2016

Mizu WebPhone

Cross-Platform SIP for browsers

Mizu-WebSIPPhone is a standard based VoIP client running from browsers as a ready to use softphone or usable as a JavaScript library. Based on the industry standard SIP protocol, it is compatible with all common VoIP devices.

One software for all OS and all browsers
The ultimate browser to SIP solution

-
Mizutech
7/20/2016

Contents

About

The Mizu WebPhone is a universal SIP client to provide VoIP capability for **all browsers** using a variety of technologies compatible with most OS/browsers. Since it is based on the open standard SIP and RTP protocols, it can inter-operate with any other SIP-based network, allowing people to make true VoIP calls directly from their browsers. Compatible with all SIP softphones (X-Lite, Bria, Jitsi others), devices (gateways, ATA's, IP Phones, others), proxies (SER, OpenSIPS, others), PBX (Asterisk, Trixbox, Avaya, 3CX, Broadsoft, Alcatel, NEC, others), VoIP servers (Mizu, Voipswitch, Cisco, Huawei, others), service providers (Vonage, others) and any SIP capable endpoint (UAC, UAS, proxy, others).

The Mizu WebPhone is truly **cross-platform**, running from both desktop and mobile browsers, offering the best browser to SIP phone functionality in all circumstances, using a variety of built-in technologies referred as "engines":

- NS (Native Service/Plugin)
- WebRTC
- Flash
- Java applet
- App
- P2P/Callback
- Native Dial

The engine to use is automatically selected by default based on OS, browser and server availability (It can be also set manually from the configuration or priorities can be changed).
The webphone can be used with the provided user interface (as a ready to use **softphone** or **click to call** button) or as a **JavaScript library** via its API.
The provided user interfaces are implemented as simple HTML/CSS and can be fully customized, modified, extended or removed/replaced.

Usage

The webphone is an all-in-one VoIP client module which can be used as-is (as a ready to use softphone or click to call) or as a JavaScript library (to implement any custom VoIP client or add VoIP call capabilities to existing applications).

Steps

1. Download

The package can be downloaded from here: webphone download.
It includes everything you need for a browser to SIP solution: the engines, the JavaScript API, the skins and also a few usage examples.

2. Deploy

Unzip and copy the webphone folder into your webserver and refer it from your html (for example from your main page) or open one of the html in your browsers by specifying its exact URL. For example: http://yourdomain.com/webphone/techdemo_example.html
Note1: You must set the webphonebasedir parameter if the html from where you refer to the webphone is not inside the webphone directory!
Note2: You might have to enable the .jar, .exe, .swf, .dll, .so and .dylib mime types in your webserver if not enabled by default (these files might be used in some circumstances depending on the client OS/browser).
Note3: If you wish to use (also) the WebRTC engine then your site should be secured (HTTPS with a valid SSL certificate). Latest Chrome and Opera requires secure connection for both your website (HTTPS) and websocket (WSS). If your website doesn't have an SSL certificate then we can host the webphone for you for free.
Alternatives:

- You can also test it without a webserver by launching the html files from your desktop, although some engines might not work correctly this way
- You can also test it by using the online demo hosted by Mizutech website, but in this case you will not be able to change the configuration (you can still set any parameters from the user interface or from URL)

3. Settings

You will need to set at least the "serveraddress" parameter to your SIP server IP address or domain name in the webphone_api.js file or by URL parameter.
Optionally the settings can be provided by URL parameters or via the API. See the "Parameters" chapter for the details.

4. Optional: JavaScript API

If you are a developer, then you might use the JavaScript API to implement any custom functionality, integrate with your existing software or create your own VoIP client. See the "Java Script API" section in this documentation for the details.

5. Optional: design

If you are a designer then you might create your own design or modify the existing HTML/CSS. For simple design changes you don't need to be a designer. Colors, branding, logo and others can be specified by the settings.

