**webrtc2sip** - Smart SIP and Media Gateway for WebRTC endpoints

**Technical Guide**

_by_

Mamadou DIOP  
diopmamadou [AT] doubango[DOT]org
License

`webrtc2sip` - Smart SIP and Media Gateway for WebRTC endpoints version 2.6.0
Copyright © 2012-2013 Doubango Telecom <http://www.doubango.org>

`webrtc2sip` is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

`webrtc2sip` is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public Licence along with `webrtc2sip`. If not, see <http://www.gnu.org/licenses/>.
## Versioning

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>SVN revision</th>
<th>Authors</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 2, 2012</td>
<td>2.0.0</td>
<td>9</td>
<td>Mamadou DIOP</td>
<td>Initial version</td>
</tr>
</tbody>
</table>
| January 7, 2013   | 2.1.0   | WEBRTC2SIP: 38+ DOUBANGO: 804+ | Mamadou DIOP    | 1. Add support for DTLS-SRTP (rfc5763 and rfc 5764)  
2. Add new command line arguments: --config, --help and --version  
3. Add new xml configuration entries: video-size-pref, enable-rtp-symmetric and srtp-type  
4. Add verify option to xml configuration entry <ssl-certificates /> to allow remote certificates verification.  
5. Fix issues: 35, 36, 37, 39, 41, 42 and 43. |
| January 14, 2013  | 2.2.0   | WEBRTC2SIP: 44+ DOUBANGO: 808+ | Mamadou DIOP    | 1. Adds support for Firefox Nightly  
2. Fix issues: 47, 48                                                                                                                     |
| March 11, 2013    | 2.3.0   | WEBRTC2SIP: 53+ DOUBANGO: 838+ | Mamadou DIOP    | 1. Adds click-to-call service (http://click2dial.org)  
2. Fix issues: 58, 59 and 60                                                                                                               |
| March 26, 2013    | 2.4.0   | WEBRTC2SIP: 64+ DOUBANGO: 856+ | Mamadou DIOP    | 1. Adds support for DTMF relaying  
2. Adds support for TCP/TLS outbound  
3. Fix issues: 64, 66, 70 and 71                                                                                                           |
| May 06, 2013      | 2.5.0   | WEBRTC2SIP: 86+ DOUBANGO: 884+ | Mamadou DIOP    | 1. Adds support for OPUS audio codec  
2. Fix issues: 13, 26, 77, 78, 81, 85, 88                                                                                           |
| June 03, 2013     | 2.5.1   | WEBRTC2SIP: 90+ DOUBANGO: 895+ | Mamadou DIOP    | 1. Add new xml configuration entries: stun-server and enable-icestun.  
2. Fix issues: 62, 92 and 95                                                                                                             |
| October 07, 2013   | 2.6.0   | WEBRTC2SIP: 116+ DOUBANGO: 1002+ | Mamadou DIOP    | 1. Make the server more robust to DDoS attacks  
2. Add new xml configuration entries: max-fds                                                                                         |
Table of Contents

1 Foreword...........................................................................................................5
2 Scope..................................................................................................................5
3 Architecture.......................................................................................................5
  3.1 SIP Proxy module..........................................................................................5
  3.2 RTCWeb Breaker...........................................................................................7
  3.3 Media Coder..................................................................................................9
  3.4 Click-to-Call..................................................................................................9
    3.4.1 SMTP client............................................................................................10
    3.4.2 HTTPS server.......................................................................................10
    3.4.3 Database connector...............................................................................10
    3.4.4 JSON API.............................................................................................10
4 Configuration....................................................................................................10
5 Building source code........................................................................................16
  5.1 Building Doubango IMS Framework............................................................17
  5.2 Building webrtc2sip....................................................................................20
  5.3 Running webrtc2sip....................................................................................21
    5.3.1 Command line arguments....................................................................21
6 Testing the gateway..........................................................................................21
7 Interoperability..................................................................................................21
  7.1 Servers.........................................................................................................21
    7.1.1 Asterisk.................................................................................................21
    7.1.2 FreeSWITCH.........................................................................................22
  7.2 Web Browsers...............................................................................................22
    7.2.1 Google Chrome.....................................................................................22
    7.2.2 Firefox Nightly.......................................................................................22
    7.2.3 Firefox, Safari, IE and Opera...............................................................22
    7.2.4 Ericsson Bowser....................................................................................23
  7.3 JavaScript SIP stacks.....................................................................................23
8 Security issues....................................................................................................24

Table of Figures

Figure 1: Architecture...........................................................................................5
Figure 2: SIP Proxy architecture...........................................................................5
Figure 3: RTCWeb Breaker architecture...............................................................7
Figure 4: Enabling RTCWeb Breaker on sipml5....................................................7
Figure 5: Media Coder architecture.....................................................................9
Figure 6: click-to-call components.....................................................................10

Table of Samples

Sample 1: config.xml...........................................................................................11
1 Foreword

RTCWeb (a.k.a WebRTC) stands for Real-Time Communication and is a new technology being drafted by the World Wide Web Consortium (W3C) and IETF groups. This technology has the ambition to bring native real-time features (audio, video and arbitrary data) to the web browsers without requiring additional plugins.

