: 910.479058
```
dialplan: XML
context: default
destination: 5000
```
ib_calls: 0
ob_calls: 0
ib_failed_calls: 0
ob_failed_calls: 0
interface_state: 0
phone_callflow: 0
```
session_uuid_str: _undef_
```
during-call: false

18.1.4 Officeroute

Generated by test script[19]:

```
Event-Subclass: officeroute
Event-Name: CUSTOM
Core-UUID: 0fb12e62-2d4a-11e0-a17f-b7d076781c4b
FreeSWITCH-Hostname: c10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-03%2012%3A41%3A55
Event-Date-GMT: Thu,%2003%20Feb%202011%2011%3A41%3A55%20GMT
Event-Date-Timestamp: 1296733315250072
Event-Calling-File: mod_spidermonkey.c
Event-Calling-Function: event_construct
Event-Calling-Line-Number: 502
FF-SMS-Sender-Number: %2B1234567
FF-SMS-Receiver-Number: localhost
proto: GSM
epoch: 1296733315000000
Content-Length: 15


[here goes SMS body]
```

Generated by SMS to OfficeRoute:

```
Event-Subclass: officeroute
Event-Name: CUSTOM
Core-UUID: 0fb12e62-2d4a-11e0-a17f-b7d076781c4b
FreeSWITCH-Hostname: c10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-03%2012%3A52%3A01
Event-Date-GMT: Thu,%2003%20Feb%202011%2011%3A52%3A01%20GMT
Event-Date-Timestamp: 1296733921409489
Event-Calling-File: mod_spidermonkey.c
Event-Calling-Function: event_construct
Event-Calling-Line-Number: 502
FF-SMS-Sender-Number: %2B46706506749
FF-SMS-Receiver-Number: 1000
proto: GSM
epoch: 1296733854000000
Content-Length: 8


[here goes SMS body]
```

---

[19] Available at: /usr/local/freedomfone/dispatcher_in/tests/sendSMS.php

Automatic SMS from operator (missed call):

```
Event-Subclass: officeroute
Event-Name: CUSTOM
Core-UUID: 0fb12e62-2d4a-11e0-a17f-b7d076781c4b
FreeSWITCH-Hostname: c10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-03%2012%3A52%3A01
Event-Date-GMT: Thu,%2003%20Feb%202011%2011%3A52%3A01%20GMT
Event-Date-Timestamp: 1296733921409489
Event-Calling-File: mod_spidermonkey.c
Event-Calling-Function: event_construct
Event-Calling-Line-Number: 502
FF-SMS-Sender-Number: undefined
FF-SMS-Receiver-Number: undefined
proto: GSM
epoch: undefined
Content-Length: 20


[here goes SMS body]
```

18.2 Default FreeSWITCH events

18.2.1 Message

Incoming SMS via Mobigater:

```
Event-Name: MESSAGE
Core-UUID: a4e11ece-32be-11e0-af25-4144060e59ac
FreeSWITCH-Hostname: sekowei
FreeSWITCH-IPv4: 192.168.1.103
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-07%2014%3A48%3A22
Event-Date-GMT: Mon,%2007%20Feb%202011%2013%3A48%3A22%20GMT
Event-Date-Timestamp: 1297086502128933
Event-Calling-File: mod_gsmopen.cpp
Event-Calling-Function: sms_incoming
Event-Calling-Line-Number: 3262
proto: SMS
login: interface1
from: %2B46706506749
date: 02/07/11%2014%3A48%3A17%20(%2B0100)
datacodingscheme: default%20alphabet
servicecentreaddress: %2B46708000708
messagetype: 0
subject: SIMPLE%20MESSAGE
during-call: false
Content-Length: 26


[here goes SMS body]
```

Incoming Skype chat message:

```
Event-Name: MESSAGE
Core-UUID: 33a93134-31dc-11e0-8df9-ef94f0b5360a
FreeSWITCH-Hostname: ffdemo12
FreeSWITCH-IPv4: 192.168.1.12
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-06%2015%3A31%3A10
Event-Date-GMT: Sun,%2006%20Feb%202011%2014%3A31%3A10%20GMT
Event-Date-Timestamp: 1297002670512049
Event-Calling-File: mod_skypopen.c
Event-Calling-Function: incoming_chatmessage
Event-Calling-Line-Number: 2426
proto: skype
login: interface1
hint: Louise%20Berthilson
from: berthilson
subject: SIMPLE%20MESSAGE
chatname: %23berthilson/$freedomfone.org%3Bce25007d81498a3c
id: 821
during-call: false

[here goes SMS body]
```