6. Launch

Launch one of the examples (the html files in the webphone folder) or your own html (from desktop by double clicking on it or from browser by entering the URL). You might launch the "index.html" to see the options.
At first start the webphone might offer to enable or download a native plugin if no other suitable engine are supported and enabled by default in your browser.
It will also ask for a SIP username/password if you use the default GUI and these are not preset.
On init, the webphone will register (connect) to your VoIP server (this can be disabled if not needed).

Then you should be able to make calls to other UA (any webphone or SIP endpoint such as X-Lite or other softphone) or to pstn numbers (mobile/landline) if outbound call service is enabled on your server or VoIP provider.

The webphone can be used as a turn-key ready to use solution or as a Java-Script library to develop custom software.

Examples and ready to use solutions:

- **index**.html: just an index page with direct links to the below examples for your convenience
- **basic_example**.html: a basic usage example. You might change/extend it to fit your needs
- **techdemo_example**.html: a simple tech demo. You might make any tests by using this html or change/extend it to fit your needs
- **softphone**.html: a full featured, ready to use browser softphone. You can use it as is on your website as a web dialer. For example you can include it in an iframe on your website. Change the parameters in the webphone_api.js). You can further customize it by changing the parameters or changing its design.
- **softphone_launch**.html: a simple launcher for the above (since the softphone.html is used usually in an iFrame)
- **click2call_example**.html: a ready to use browser to SIP click to call solution. You might further customize after your needs
- **linkify_example**.html: can be used to convert all phone number strings on a website to click to call
- **custom**: you can easily create any custom browser VoIP solution by using the webphone java script library

More details about customization can be found here.
Another quick start guide can be found here.

Web Softphone

The webphone package contains a ready to use web softphone.

Just copy the webphone folder to your webserver and change the “serveraddress” setting in the in webphone_api.js file to your SIP server IP or domain to have a fully featured softphone presented on your website. You can just simply include (refer to) the softphone.html via an iframe (this way you can even preset the webphone parameters in the iframe URL).
Note: you might have to enable the following mime types in your web server if not enabled by default: .jar, .swf, .dll, .dylib, .so, .exe

The web softphone can be configured via URL parameters or in the "webphone_api.js" file, which can be found in the root directory of the package. The most important configuration is the “serveraddress” parameter which should be set to your SIP server IP address or domain name. More details about the parameters can be found below in this documentation in the “Parameters” section.

We can also send you a build with all your branding and settings pre-set: contact us.
See the “User interface Skin/Design” chapter for more details.

Click-to-call

The webphone package contains a ready to use click to call solution.
Just copy the whole webphone folder to your website, set the parameters in the webphone_api.js file and use it from the click2call_example.html.
Rewrite or modify after your needs with your custom button image or you can just use it via a simple URI or link such as:
http://www.yourwebsite.com/webphonedir/clicktocall.html?wp_serveraddress=YOURSIPDOMAIN&wp_username=USERNAME&wp_password=PASSWORD&wp_callto=CALLEDNUMBER
You can find more details in the click to call section.

Developers

Use it as a JavaScript library by including webphone_api.js into your project.

The public JavaScript API can be found in "webphone_api.js" file, under global javascript namespace "webphone_api".
The library parameters can be preconfigured in webphone_api.js, changed runtime from JavaScript or dynamically set by any server side script such as PHP, .NET, java servlet, J2EE or Node.js.

Simple example:

```
<head>
        //Include the webphone_api.js to your webpage
        <script src="webphone_api.js"></script>
</head>
<body>
<script>
        //Wait until the webphone is loaded, before calling any API functions
        webphone_api.onLoaded(function () {

                //Set parameters (Replace upper case worlds with your settings. Alternatively these can be also preset in your html or passed as URL parameters
                webphone_api.setparameter('serveraddress', SERVERADDRESS);
                webphone_api.setparameter('username', USERNAME);
                webphone_api.setparameter('password', PASSWORD);
```

```
            //See the "Parameters" section below for more options

            //Start the webphone (optional but recommended)
            webphone_api.start();

            //Make a call (Usually initiated by user action, such as click on a click to call button. Number can be extension, SIP username, SIP URI or mobile/landline phone)
            webphone_api.call(NUMBER);

            //Hang-up (usually called from "disconnect" button click)
            webphone_api.hangup();

            //Send instant message (Number can be extension, SIP username. Usually called from a "send chat" button)
            webphone_api.sendchat(NUMBER, MESSAGETEXT);
    });
    //You should also handle events from the webphone and change your GUI accordingly (onXXX callbacks)
</script>
</body>
```