SIP stands for Session Initiation Protocol and is a signaling protocol defined by the IETF in RFC 3261. SIP is widely used today to manage VoIP (Voice over IP) communication sessions and has been chosen as signaling protocol for Next Generations Networks such as IMS (IP Multimedia Subsystem) or LTE (Long Term Evolution). The protocol has quickly become the de facto standard used to interconnect the IP world (Internet) with the PSTN (circuit-switched telephone networks).

webrtc2sip is a smart and powerful gateway using RTCWeb and SIP to turn your browser into a phone with audio, video and SMS capabilities. The gateway allows your web browser to make and receive calls from/to any SIP-legacy network or PSTN. As an example, you will be able to make a call from your preferred web browser to a mobile or fixed phone.

2 Scope

This technical guide is a reference document explaining why you need webrtc2sip and how to leverage its power.

3 Architecture

The gateway contains four modules: SIP Proxy, RTCWeb Breaker, Media Coder and click-to-call service.

Figure 1: Architecture

The HTML SIP client is any endpoint implementing draft-ibc-sipcore-sip-websocket-06. We highly recommend using sipML5 which is known to work and provide good performances.

3.1 SIP Proxy module

Figure 2: SIP Proxy architecture
The role of the SIP Proxy module is to convert the SIP transport from WebSocket protocol to UDP, TCP or TLS which are supported by all SIP-legacy networks. If your provider or hosted server supports SIP over WebSocket (e.g. Asterisk or Kamailio) then, you can bypass the module and connect the client directly to the endpoint. Bypassing the SIP Proxy is not recommended if you’re planning to use the RTCWeb Breaker or Media Coder modules as this will requires maintaining two different connections.

There are no special requirements for the end server to be able to talk to the Proxy module.

**F1 REGISTER Web Browser -> webrtc2sip (transport WS)**

```
REGISTER sip:proxy.example.com SIP/2.0
Via: SIP/2.0/WS df7jal23ls0d.invalid;branch=Z9hG4b5
From: sip:browse@example.com;tag=abc
To: sip:browse@example.com
Call-ID: abcdefghijklmnopqrstuvwxyz
CSeq: 1 REGISTER
Max-Forwards: 70
Contact: <sip:browse@df7jal23ls0d.invalid;transport=ws>
```

This request contains an invalid IP address in the Contact (df7jal23ls0d.invalid) and Via headers because there is no way for the browser to retrieve its local binding IP:Port address. The transport type is WebSocket (ws). A SIP-legacy server cannot handle this request as the transport is probably not supported and the IP address and port are not valid (not reachable), this is why we need the SIP Proxy module to patch the request before forwarding.

**F2 REGISTER webrtc2sip -> SIP-legacy Network (transport UDP)**

```
REGISTER sip:proxy.example.com SIP/2.0
Via: SIP/2.0/UDP 66.66.66.66:5060;branch=Z9hG4b5;rport
Via: SIP/2.0/TCP 192.168.0.9:55210;branch=Z9hG4b6;ws-hacked=WS
From: sip:browse@example.com;tag=abc
To: sip:browse@example.com
Call-ID: abcdefghijklmnopqrstuvwxyz
CSeq: 1 REGISTER
Max-Forwards: 70
Contact: <sip:browse@66.66.66.66:5060;transport=udp>
```

The Via header is patched to use a well-known protocol (TCP) and to use the IP address and port (192.168.0.9:55210) from which the request has been received (WebSocket connection).

The SIP Proxy adds its own Via header (66.66.66.66:5060) where it’s willing to receive the
response. The same address is used in the Contact header for incoming requests (e.g. INVITE).

Before forwarding the request the SIP Proxy determines the destination address using the following algorithm:

```c
char* dst_host = get_host(request_uri); // dst_host = "proxy.example.com"
int dst_port = 5060;
if(has_route(request)){ // there a route header
    dst_host = get_host(first_route);
    dst_port = get_port(first_route);
}
if((dns_result = dns_srv(dns_naptr(dst_host)))){
    dst_host = get_host(dns_result);
    dst_port = get_port(dns_result);
}
```

### 3.2 RTCWeb Breaker

![RTCWeb Breaker architecture](image)

The RTCWeb specifications make support for ICE and DTLS/SRTP mandatory. The problem is that many SIP-legacy endpoints (e.g. PSTN network) do not support these features. It’s up to the RTCWeb Breaker to negotiate and convert the media stream to allow these two worlds to interop.

For example, FreeSWITCH do not support ICE which means it requires the RTCWeb Breaker in order to be able to connect the browser to a SIP-legacy endpoint.

The RTCWeb Breaker is disabled by default and it’s up to the client to enable it before registering to the server.

To activate the RTCWeb Breaker, the client must include “rtcweb-breaker=yes” as Uri parameter of its AoR (Address of Record). When the module is enabled it acts as a b2bua (back 2 back user agent) by answering to the INVITE and making a new one.