18.2.2   CHANNEL_STATE

```
Event-Name: CHANNEL_STATE
Core-UUID: 0fb12e62-2d4a-11e0-a17f-b7d076781c4b
FreeSWITCH-Hostname: c10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-03%2012%3A33%3A49
Event-Date-GMT: Thu,%2003%20Feb%202011%2011%3A33%3A49%20GMT
Event-Date-Timestamp: 1296732829919942
Event-Calling-File: switch_channel.c
Event-Calling-Function: switch_channel_perform_set_running_state
Event-Calling-Line-Number: 1301
Channel-State: CS_ROUTING
Channel-State-Number: 2
Channel-Name: sofia/internal/1008%40192.168.1.10%3A5060
Unique-ID: 787613c2-2f89-11e0-a296-b7d076781c4b
Call-Direction: inbound
Presence-Call-Direction: inbound
Channel-Presence-ID: 1008%40192.168.1.10
Answer-State: ringing
Channel-Read-Codec-Name: PCMA
Channel-Read-Codec-Rate: 8000
Channel-Write-Codec-Name: PCMA
Channel-Write-Codec-Rate: 8000
Caller-Username: 1008
Caller-Dialplan: XML
Caller-Caller-ID-Name: OPENVZ10
Caller-Caller-ID-Number: 1008
Caller-Network-Addr: 192.168.1.41
Caller-ANI: 1008
Caller-Destination-Number: 4104
Caller-Unique-ID: 787613c2-2f89-11e0-a296-b7d076781c4b
Caller-Source: mod_sofia
Caller-Context: default
```

```
Caller-Channel-Name: sofia/internal/1008%40192.168.1.10%3A5060
Caller-Profile-Index: 1
Caller-Profile-Created-Time: 1296732829909672
Caller-Channel-Created-Time: 1296732829909672
Caller-Channel-Answered-Time: 0
Caller-Channel-Progress-Time: 0
Caller-Channel-Progress-Media-Time: 0
Caller-Channel-Hangup-Time: 0
Caller-Channel-Transfer-Time: 0
Caller-Screen-Bit: true
Caller-Privacy-Hide-Name: false
Caller-Privacy-Hide-Number: false
```

```
Event-Name: CHANNEL_STATE
Core-UUID: 0fb12e62-2d4a-11e0-a17f-b7d076781c4b
FreeSWITCH-Hostname: c10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-03%2012%3A35%3A15
Event-Date-GMT: Thu,%2003%20Feb%202011%2011%3A35%3A15%20GMT
Event-Date-Timestamp: 1296732915399845
Event-Calling-File: switch_channel.c
Event-Calling-Function: switch_channel_perform_set_running_state
Event-Calling-Line-Number: 1301
Channel-State: CS_DESTROY
Channel-State-Number: 12
Channel-Name: sofia/internal/1008%40192.168.1.10%3A5060
Unique-ID: 787613c2-2f89-11e0-a296-b7d076781c4b
Call-Direction: inbound
Presence-Call-Direction: inbound
Answer-State: answered
```

18.2.3 HEARBEAT

```
Event-Name: HEARTBEAT
Core-UUID: 398fd5cc-31db-11e0-817e-914a7b5848e2
FreeSWITCH-Hostname: ffdevel10
FreeSWITCH-IPv4: 192.168.1.10
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2011-02-06%2014%3A53%3A05
Event-Date-GMT: Sun,%2006%20Feb%202011%2013%3A53%3A05%20GMT
Event-Date-Timestamp: 1297000385226097
Event-Calling-File: switch_core.c
Event-Calling-Function: send_heartbeat
Event-Calling-Line-Number: 65
Event-Info: System%20Ready
Up-Time:0%20years,%200%20days,%203%20hours,%2028%20minutes,%2059%20seconds,
%20782%20milliseconds,%209%20microseconds
Session-Count: 0
Session-Per-Sec: 30
Session-Since-Startup: 0
```