

## Mod\_event\_zmq

This module provides on and off-machine connectivity (inproc, IPC, TCP, Multicast) to FreeSWITCH using the [ØMQ library](#). This page is not meant to be an exhaustive reference of how ØMQ can be used, for that we recommend going to the [learning section](#) of the ØMQ site; along with a couple video presentations you will find **The Guide**, which is particularly well written.

This module was written to make integration with FreeSWITCH much easier, more scalable, and available to more languages than [mod\\_event\\_socket](#) or [mod\\_erlang\\_event](#) could effectively provide. The ØMQ library has bindings for over 20 languages, so chances are that you can easily integrate with FreeSWITCH using your favorite.

## Contents

- [1 To Do](#)
- [2 Data Format](#)
- [3 Receiving Events](#)
- [4 Installation Prerequisites](#)

## To Do

- Configuration file
- Access control
- API channel
- Application to call out from dialplan
- XML search bindings (provide configuration via [mod\\_event\\_zmq](#) similar to [mod\\_xml\\_curl](#))

## Data Format

All data interactions with modzmq use JSON. Almost every language has some kind of JSON framework available, and if not JSON is simple enough to parse with a handful of lines of code.

Here's an example JSON encoded event:

```
{
  "Event-Name": "MESSAGE_WAITING",
  "Core-UUID": "43190614-3ab7-43b2-8697-849bad26b9ab",
  "FreeSWITCH-Hostname": "myfreeswitch.example.com",
  "FreeSWITCH-IPv4": "192.168.0.147",
  "FreeSWITCH-IPv6": "fe0:470:1f04:994::2",
  "Event-Date-Local": "2011-03-26 12:33:27",
  "Event-Date-GMT": "Sat, 26 Mar 2011 18:33:27 GMT",
  "Event-Date-Timestamp": "1301164407584572",
  "Event-Calling-File": "mod_voicemail.c",
  "Event-Calling-Function": "message_query_handler",
  "Event-Calling-Line-Number": "3599",
  "MWI-Messages-Waiting": "yes",
  "MWI-Message-Account": "sip:301@216.112.114.147",
  "MWI-Voice-Message": "15/0 (0/0)",
  "Sofia-Profile": "internal",
  "Call-ID": "1d54ee65-f038111b@192.168.0.147"
```

```
}
```

## Receiving Events

The current version of modzmq stands up a TCP PUB endpoint to publish all switch events to any subscribers; this endpoint listens on all IP addresses on port 5556.

Here's a simple Python example client that subscribes to the event publications, printing 100 events before quitting:

```
import zmq
context = zmq.Context()
subscriber = context.socket(zmq.SUB)
subscriber.connect("tcp://myfreeswitch.example.com:5556")
subscriber.setsockopt(zmq.SUBSCRIBE, "")

for count in range(100):
    string = subscriber.recv()
    print string
```

A couple interesting things to note:

You can start the client before or after the server starts, ØMQ will take care of establishing the network connection, and reconnection if the server goes down and up while the client is running. You can also connect any number of clients to the publisher simultaneously, they'll all receive the same messages.

### Warning

The ZMQ Library and fork() don't work together very well. mod\_event\_zmq may core dumps if FreeSWITCH uses fork() for system calls (e.g. sending mail, use voicemail, etc). You can put `<param name="threaded-system-exec" value="true"/>` in switch.conf.xml to overcome this for system calls but this doesn't mean you are safe from crashes.

## Installation Prerequisites

You will need to have libuuid-devel or libuuid-dev package installed on your system before compiling this module.