![Enabling RTCWeb Breaker on sipml5](image)

**webrtc2sip** – Smart SIP and Media Gateway for WebRTC endpoints
**F1 REGISTER** web browser -> webrtc2sip (transport WSS)

-- TODO --

**F2 REGISTER** webrtc2sip -> SIP-legacy network (transport UDP)

-- TODO --

**F3 200 OK** SIP-legacy network -> webrtc2sip (transport UDP)

-- TODO --

**F4 200 OK** webrtc2sip -> web browser (transport WSS)

-- TODO --

**F5 INVITE** SIP-legacy endpoint -> SIP-legacy network (transport UDP)

-- TODO --

**F6 INVITE** SIP-legacy network -> webrtc2sip (transport UDP)

-- TODO --

**F7 100 Trying** webrtc2sip -> SIP-legacy network (transport UDP)

-- TODO --

**F8 INVITE** webrtc2sip -> web browser (transport WSS)

-- TODO --

**webrtc2sip** – Smart SIP and Media Gateway for WebRTC endpoints
3.3 Media Coder

The RTCWeb standard defined two MTI (Mandatory To Implement) audio codecs: \textit{opus} and \textit{g.711}. For now there are intense discussions about the MTI video codecs. The choice is between \textit{VP8} and \textit{H.264}. \textit{VP8} is royalty-free but not widely deployed while \textit{H.264 AVC} is not free but widely deployed. Google has decided to use \textit{VP8} in Chrome while Ericsson uses \textit{H.264 AVC} in \textit{Bowser}. Mozilla and Opera Software will probably use \textit{VP8} and Microsoft \textit{H.264 AVC}. As an example, the \textit{Media Coder} will allow to make video calls between Chrome and \textit{Bowser}. Another example is calling a Telepresence system (e.g. Cisco) which most likely uses \textit{H.264 SVC} from Chrome.

The \textit{Media Coder} is enabled using the xml configuration file and requires \textit{RTCWeb breaker} module to be enabled.

3.4 Click-to-Call

This is more a service than a module as it’s a complete SIP \textit{click-to-call} solution based on the three other components. The goal is to allow any person receiving your mails, visiting your website, reading your twitts, watching your Facebook/Google+ profile to call you on your mobile phone with a single click.

The client is hosted at \url{http://click2dial.org/}

A short user guide is available at \url{http://click2dial.org/u/ug.htm}
3.4.1 SMTP client
This component is used to send activation mails for newly registered users. It’s coded from scratch and has no external dependencies.

3.4.2 HTTPS server
For now, the HTTPS server is used exclusively by the JSON API to exchange content between the browser and the click-to-call service. It’s coded from scratch and depends on tinyHTTP (from Doubango VoIP framework).

3.4.3 Database connector
Agnostic API functions to connect to any database used to store users information, configuration… In this beta version, only SQLite is supported. Next release will add support to MySQL and SQL Server.

3.4.4 JSON API
The JSON API is used to authenticate the users and manage their accounts. The documentation will be released soon at http://click2dial.org/doc.htm. On the server-side, the parser is based on Json-Cpp.

4 Configuration
The gateway is configured using an xml file named config.xml and stored in the same folder where the gateway is running.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<config>
  <debug-level>INFO</debug-level>
  <transport>udp;*:10060</transport>
  <transport>ws;*:10060</transport>
  <transport>wss;*:10062</transport>
  <enable-rtp-symmetric>yes</enable-rtp-symmetric>
  <enable-100rel>no</enable-100rel>
</config>
```

webrtc2sip — Smart SIP and Media Gateway for WebRTC endpoints
Sample 1: config.xml

```
<config>
  <debug-level />
  Define the minimum debug-level to display.
  Format: debug-level-value
  Debug-level-value = INFO | WARN | ERROR | FATAL

  <enable-media-coder>no</enable-media-coder>
  <enable-videojb>yes</enable-videojb>
  <video-size-pref>vga</video-size-pref>
  <rtp-buffsize>65535</rtp-buffsize>
  <avpf-tail-length>100;400</avpf-tail-length>
  <srtp-mode>optional</srtp-mode>
  <srtp-type>sdes;dtls</srtp-type>
  <dtmf-type>rfc4733</dtmf-type>
  <codecs>opus;pcma;pcmu;gsm;vp8;h264-bp;h264-mp;h263;h263+</codecs>
  <codec-opus-maxrates>48000;48000</codec-opus-maxrates>
  <stun-server>stun.l.google.com;19302;stun-user;stun-password</stun-server>
  <enable-icestun>yes</enable-icestun>
  <max-fds>65535</max-fds>
  <nameserver>66.66.66.66</nameserver>
  <nameserver>77.77.77.77</nameserver>
  <ssl-certificates/>
    /tmp/priv.pem;
    /tmp/pub.pem;
    /tmp/cacert.pem;
    no
</ssl-certificates>