For more details, see the "JavaScript API" section below in this documentation.
As a better example, see the tech demo page (techdemo_example.html / techdemo_example.js).

If you are more comfortable with server side web development (ASP, .NET, java or others) then you can still control the webphone in many ways by:
-dynamically write out the webphone settings or generate its URL with the desired parameters depending on the user/session
-setup a HTTP API to catch events such as login/call start/disconnect and drive your app logic accordingly (see the API/callbacks for these)

Designers

If you are a designer, you can modify all the included HTML/CSS/images or create your own design from scratch using any technology that can bind to JS such as HML5/CSS, Flash or others.

For simple design changes you don't need to be a designer. Colors, branding, logo and others can be set by the settings.
See the "User Interface Skin/Design" section for more details.

Features

- Standard SIP client for voice calls (in/out), chat, conference and others (Session Initiation Protocol)
- Maximum browsers compatibility. Runs in all browsers with Java, WebRTC or native plugin support (Chrome, Firefox, IE, Edge, Safari, Opera)
- Includes several different technologies to make phone calls (engines): Java applet, WebRTC, NS (Native Service or Plugin), Flash, App, Native and server assisted conference rooms, P2P and callback
- SIP and RTP stack compatible with any standard VoIP servers and devices like Cisco, Voipswitch, Asterisk, softphones, ATA and others
- Transport protocols: UDP, TCP, HTTP, RTMP, websocket (uses UDP for media whenever possible)
- Encryption: SIPS, TLS, DTLS, SRTP, end to end encryption for webphone to webphone calls
- NAT/Firewall support: auto detect transport method (UDP/TCP/HTTP), stable SIP and RTP ports, keep-alive, rport support, proxy traversal, auto tunneling when necessary, ICE/STUN/TURN protocols and auto configuration, firewall traversal for corporate networks, VoIP over HTTP/TCP when firewalls blocks the UDP ports
- RFC's: 2543, 3261, 2976, 3892, 2778, 2779, 3428, 3265, 3515, 3311, 3911, 3581, 3842, 1889, 2327, 3550, 3960, 4028, 3824, 3966, 2663 and others
- Supported methods: REGISTER, INVITE, re-INVITE, ACK,PRACK, BYE, CANCEL, UPDATE, MESSAGE, INFO, OPTIONS, SUBSCRIBE, NOTIFY, REFER
- Audio Codec: PCMU, PCMA, G.729, GSM, iLBC, SPEEX, OPUS (including wide-band HD audio)
- Video codec: H.263, H.264 and VP8 for WebRTC only
- Call divert: rewrite, redial, mute, hold, transfer, forward, conference
- Voice call recording
- IM/Chat (RFC 3428), SMS, file transfer, DTMF, voicemail (MWI)
- Contact management: flags, synchronization, favorites, block, presence (DND/online/offline/others)
- Balance display, call timer, inbound/outbound calls, caller-id display
- High level JavaScript API
- Integration with any website or application including simple static pages, apps with client side code only (like a simple static page) or any server side stack such as PHP, .NET, java servlet, J2EE, Node.js and others (sign-up, CRM, callcenter, payments and others)
- Branding and customization: customizable user interface with your own brand, skins and languages (with ready to use, modifiable themes)
- Flexibility: all parameters/behavior can be changed/controlled by URL parameters, preconfigured parameters, from javascript or from server side

Requirements

Server side:

- Any **web hosting** for the webphone files (any webserver is fine: IIS, nginx, Apache, NodeJS, Java, others; any OS: Windows, Linux, others).

