Abstract the SIP Handling Using the Call API

Liviu Chircu
2021, Feb 6th
About Me

- opensips developer & maintainer
- Design, Develop & Maintain SIP Platforms

@liviuchircu
Outline

- Introduction
- Architecture
- Writing API Clients
- API Scenarios
- Demo
Introduction
Call API - Scope

- **Manage** SIP call flows from external applications (not a SIP UA!)
- Abstracts the handling of complex SIP scenarios
- Uses advanced in-dialog SIP handling in order to emulate the behaviour of a SIP B2BUA or UAC
Call API - Implementation

- Stand-alone tool
- Written in Golang (1.14+)
- JSON-RPC over WebSocket
  - CLI tools are available
- Modular design
- Highly scalable
  - Asynchronous design
  - Efficient goroutines
Architecture
Call API - Interconnection Diagram

WebSocket API & Events

UDP MI & EVI

SIP

SIP UA

OpenSIPS

SIP UA

Application Layer
Call API - Application Interconnection

JSON-RPC Requests

- Triggered from Application Layer
- Parameters are validated
- Replies are immediately sent

JSON-RPC Notifications

- Triggered from the Call API
- Carries useful payload data
- Updates regarding the status of the call (success, fail, etc.)
Call API - Application Interconnection

```
Call API

Application Layer

WebSocket API Calls

Call API

Application Layer

WebSocket Events

{  "method": "CallStart",  "params": {    "caller": "sip:alice@10.0.0.10",    "callee": "sip:bob@10.0.0.11"  },  "id": "831717ed97e5",  "jsonp": "2.0" }

{  "method": "CallStart",  "params": {    "cmd_id": "b8179f1e-b4e4-4ac7-9990-4bf64f084178",    "status": "CallerAnswered"  },  "data": {    "caller": "sip:alice@10.0.0.10",    "callee": "sip:bob@10.0.0.11"  },  "id": "831717ed97e5",  "jsonp": "2.0" }
```
Call API - OpenSIPS Interconnection

Management Interface
- MI datagrams
- Triggers MI commands
- Subscribes to different events

Event Interface
- Event datagram
- Sends asynchronous events

OpenSIPS

Call API

WebSocket API & Events

UDP MI & EVI
Call API - OpenSIPS Interconnection

Call API: UDP

OpenSIPS: MI Command

UDP

Call API: UDP

OpenSIPS: EVI Event

Liviu Chircu - OpenSIPS Project

- FOSDEM '21 -
Tips on Writing API Clients

- your language needs WebSocket support
  - C, C++, Python, Java, Golang, JavaScript, Rust, Haskell, Erlang…
- JSON-RPC is trivial, may be easier to do by hand
- **concurrently re-use** a WebSocket connection!
- Example Go client:
- What about REST + Webhook support?!
  - On the TODO list!
API Scenarios
Call API - Commands

- **CallStart** - start a call
- **CallEnd** - terminate a call
- **CallHold** - put a call on hold
- **CallUnhold** - resume an on-hold call
- **CallBlindTransfer** - unattended
- **CallAttendedTransfer**
**Call API - Call Hold**

<table>
<thead>
<tr>
<th>App</th>
<th>Call API</th>
<th>OpenSIPS</th>
<th>SIP Caller</th>
<th>SIP Callee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WS:CallHold</td>
<td>MI:call_hold</td>
<td>SIP:INVITE</td>
<td>SIP:INVITE</td>
</tr>
<tr>
<td></td>
<td>WS:CHStart(A)</td>
<td>EVI:CHStart(A)</td>
<td>SIP:200OK</td>
<td>SIP:200OK</td>
</tr>
<tr>
<td></td>
<td>WS:CHStart(B)</td>
<td>EVI:CHStart(B)</td>
<td>SIP:ACK</td>
<td>SIP:ACK</td>
</tr>
<tr>
<td></td>
<td>WS:CHSuccess(A)</td>
<td>EVI:CHSuccess(A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WS:CHSuccess(B)</td>
<td>EVI:CHSuccess(B)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Call API - Call Start

App

Call API

OpenSIPS

SIP Caller

SIP Callee

<table>
<thead>
<tr>
<th>App</th>
<th>Call API</th>
<th>OpenSIPS</th>
<th>SIP Caller</th>
<th>SIP Callee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WS:CallStart</td>
<td>MI:t_uac_dlg</td>
<td>SIP:INVITE</td>
<td>SIP:INVITE</td>
</tr>
<tr>
<td></td>
<td>WS:Started</td>
<td>MI:call_transfer</td>
<td>SIP:200OK</td>
<td>SIP:200OK</td>
</tr>
<tr>
<td></td>
<td>WS:Answered(A)</td>
<td></td>
<td>SIP:REFER</td>
<td>SIP:INVITE</td>
</tr>
<tr>
<td></td>
<td>WS:Answered(B)</td>
<td></td>
<td>SIP:BYE</td>
<td>SIP:200OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIP:200OK</td>
</tr>
</tbody>
</table>
NEW CallOps Module

- Abstracts SIP signaling MI cmds into high-level primitives
  - call_transfer, call_hold, call_unhold
- NOT a B2BUA!
- Injects in-dialog requests
  - Re-INVITE for on-hold
  - REFER for transfer
- Intercepts replies and treats them accordingly
  - Triggers failure or success events based on them
- Commands can be triggered both through script and MI
DEMO: Click-to-dial service

Browser

Call API

OpenSIPS

WebSockets API & Events

MI & EVI

SIP

SIP UA (joe)

SIP UA (donald)

Liviu Chircu - OpenSIPS Project

- FOSDEM ’21 -
Call API - References

- **Source Code**
  https://github.com/OpenSIPS/call-api

- **Command Docs**
  https://github.com/OpenSIPS/call-api/blob/master/docs/Commands.md

- **Call API Blog Post**

- **OpenSIPS CallOps Module Docs**
  https://opensips.org/docs/modules/3.1.x/callops.html
Call API take-aways

Helps build class 5 services on top of OpenSIPS


- Liviu Chircu
  - OpenSIPS Project: www.opensips.org
  - Email: liviu@opensips.org