<!-- ***CLICK-TO-CALL SERVICE*** -->
<transport>c2c;*:;10070</transport>
<transport>c2cs;*:;10072</transport>
<database>sqlite;*</database>
<account-mail>smtps;*:;e.org;465;noreply@e.org;noreply@e.org;mysecret</account-mail>
<account-sip-caller>*;sip:13131313@b.c;13131313;b.c;mysecret</account-sip-caller>
<account-sip-caller>*;sip:13131313@a.c;13131313;a.c;mysecret</account-sip-caller>
</config>
```
Each entry defines a protocol, local IP address and port to bind to.
Format: proto-value;local-ip-value;local-port-value
proto-value: udp | tcp | tls | ws | wss | c2c | c2cs
“ws” protocol defines WebSocket and “wss” the secure version. At least one WebSocket transport must be added to allow the web browser to connect to the server. The other protocols (tcp, tls and udp) are used to forward the request from the web browser to the SIP-legacy network. “C2c” and “c2cs” are used for the click-to-call service and runs on top of HTTP or HTTPS protocols respectively.
local-ip-value: Any valid IP address. Use star (*) to let the server choose the best local IP address to bind to. Examples: udp:*;5060 or ws:*;5061 or wss;192.168.0.10;5062
local-port-value: Any local free port to bind to. Use star (*) to let the server choose the best free port to bind to. Examples: udp:**, ws:*;* or wss:*;5062

This option is used to force symmetric RTP and RTCP streams to help NAT and firewall traversal. It only applies on remote RTP/RTCP as local stream is always symmetric. If both parties (remote and local) have successfully negotiated ICE candidates then, none will be forced to use symmetric RTP/RTCP.
An RTP/RTCP stream is symmetric if the same port is used to send and receive packets. This helps for NAT and firewall traversal as the outgoing packets open a pinhole for the ongoing ones.
Let’s imagine you have a server on public network and a client on private network:
1. Server: Public IP address is 1.1.1.1
2. Client: Private IP address is 2.2.2.2 and public IP address is 1.1.1.2
3. The SDP from the client to the server will contain client’s private IP address (2.2.2.2) which is not reachable
4. The RTP/RTCP packets from the client to the server will be received with source IP address equal to the client’s public IP address (1.1.1.2)
5. If <enable-rtp-symmetric /> option is used then, the server will send RTP/RTCP packets to 1.1.1.2 (learnt from the received packets) instead of 2.2.2.2 which is private.

Indicates whether to enable SIP 100rel extension.

Indicates whether to enable the Media Coder module or not. This option requires the RTCWeb Breaker to be enabled at the web browser level. When the Media Coder is enabled the gateway acts as a b2bua and both audio and video streams are transcoded if the remote peers don’t share same codecs.
<srtp-mode />
Format: srtp-mode-value

srtp-mode-value: none | optional | mandatory

Defines the SRTP mode to use for negotiation when the RTCWeb Breaker is enabled. Please note that only optional and mandatory modes will work when the call is to a WebRTC endpoint.

Based on the mode, the SDP for the outgoing INVITEs will be formed like this:

  - none: profile = RTP/AVP ||| neither crypto lines or certificate fingerprints
  - optional: profile = RTP/AVP ||| two crypto lines if <srtp-type /> includes
  - mandatory: profile = RTP/AVP ||| two crypto lines if <srtp-type /> includes

<video-size-pref />
Format: video-size-pref-value

video-size-pref-value: sqcif | qcif | qvga | cif | hvga | vga | 4cif | svga | 480p | 720p | 16cif | 1080p
Available since: 2.1.0

This option defines the preferred video size to negotiate with the peers. There is no guarantee that the exact size will be used: video size to use = Min (Preferred, Proposed);

<rtp-buffsize />
Format: rtp-buffsize-value

rtp-buffsize-value: Any positive 32 bits integer value. Recommended: 65535.

Code usage:

setsockopt(SOL_SOCKET, SO_RCVBUF, rtp-buffsize-value);
setsockopt(SOL_SOCKET, SO_SNDBUF, rtp-buffsize-value);

Defines the internal buffer size to use for RTP sockets. The higher this value is, the lower will be the RTP packet loss. Please note that the maximum value depends on your system (e.g. 65535 on Windows). A very high value could introduce delay on video stream and it’s highly recommended to also enable videojb option.

<avpf-tail-length />
Format: avpf-tail-length-min;avpf-tail-length-max

avpf-tail-length-min: Any positive 32 bits integer
avpf-tail-length-max: Any positive 32 bits integer

Defines the minimum and maximum tail length used to honor RTCP-NACK requests. This option require the Media Breaker module to be enabled on the web browser size. The higher this value is, the better will be the video quality. The default length will be equal to the number of unrecoverable packet loss. The final value will be at most equal to the maximum defined in the xml file. Unrecoverable packet loss occurs when the b2bua receives an RTCP-NACK for a sequence number already removed (very common when network RTT is very high or bandwidth very low).
### <stun-server />

Format: server-fqdn-value; server-port-value; user-name-value; user-password-value

- **server-fqdn-value**: A valid IPv4/v6 address or host name.
- **server-port-value**: A valid port number.
- **user-name-value**: The login to use for TURN authentication. Use star (*) to ignore.
- **user-password-value**: The password to use for TURN authentication. Use star (*) to ignore.

Defines the STUN/TURN server to use to gather reflexive addresses for the ICE candidates. If no server is defined then, a default one will be used. The default STUN server is numb.viagenie.ca:3478.

### <enable-icestun />

Format: enable-icestun-value

- **enable-icestun-value**: yes | no

Defines whether to use STUN to gather reflexive addresses or not. This option is useful when the server is on a public network or all peers are on the same local network.

### <dtmf-type />

Format: dtmf-type-value