Chrome and Opera requires secure connection for WebRTC engine to work (otherwise this engine will be automatically skipped). We can also host the webphone for free if you wish on secure http. Note that the webphone itself doesn't require any framework, just host it as static files (no PHP, .NET, JEE, NodeJS or similar server side scripting is required to be supported by your webserver)

- At least one **SIP account** at any VoIP service provider or your own SIP server (such as Asterisk, FreePBX, Trixbox, Voipswitch, 3CX, SER, Cisco and others)
- Optional: WebRTC capable SIP server or SIP to WebRTC gateway (Mizutech free WebRTC to SIP service is enabled by default. The webphone can be used and works fine also without WebRTC, however if you prefer this technology then free software is available and Mizutech also offers WebRTC to SIP gateway (free with the Advanced license) and free service tier. Common VoIP servers also has built-in WebRTC support nowadays)

Client side:

- Any **browser** supporting WebRTC OR Java OR native plugins with JavaScript enabled (most browsers are supported)
- **Audio device**: headset or microphone/speakers

Compatibility:

- OS: Windows (XP,Vista,7,8,10) , Linux, MAC OSX, iOS (app only), Android, BlackBerry, Chromium OS, Firefox OS and others
- Browsers: Firefox, Chrome, IE (6+), Edge, Opera and others
- Different OS/browser might have different compatibility level depending on the usable engines. For example the rarely used Flash engine doesn't implement all the functionalities of the WebRTC/Java/NS engines (these differences are handled automatically by the webphone API)

If you don't have a SIP server or VoIP account yet, you can use or test with our VoIP service.

- Server address: voip.mizu-voip.com
- Account: create free VoIP account from here or use the following username/passwords: webphonetest1/webphonetest1, webphonetest2/webphonetest2 (other people might also use these public accounts so calls might be misrouted)

Technical details

The goal of this project is to implement a VoIP client compatible with all SIP servers, running in all browsers under all OS with the same user interface and API. At this moment no technology exists to implement a VoIP engine to fulfill these requirements due to browser/OS fragmentation. Also different technologies have some benefits over others. We have achieved this goal by implementing different "VoIP engines" targeting each OS/browser segment. This also has the advantage of just barely run a VoIP call, but to offer the best possible quality for all environments (client OS/browser). All these engines are covered by a single, easy to use unified API accessible from JavaScript. To ease the usage, we also created a few different user interfaces in HTML/JS/CSS addressing the most common needs such as a VoIP dialer and a click to call user interface.

Each engine has its advantages and disadvantages. The webphone will automatically choose the "best" engine based on your preferences, OS/Browser/server side support and the enduser preferences (this can be overridden by settings if you have some special requirements): VoIP availability in browsers.

Engines

NS

Our native VoIP engine implemented as a native service or browser plugin. The engine works like a usual SIP client, connecting directly from client PC to your SIP server, but it is fully controlled from web (the client browser will communicate in the background with the native engine installed on the client pc/mobile device, thus using this natively installed sip/media stack for VoIP).

Pros:

- All features all supported, native performance

Cons

- Requires install (one click installer)

