Events

0. About

Events are generated by

- commands
- changes in status or configuration
- the core and modules of FreeSWITCH
- external applications

...What else?

and the event system is responsible of controlling the distribution of events among (internal and external) subscribers.

For a comprehensive list of events, see Event List.

1. Event structure

An event message has two parts:

- Header section
- Body (optional)

1.1 Header section

Headers are key/value pairs, and contain almost all of the information regarding an event. Most events only have a header section.

The Event headers page shows the list of available headers.
1.1.1 Common event headers

Common headers

There is a minimum amount of information sent for every event (no matter what the type they are). See list below.

Headers common to all events:

<table>
<thead>
<tr>
<th>Field/Header</th>
<th>Type</th>
<th>Description</th>
<th>Example value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core-UUID</td>
<td>common</td>
<td>Unique ID of FreeSWITCH instance, changes on every reboot [1].</td>
<td>11f751fa-47a7-11e2-9f24-bf76d9fc9ea3</td>
</tr>
<tr>
<td>Event-Calling-File</td>
<td>common</td>
<td>Source file that triggered this event.</td>
<td>mod_voicemail.c</td>
</tr>
<tr>
<td>Event-Calling-Function</td>
<td>common</td>
<td>Source function that triggered this event.</td>
<td>sofia_reg_parse_auth</td>
</tr>
<tr>
<td>Event-Calling-Line-Number</td>
<td>common</td>
<td>Source file line that triggered this event.</td>
<td>710</td>
</tr>
<tr>
<td>Event-Date-GMT</td>
<td>common</td>
<td>Date/time including timezone from FreeSWITCH instance at the point the event was triggered.</td>
<td>Fri, 28 Dec 2012 19:08:39 GMT</td>
</tr>
<tr>
<td>Event-Date-Local</td>
<td>common</td>
<td>Local date/time from FreeSWITCH instance at the point the event was triggered.</td>
<td>2012-12-28 19:08:39</td>
</tr>
<tr>
<td>Event-Date-Timestamp</td>
<td>common</td>
<td>Unix epoch time from FreeSWITCH instance at the point the event was triggered. (in microseconds, divide by 1000 to get milliseconds).</td>
<td>1356721719352143</td>
</tr>
<tr>
<td>Event-Name</td>
<td>common</td>
<td>Name of the event. See Classes section.</td>
<td>RE_SCHEDULE</td>
</tr>
<tr>
<td>Event-Sequence</td>
<td>common</td>
<td>Sequential ID of event on the FreeSWITCH instance. This resets to zero after every reboot, and is not unique to each instance. [1]</td>
<td>9274</td>
</tr>
<tr>
<td>FreeSWITCH-IPv4</td>
<td>common</td>
<td>IPv4 address of FreeSWITCH instance.</td>
<td>192.168.0.2</td>
</tr>
<tr>
<td>FreeSWITCH-IPv6</td>
<td>common</td>
<td>IPv6 address of FreeSWITCH instance.</td>
<td>::1</td>
</tr>
</tbody>
</table>
### 1.2 Event body

Events may have a body, carrying additional content generated with the event. It is usually not in the key/value form of headers, and may contain its own native formatting.

If the event does have a body, then

- Content-Length header will be included in the header section, and
- it will contain the size of the payload in bytes (analogous to the HTTP header of the same name).

Events with bodies include MESSAGE, MESSAGE_WAITING, DETECTED_SPEECH, BACKGROUND_JOB, and so on (see Event List). For parsing instructions, see How to parse events section.

The example below is in the format of mod_event_socket's event plain command.

**TODO** Update example because (1) not sure how old this example is (and thus whether it's still relevant), (2) does not contain all common fields, and (3) there are 3 line feeds before the event body, instead of 2 (is this valid?).
### 2. Event hierarchy

#### 2.1 Classes

The below events (i.e., `Event-Name` header values) are defined by the core of FreeSWITCH, and thus they are the main class of events.

See [Event List](#) for their descriptions with examples.

- ADD_SCHEDULE
- API
- BACKGROUND_JOB
- CALL_DETAIL
- CALL_SECURE
- CALL_SETUP_REQ
- CALL_UPDATE
- CDR
- CHANNEL_ANSWER
- CHANNEL_APPLICATION
- CHANNEL_BRIDGE
- CHANNEL_CALLSTATE
- CHANNEL_CREATE
- CHANNEL_DATA
- CHANNEL_DESTROY
- CHANNEL_EXECUTE
- CHANNEL_EXECUTE_COMPLETE
- CHANNEL_GLOBAL
- CHANNEL_HANGUP
- CHANNEL_HANGUP_COMPLETE
- CHANNEL_HOLD
- CHANNEL_ORIGINATE
- CHANNEL_OUTGOING
- CHANNEL_PARK
- CHANNEL_PROGRESS
2.2 Subclasses (or custom events)

A custom event (i.e., where the Event-Name header's value is custom) is a generic class to define events that is specific to a module, a feature, or to a service.

Custom events are used with the Event-Subclass header denoting the event's role and meaning.

Custom events are employed by

- most endpoint modules (see Creating a New Endpoint: Lifecycle of a Session), for example, when a user logs into Verto, or a SIP phone tries to register

- many dialplan applications, such as the conference, voicemail, and call center applications, the fax management in mod_spandsp, and so on. The ivr application also generates custom events, and subclasses describe entering and exiting of menus (see “Events” section in mod_dptools: IVR Menu).
See List of CUSTOM Events for a complete list of subclasses.

### Modules with CUSTOM events

- **mod_sofia** - Sofia-SIP module.
- **mod_dingaling** - Jabber/GoogleTalk Talk integration module.
- **mod_portaudio** - Host machine sound card.
- **mod_conference** - Conference module.
- **mod_event_multicast**
- **mod_skinny** - Skinny (aka SCCP) module.
- **mod_callcenter** - Call queuing application for call center needs.
- **mod_spandsp** - SpanDSP module including T.38 fax support.
- **IVR Menu** - Built IVR Engine events since FreeSWITCH 1.6

### 3. How to process events

The steps depend on what method is used to tap into the event system.

For example, in case of using

- **mod_erlang_event** or **mod_kazoo**, the values are returned as Erlang terms.

- the **Event Socket Library**, the event building blocks are parsed by the programming language used, and the ESLEvent object has a `getBody()` method.

- **mod_event_socket**, one has to parse TCP packets. (See parsing instructions there under 3.5 `event`.)

- **TODO** What else?