- **dtmf-type-value**: rfc4733 | rfc2833

Available since: 2.4.0

Defines the DTMF type to use when relaying the digits. Requires the RTCWeb Breaker to be enabled. **rfc4733** will send the DTMF digits using RTP packets while **rfc2833** uses SIP INFO.

### <codecs />

Format: codec-name (: codec-name)*

- **codec-name**: opus|pcma|pcmu|amr-nb-be|amr-nb-oa|speex-nb|speex-wb|speex-uwb|g729|g729a|g722|ilbc|h264-bp|h264-mp|vp8|vp9|h263|h263+|theora|mp4v-es

Defines the list of all supported codecs. Only G.711 and G.722 are natively supported and all other codecs have to be enabled when building the Doubango IMS Framework source code.

Each codec priority is equal to its position in the list. First codecs have highest priority.

### <srtp-type />

Format: srtp-type-value; (srtp-type-value)*

- **srtp-type-value**: sdes | dtls

Available since: 2.1.0

Defines the list of all supported SRTP types. Defining multiple values only make sense if the **srtp-mode** value is optional which means we want to negotiate the best one.

Please note that DTLS-SRTP requires valid TLS certificates and source code must be compiled with OpenSSL version 1.0.1 or later.

---

'SDES' plus certificate fingerprints if `<srtp-type />` include 'DTLS'.

`mandatory: profile = RTP/SAVP if `<srtp-type />` is equal to 'SDES' or UDP/TLS/RTP/SAVP if `<srtp-type />` is equal to 'DTLS' | | two crypto lines if `<srtp-type />` is equal to 'SDES' or certificate fingerprints if `<srtp-type />` is equal to 'DTLS'.
Disabling STUN for ICE will speed up the call setup.

<codec-opus-maxrates />
Format: maxrate-playback-value; maxrate-capture-value

Maxrate-playback-value: 8000|12000|16000|24000|48000

Maxrate-capture-value: 8000|12000|16000|24000|48000

Defines the maximum playback and capture rates to negotiate. The final rates to use will be \text{min}(offer, answer). Default value = 48000 for both.

The higher this value is, the better will be the voice quality. The bandwidth usage is proportional to the value. In short: high value = high bandwidth usage = good voice quality.

max-fds

Format: max-fds-value

Available since: 2.6.0

Max-fds-value: Any integer value from 1 to 65535.

Defines the number of file descriptors (FDs) the process is allowed to open. The FDs include the pipes and sockets only.

Setting this value is like running \texttt{ulimit –n max-fds-value} on Linux.

<nameserver />

Format: nameserver-value

Nameserver-value: Any IPv4 or IPv6 address.

Defines additional entries for DNS servers to use for SRV and NAPTR queries. Please note that this option is optional and should be used carefully.

On Windows and OS X the server will automatically load these values using APIs provided by the OS. On Linux, the values come from /etc/resolv.conf. The port must not be defined and the gateway will always use 53.

<ssl-certificates />

Format: private-key-value; public-key-value; cacert-key-value; verify-value

Private-key-value: A valid path to a PEM file.

Public-key-value: A valid path to a PEM file.

Cacert-key-value: A valid path to a certificate authority file. Should be equal to *.

Verify-value: Yes | No. This additional option is only available since version 2.1.0. It indicates whether the connection should fail if the remote peer certificates are missing or do not match. This option only applies to TLS/SIP or WSS and is useless for DTLS-SRTP as certificates are required.

Code usage:

\texttt{SSL_CTX_use_PrivateKey_file(ssl_ctx, private-key-value, SSL_FILETYPE_PEM);}
\texttt{SSL_CTX_use_certificate_file(ssl_ctx, public-key-value, SSL_FILETYPE_PEM);}
\texttt{SSL_CTX_load_verify_locations(ssl_ctx, cacert-key-value, CaPath);}

<database />

Format: db-type-value; db-connection-info-value

Available since: 2.3.0
db-type-value: sqlite | mysql. For now only “sqlite” is supported.

For now this configuration entry is only used for the click-to-call service.

For now this configuration entry is only used for the click-to-call service.

<account-mail />

Format: 

-scheme-value: smtp | smtps
-local-ip-value: A valid local host name or IP address to be used by the SMTP client.
rett use the best value.
-local-port-value: A valid local port number to be used by the SMTP client. Use star (*)
rett use a random value.
-smtp-host-value: A valid host name or IP address of the SMTP server.
-smtp-port-value: A valid port of the SMTP server.
-email-value: Email address used as sender.
-auth-name-value: Authorization name used to authenticate to the SMTP server. Most
urtly the user part of the Public Identity (e.g. 003).
-auth-pwd-value: Password used to authenticate to the SMTP server.

The email account is used to send activation mails to the newly registred users.

<account-sip-caller />

Format: 

-displayname-value: SIP account display name. Optional.
-impu-value: Public Identity. Must be a valid SIP address (e.g. sip:003@example.org).
-impi-value: Private Identity (a.k.a authorization name) for authentication. Most
urtly the user part of the Public Identity (e.g. 003).
-realm-value: SIP domain name (e.g. example.org). Should be same as the domain name in
the Public Identity.
-password-value: SIP authentication password.

The SIP account callers are used to make calls to users by the click-to-call service. The
callers in the config.xml file are globals (shared by all users) and are override
d and a user define one using the JSON API.

5 Building source code

This section explains how to build the project using CentOS 64 but could be easily adapted for
Linux, Windows or OS X.

webrtc2sip gateway depends on Doubango IMS Framework v2.0.

1. Preparing the system

sudo yum update
5.1 Building Doubango IMS Framework

Doubango is an IMS framework and contains all signaling protocols (SIP, SDP, WebSocket…) and media engine (RTP stack, audio/video codecs…) required by webrtc2sip gateway.

The first step is to checkout Doubango 2.0 source code:

```
svn checkout http://doubango.googlecode.com/svn/branches/2.0/doubango doubango
```

1. Building libsrt

libsrt is required.

```
git clone https://github.com/cisco/libsrt/
cd libsrt
CFLAGS="-fPIC" ./configure --enable-pic && make && make install
```

2. Building OpenSSL

OpenSSL is required if you want to use the RTCWeb Breaker module or Secure WebSocket transport (WSS). OpenSSL version 1.0.1 is required if you want support for DTLS-SRTP.

This section is only required if you don’t have OpenSSL installed on your system or using version prior to 1.0.1 and want to enable DTLS-SRTP.

A quick way to have OpenSSL may be installing openssl-devel package but this version will most likely be outdated (prior to 1.0.1). Anyway, you can check the version like this: openssl version.

```
wget http://www.openssl.org/source/openssl-1.0.1c.tar.gz
tar -xvzf openssl-1.0.1c.tar.gz
cd openssl-1.0.1c
./config shared --prefix=/usr/local --openssldir=/usr/local/openssl && make && make install
```

3. Building libspeex and libspeexdsp

libspeex (audio codec) is optional and libspeexdsp (audio processing and jitter buffer) is required.

You can install the devel packages:

```
yum install speex-devel
```

Or build the source by yourself:

```
wget http://downloads.xiph.org/releases/speex/speex-1.2beta3.tar.gz
tar -xvzf speex-1.2beta3.tar.gz
cd speex-1.2beta3
./configure --disable-oggtest --without-libogg && make && make install
```

4. Building YASM

YASM is only required if you want to enable VPX (VP8 video codec) or x264 (H.264 codec).

```
wget http://www.tortall.net/projects/yasm/releases/yasm-1.2.0.tar.gz
```

webrtc2sip — Smart SIP and Media Gateway for WebRTC endpoints
5. Building libvpx

Date: December 1, 2012.

libvpx adds support for VP8 and is optional but highly recommended if you want support for video when using Google Chrome or Mozilla Firefox.

You can install the devel packages:

```bash
sudo yum install libvpx-devel
```

Or build the source by yourself:

```bash
git clone http://git.chromium.org/webm/libvpx.git
cd libvpx
./configure --enable-realtime-only --enable-error-concealment --disable-examples --enable-vp8 --enable-pic --enable-shared --as=yasm
make && make install
```

6. Building libyuv

libyuv is optional. Adds support for video scaling and chroma conversion.

```bash
mkdir libyuv && cd libyuv
./gclient config http://libyuv.googlecode.com/svn/trunk
./gclient sync && cd trunk
make -j6 V=1 -r libyuv BUILDTYPE=Release
make -j6 V=1 -r libjpeg BUILDTYPE=Release
cp out/Release/obj.target/libyuv.a /usr/local/lib
cp out/Release/obj.target/third_party/libjpeg_turbo/libjpeg_turbo.a /usr/local/lib
mkdir --parents /usr/local/include/libyuv/libyuv
cp -rf include/libyuv.h /usr/local/include/libyuv
cp -rf include/libyuv/*.h /usr/local/include/libyuv/libyuv
```

7. Building opencore-amr

opencore-amr is optional. Adds support for AMR audio codec.

```bash
git clone git://opencore-amr.git.sourceforge.net/gitroot/opencore-amr/opencore-amr
autoreconf --install && ./configure && make && make install
```

8. Build libopus

libopus is optional but highly recommended as it’s an MTI codec for WebRTC. Adds support for Opus audio codec.