WebRTC

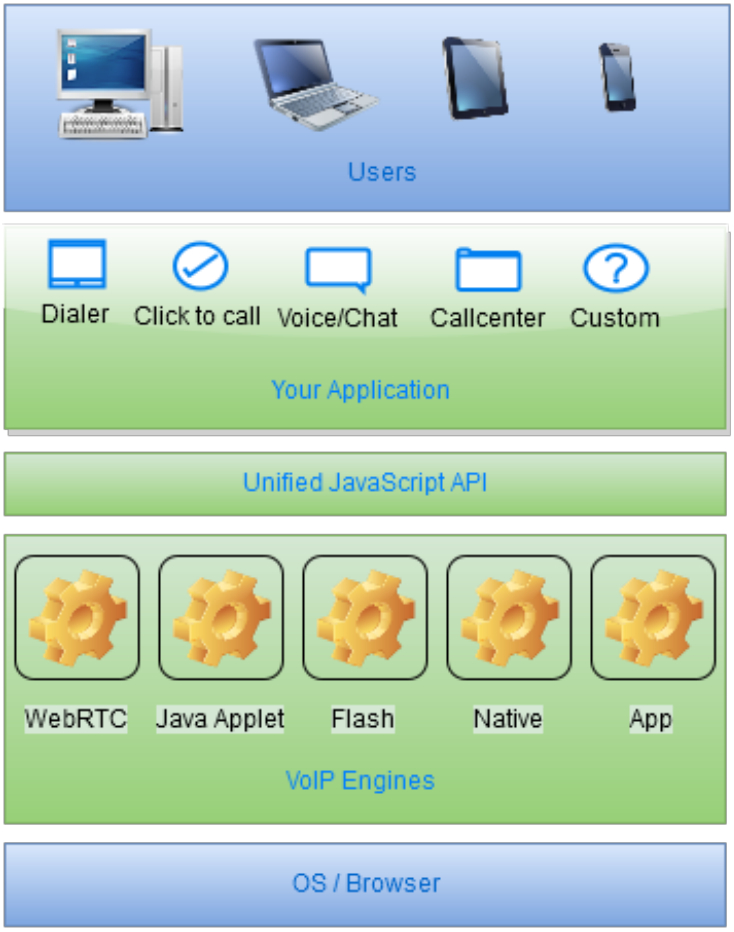
A new technology for media streaming in modern browsers supporting common VoIP features. WebRTC is a built-in module in modern browsers which can be used to implement VoIP. Signaling goes via websocket (unencrypted or TLS) and media via encrypted UDP (DTLS-SRTP). These are then converted to normal SIP/RTP by the VoIP server or by a gateway.

Pros:

- Comfortable usage in browsers with WebRTC support because it is has no dependencies on plugins

Cons:

- It is a black-box in the browser with a restrictive API
- Lack of popular VoIP codec such as G.729 (this can be solved by CPU intensive server side transcoding)
- A WebRTC to SIP gateway may be required if your VoIP server don't have built-in support for WebRTC (there are free software for this and we also provide a free service tier, included by default)



Flash

A browser plugin technology developed by Adobe with its proprietary streaming protocol called RTMP.
Pros:

- In some old/special browsers only Flash is available as a single option to implement VoIP

Cons:

- Requires server side Flash to SIP gateway to convert between flash RTMP and SIP/RTP (we provide free service tier)
- Basic feature set

Java Applet

Based on our powerful JVoIP SDK, compatible with all JRE enabled browsers. Using the Java Applet technology you can make SIP calls from browsers the very same way as a native dialer, connecting directly from client browser to SIP server without any intermediary service (SIP over UDP/TCP and RTP over UDP).
Pros:

- All SIP/media features are supported, all codecs including G.729, wideband and custom extra modules such as call recording
- Works exactly as a native softphone or ip phone connecting directly from the user browser to your SIP capable VoIP server or PBX (but with your user interface)

Cons:

- Java is not supported by some browser, most notable mobile devices and Chrome which has dropped NPAPI support required for the Java plugin (in this case the webphone will use WebRTC, Flash or Native engine instead of Java)
- Some browsers may ask for user permission to activate the Java plugin

App

Some platforms don't have any suitable technology to enable VoIP in browsers (a minor percentage, most notably iOS/Safari). In these cases the webphone can offer to the user to install a native softphone application. The apps are capable to fully auto-provision itself based on the settings you provide in your web application so once the user installs the app from the app store, then on first launch it will magically auto-provision itself with most of your settings/branding/customization/user interface as you defined for the webphone itself.
Pros:

- Covering platforms with lack of VoIP support in browsers (most notable: iOS Safari)

Cons:

- No API support. Not the exactly same HTML user interface (although highly customized based on the settings you provided for the webphone)

P2P and callback

These are just "virtual" engines with no real client VoIP stack.

- P2P means server initiated phone to phone call initiated by an API call into your VoIP server. Then the server will first call you (on your regular mobile/landline phone) and once you pick it up it will dial the other number and you can start talking (Just set the "p2p" setting to point to your VoIP server API for this to work)
- Callback is a method to make cheap international calls triggering a callback from the VoIP server by dialing its callback access number. It might be used only as a secondary engine if you set a callback access number (Just set the "callback" setting to point to your VoIP server API for this to work)

These are treated as a secondary (failback) engines and used only if no other engines are available just to be able to cover all uncommon/ancient devices with lack of support for all the above engines which is very rare. However it might be possible that these fits into your business offer and in that case you might increase their priority to be used more frequently.

Native Dial

This means native calls from mobile using your mobile carrier network. This is a secondary "engine" to failback to if no any VoIP capabilities were found on the target platform or there is no network connection. In these circumstances the webphone is capable to simply trigger a phone call from the user smartphone if this option is enabled in the settings. Rarely used if any.

The most important engines are: Java, WebRTC, NS and Flash. The other engines are to provide support for exotic and old browsers maximizing the coverage for all OS/browser combinations ensuring that enduser has call capabilities regardless of the circumstances.

API

All the above engines are covered with an easy to use unified Java Script API, hiding all the differences between the engines as described below in the "JavaScript API" section.

GUI

The webphone can be used with or without a user interface.
The user interface can be built using any technology with JS binding. The most convenient is HTML/CSS, but you can also use any others such as Flash.
The webphone package comes with a few prebuilt feature rich responsive user interfaces covering the most common usage such as a full featured softphone user interface and a click to call implementation. You are free to use these as is, modify them after your needs or create your own from scratch. For more details check the "User interface Skin/Design" section.

Licensing

The webphone is sold with unlimited client license (Advanced and Gold) or restricted number of licenses (Basic and Standard). You can use it with any VoIP server(s) on your own and you can deploy it on any webpage(s) which belongs to you or your company. Your VoIP server(s) address (IP or domain name) and optionally your website(s) address will be hardcoded into the software to protect the licensing. You can find the licensing possibilities on the pricing page. After successful tests please ask for your final version at webphone@mizu-voip.com. Mizutech will deliver your webphone build within one workday after your payment.

Release versions don't have any limitations (mentioned below in the "Demo version" section) and are customized for your domain. All "mizu" and "mizutech" words and links are removed so you can brand it after your needs (with your company name, brand-name or domain name), customize and skin (we also provide a few skin which can be freely used or modified).

Your final build must be used only for you company needs (including your direct sip endusers or service customers).

Title, ownership rights, and intellectual property rights in the Software shall remain with MizuTech.
The agreement and the license granted hereunder will terminate automatically if you fail to comply with the limitations described herein. Upon termination, you must destroy all copies of the Software. The software is provided "as is" without any warranty of any kind. You must accept the software SLA before to use the webphone.

You may:

- Use the webphone on any number of computers, depending on your license
- Give the access to the webphone for your customers or use within your company
- Offer your VoIP services via the webphone
- Integrate the webphone to your website or web application
- Use the webphone on multiple webpage's and with multiple VoIP servers (after the agreement with Mizutech). All the VoIP servers must be owned by you or your company. Otherwise please contact our support to check the possibilities

You may not:

- Resell the webphone
- Sell "webphone" services for third party VoIP providers and other companies
- Resell the webphone or any derivative work which might compete with the Mizutech "webphone" software offer
- Reverse engineer, decompile or disassemble or modify the software in any way except modifying the settings and the HTML/CSS skins or the included JavaScript examples