```bash
wget http://downloads.xiph.org/releases/opus/opus-1.0.2.tar.gz
tar -xvzf opus-1.0.2.tar.gz
cd opus-1.0.2
```

---

**webrtc2sip** – Smart SIP and Media Gateway for WebRTC endpoints
9. Building **libgsm**

libgsm is optional. Adds support for GSM audio codec.

You can install the devel packages (**recommended**):

```
sudo yum install gsm-devel
```

Or build the source by yourself:

```
wget http://www.quut.com/gsm/gsm-1.0.13.tar.gz
tar -xvzf gsm-1.0.13.tar.gz
cd gsm-1.0-pl13 && make && make install
#cp -rf ./inc/* /usr/local/include
#cp -rf ./lib/* /usr/local/lib
```

10. Building **g729**

G729 is optional. Adds support for G.729 audio codec.

```
svn co http://g729.googlecode.com/svn/trunk/ g729b
cd g729b
./autogen.sh && ./configure --enable-static --disable-shared && make && make install
```

11. Building **iLBC**

iLBC is optional. Adds support for iLBC audio codec.

```
svn co http://doubango.googlecode.com/svn/branches/2.0/doubango/thirdparties/scripts/ilbc
cd ilbc
wget http://www.ietf.org/rfc/rfc3951.txt
awk -f extract.awk rfc3951.txt
./autogen.sh && ./configure
make && make install
```

12. Building **x264**

Date: December 2, 2012

x264 is optional and adds support for H.264 video codec (requires FFmpeg).

```
tar -xvf last_x264.tar.bz2
# the output directory may be difference depending on the version and date
cd x264-snapshot-20121201-2245
./configure --enable-shared --enable-pic && make && make install
```

13. Building **FFmpeg**

**webrtc2sip** – Smart SIP and Media Gateway for WebRTC endpoints
FFmpeg is optional and adds support for H.263, H.264 (requires x264) and MP4V-ES video codecs.

```bash
git clone git://source.ffmpeg.org/ffmpeg.git ffmpeg

cd ffmpeg

# grap a release branch
git checkout n1.2

# configure source code
./configure \
  --extra-cflags="-fPIC" \
  --extra-ldflags="-lpthread" \
  --enable-pic --enable-memalign-hack --enable-pthreads \
  --enable-shared --disable-static \
  --disable-network --enable-pthreads \
  --disable-ffmpeg --disable-ffplay --disable-ffserver --disable-ffprobe \
  --enable-gpl \
  --disable-debug

make && make install
```

14. Building Doubango

Minimal build

```bash
cd doubango && ./autogen.sh && ./configure --with-ssl --with-srtp --with-speexdsp
make && make install
```

Recommended build

```bash
cd doubango && ./autogen.sh && ./configure --with-ssl --with-srtp --with-speexdsp
  --with-ffmpeg
make && make install
```

Full build

```bash
cd doubango && ./autogen.sh && ./configure --with-ssl --with-srtp --with-vpx --with-yuv
  --with-amr --with-speex --with-speexdsp --with-gsm --with-ilbc --with-g729 --with-ffmpeg
make && make install
```

5.2 Building webrtc2sip

webrtc2sip depends on Doubango IMS Framework v2.0 and libxml2. The first step is to checkout the source code:

```bash
svn co http://webrtc2sip.googlecode.com/svn/trunk/ webrtc2sip
```
3. Installing libxml2