Demo version

The downloadable demo version can be used to try and test before any purchase. The demo version has all features enabled but with some restrictions to prevent commercial usage. The limitations are the followings:

- maximum 10 simultaneous webphone at the same time
- will expire after several months of usage (usually 2 or 3 months)
- maximum ~100 sec call duration restriction
- maximum 10 calls / session limitation. (After ~10 calls you will have to restart your browser)
- will work maximum ~20 minutes after that you have to restart it or restart the browser
- can be blocked from Mizutech license service

In short: the demo version can be used for all kind of tests or development, but it can't be used for production.
Note: for the first few calls and in some circumstances the limitations might be weaker than described above, with fewer restrictions.
On request we can also provide test builds with only trial period limitation (will expire after ~3 weeks of usage) and without the above demo limitations.

See the pricing and order your licensed copy from here.

Parameters

The parameters can be used to customize the user interface or control the settings like the SIP server domain, authentication, called party number, autodial and many others.

Most of the settings are optional except the "**serveraddress**" (but also this can be provided at runtime via the API).

The other important parameters are the SIP user credentials (**username**, **password**) and the called number (**callto**) which you can also preset (for example if you wish to implement click to call) however these are usually entered by user (and optionally can be saved in local cookie for later reuse).

The webphone parameters can be set in multiple ways:

- Preset in the "**webphone_api.js**" file, under "parameters" variable (in "parameters" Javascript object)
- Using the **setparameter()** API call from JavaScript (Other function calls might also change settings parameters)
- Webpage **URL** (The webphone will look at the embedding document URL at startup. Prefix all keys with "wp_". For example &wp_username=x)
- Cookies (prefix all keys with "wp_". For example wp_username)
- SIP signaling (sent from server) with the x-mparam header (or x-mparamp if need to persist). Example: x-mparam=loglevel=5;aec=0
- Auto-provisioning: the webphone is also capable to download it's settings from a config file based on user entered OP CODE (although this way of configuration is a little bit redundant for a web app, since you can easily create different versions of the app –for example by deploying it in different folders- already preconfigured for your customers, providing a direct link to the desired version instead of asking the users to enter an additional OPCODE)
- User input: You can let the user to modify the settings. For example to enter username/password for SIP authentication

Any of these methods can be used or they can be even mixed.
All parameters can be passed as strings and will be converted to the proper type internally by the webphone.

Note: once a parameter is set, it might be cached by the webphone and used even if you remove it later. To prevent this, set the parameter to "DEF" or "NULL". So instead of just deleting or setting an empty value, set its value to "DEF" or "NULL". "DEF" means that it will use the parameter default value if any. "NULL" means empty for strings, otherwise the parameter default value.

Parameters can be also encrypted or obfuscated. See the "Parameter security" section for the details.

For a basic usage you will have to set only your VoIP server ip or domain name ("serveraddress" parameter).
The rest of the parameters are optional and should be changed only if you have a good reason for it.

SIP account settings

Credentials and other SIP parameters:

serveraddress

(string)
The domain name or IP address of your SIP server. By default it uses the standard SIP port (5060). If you need to connect to other port, you can append the port after the address separated by colon.
Examples:

"mydomain.com" (this will use the default SIP port: 5060)
"sip.mydomain.com:5062"
"10.20.30.40:5065"

This is the single most important parameter (along with the username/password but those can be also entered by the user).
Default value is empty.

username

(string)
This is the SIP username (used for authentication and as A number/Caller-ID for the outgoing calls).
Note: if you wish to set a separate caller-id you can use this parameter to specify it and then use the "sipusername" parameter to specify the username used for authentication as specified in SIP standards. However please note that most SIP server can treat also the below mentioned "displayname" parameter as the caller-id so the usage of separate username/sipusername is usually not necessary and confusing.
Default value is empty.

password

(string)
SIP authentication password.
Default value is empty.

displayname

(string)
Specify default display name used in “from” or “contact” SIP headers.
Default value is empty (the “username” field will be displayed for the peers).