```
yum install libxml2-devel
```

4. Building webrtc2sip

```
export PREFIX=/opt/webrtc2sip
cd webrtc2sip && ./autogen.sh && ./configure --prefix=$PREFIX
make clean && make && make install
cp -f ./config.xml $PREFIX/sbin/config.xml
```

5.3 Running webrtc2sip

Running webrtc2sip is as easy as executing “webrtc2sip” binary file. Please note that it requires a valid configuration file. The default configuration file should be named “config.xml” and placed in the same folder as “webrtc2sip”.

5.3.1 Command line arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Available since</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--config=PATH</td>
<td>2.1.0</td>
<td>Overrides the default path to the “config.xml”</td>
<td>--config=/tmp/config.xml</td>
</tr>
<tr>
<td>--help</td>
<td>2.1.0</td>
<td>Displays the help message</td>
<td></td>
</tr>
<tr>
<td>--version</td>
<td>2.1.0</td>
<td>Displays the gateway version</td>
<td></td>
</tr>
</tbody>
</table>

For more information on supported command line arguments, please execute `webrtc2sip --help`.

6 Testing the gateway

Let’s say the webrtc2sip gateway and SIP server are running on two different PCs with IP addresses equal to `192.168.0.1` and `192.168.0.2` respectively.

2. Fill “WebSocket Server URL” field with the IP address and port where your webrtc2sip gateway is listening for incoming Websocket connections (e.g `ws://192.168.0.1:10060` or `wss://192.168.0.1:10062`). **IMPORTANT:** Do not forget the url scheme (`ws://` or `wss://`).
3. The “SIP outbound Proxy URL” is used to set the destination IP address and Port to use for all outgoing requests regardless the domain name (a.k.a realm). This is a good option for developers using a SIP domain name without valid DNS A/NAPTR/SRV records. E.g. `udp://192.168.0.2:5060`.
4. Check “Enable RTCWeb Breaker” if you want to call a SIP-legacy endpoint.

7 Interoperability

This section contains good tips to help you to debug some issues you can find when you’re trying to make/receive calls to/from well-known SIP clients or servers using a web browser. Please note that if your preferred web browser is Google Chrome then, we highly recommend using the STABLE version.

7.1 Servers

This section explains know issues and how to tackle them.

7.1.1 Asterisk

*Date: November 29, 2012*
There are some issues (on both Asterisk and Chrome) to get both way audio and video when using Google Chrome stable. There are two solutions.

1. **Patching Asterisk:** This is only recommended if you’re a developer and trying to learn new cool features. Please note that this will not allow video to flow as Asterisk doesn’t support VP8. For more information on how to patch Asterisk, visit [http://code.google.com/p/sipml5/wiki/Asterisk](http://code.google.com/p/sipml5/wiki/Asterisk)

2. **Enabling the RTCWeb Breaker:** This is the recommended solution and it allows both audio and video to flow. Video stream will flow even if the web browser and the SIP client/server do not share the same codecs (thanks to the Media Coder module).

### 7.1.2 FreeSWITCH

The problem here is that FreeSWITCH do not support ICE and some other mandatory RTCWeb features. Enabling the RTCWeb Breaker module (web browser side) is enough to fix the issue.

### 7.2 Web Browsers

#### 7.2.1 Google Chrome

*Date: November 29, 2012*

We highly recommend using the STABLE version for your tests. Please note that we don’t provide any kind of help or support if you’re using the DEV or CANARY versions.

5. Chrome uses SAVPF profile. The S is for secure (SRTP) and the F for feedbacks ([RFC 4585](https://tools.ietf.org/html/rfc4585)). If one of these features is not supported by the remote SIP client/server then you have to enable the RTCWeb Breaker module (web browser side).

6. Chrome only includes VP8 video codec which is not supported by most of SIP clients/servers (e.g. xlite, Asterisk…). If your SIP client/server supports H.264, H.263, Theora or MP4V-ES then, you have to enable both the RTCWeb Breaker (web browser side) and Media Coder (server side) modules to have video. Please note that the Media Coder module will most likely not be enabled on the sipml5.org hosted servers.

#### 7.2.2 Firefox Nightly

*Date January 14, 2012*

Right now only Nightly version of Firefox natively supports RTCWeb. The latest version known to work is **21.0a1 (2013-01-12)**. Please also note that there is a known issue on DTLS-SRTCP decoding (check [issue 194](https://code.google.com/p/webrtc/issues/detail?id=194) for more information).

The RTCWeb implementation in Firefox Nightly uses **DTLS-SRTP** while Chrome uses **SDES-SRTP** which means you need to enable the RTCWeb Breaker module to make calls from one browser to another.

#### 7.2.3 Firefox, Safari, IE and Opera

*Date: November 29, 2012*

---This section intentionally left blank---
7.2.4 Ericsson Bowser  
*Date: November 29, 2012*

Ericsson Bowser does not support Secure RTP (SRTP) and only include *H.264* video codec. Bowser can talk to most of SIP clients but is not compatible with Canary or any RTCWeb client. Enabling the *RTCWeb Breaker* (browser side) will allow Bowser to talk to Chrome for audio only as G.711 is a common codec but video requires the *Media Coder* to be enabled (server side).

7.3 JavaScript SIP stacks  
*Date: November 29, 2012*

--This section intentionally left blank--
8 Security issues

When the RTCWeb Breaker module is enabled on the client side (web browser) then, the server will act as a b2bua for all incoming and outgoing INVITEs to this web browser. Please note that this only apply to the SIP account tied to this particular web browser. Acting as a b2bua means the server will generate a completely new request for each INVITE. The new INVITE request from the b2bua could be challenged (SIP 401/407 response) by the remote SIP-legacy network which means the b2bua must have the SIP account credentials. Instead of sending the username and password to the b2bua we transmit an authentication token (HA1). Off course there is no possibility to retrieve the password from the token but it’s highly recommended not to allow any intermediate node to intercept it and this is why sipML5 automatically use secure websocket (WSS) when RTCWeb Breaker is enabled.

\[
HA1 = MD5(username:realm:password)
\]

```
INVITE sip:1061@sip2sip.info SIP/2.0
Via: SIP/2.0/WSS df7jal23ls0d.invalid;branch=z9hG4bK1tvqE4UJ9VNwxbRNKODUvXQeoDUPL
w2W;rport
From: <sip:13131313@sip2sip.info>;tag=JA2uxtI28xUAM4ZyForT
To: <sip:1061@sip2sip.info>
Contact: "13131313"<sip:13131313@df7jal23ls0d.invalid;rtcweb-breaker=yes;transport=wss>;impi=13131313;ha1=050a0170e77b5d345388598f70d2d1bf/+sip.ice
Call-ID: e7c9abfc-67ce-3192-75e6-4429cbdf2626
CSeq: 9517 INVITE
```

The above INVITE request is received from the web browser when RTCWeb Breaker module is enabled. The b2bua will not include the HA1 parameter when making a new INVITE to the SIP-legacy network even if a secure transport (e.g. DTLS or TLS) is used to forward it.