realm

(string)
Optional parameter to set the SIP realm if not the same with the serveraddress or domain.
Rarely required.
Default value is empty. (By default the serveraddress will be used without the port number)

proxyaddress

(string)
Outbound proxy address (Examples: mydomain.com, mydomain.com:5065, 10.20.30.40:5065)
Leave it empty if you don’t have a stateless proxy. (Use only the serveraddress parameter)
Default value is empty.

register

(number)
With this parameter you can set whether the softphone should register (connect) to the sip server.
0: no
1: auto guess (yes if username/password are preset, otherwise no)
2: yes
Default value is 1.

voicemailnum

(string)
Specify the voicemail number (which the user can call to hear its own voicemails) if any.
Most servers will automatically send the voicemail access number so usually this is detected automatically.
Default value is empty (auto-detect).

callto

(string)
The webphone will initiate call on startup if this is set. It can be used to implement click to call or similar functionality.
Can be any phone number, username or SIP URI acceptable by your VoIP server.
Default value is empty.

autoaction

(number)
Useful for click-to-call to specify what to do if you pass the “callto” parameter
0: nothing (do nothing, just preset the destination number; the user will have to initiate the call/chat)
1: call (default. Will auto start the call to “callto”)
2: chat (will show the chat user interface presenting a chat session with “callto”)
3: video call (will auto start a vide call)

registerinterval

(number)
Registration interval in seconds (used by the re-registration expires timer).
Default value is 120.

customsipheader

(string)
Set a custom sip header (a line in the SIP signaling) that will be sent with all messages. Can be used for various integration purposes and usually has a key:val format. For example: myheader:myvalue.

Custom SIP headers should begin with “X-“ to be able to bypass servers, gateways and proxies (For example: X-MyHeader: 47).
You can add more than one header, separated by semicolon (For example: customsipheader: 'key1: val1;key2: val2',).
Default is empty.

prefcodec

(string)
Set your preferred audio codec. Will accept one of the followings: pcmu, pcma, g.719, gsm, ilbc, speex, speexwb, speexuwb, opus, opuswb, opusuwb, opusswb
Default is empty which means the built-in optimal prioritization.
By default the engine will present the codec list optimized regarding the circumstances (the combination of the followings):

- available client codec set (not all engines supports all codecs)
- server codec list (depending on your server, peer device or carrier)
- internal/external call: for IP to IP calls will prioritize wideband codecs if possible, while for outbound calls usually G.729 will be selected if available
- network quality (bandwidth, delay, packet-loss, jitter): for example iLBC is more tolerant to network problems if supported
- device CPU: some old mobile devices might not be able to handle high-complexity codec’s such as opus or G.729. G711 and GSM has low computational costs

You can also fine-tune the codec settings with the use_xxx settings where xxx is the codec name as described in JVoIP documentation.

dtmfmode

(number)
DTMF send method
0: disabled
1: sip INFO method
2: RFC2833 in the RTP (if RTP stream is working, otherwise it will send also SIP INFO)
3: both INFO and RFC2833
4: RFC2833 (will not send SIP INFO even if there is no RTP stream negotiated)
Default is 2.

Note: Received DTMF are recognized by default in both INFO or RFC2833 formats (No In-Band DTMF processing)

Engine related

Library and engine related settings:

webphonebasedir

If the html page, where you are including the webphone, is not in the same directory as the webphone, then you must set the "webphonebasedir" parameter. This must be the relative path to the webphone base directory in relation to your html page.
The base directory is the "webphone" directory as you download it from Mizutech (which contains the css,js,native,... directories).

For example if your page is located at http://yoursite.com/content/page.html and the webphone files are located at http://yoursite.com/modules/webphone then the webphonebasedir have to be set to '../modules/webphone/'
Default is empty (assumes that your html is in the webphone folder)

engine priority

By default the webphone will choose the “best” suitable engines automatically based on OS/browser/server support. This algorithm is optimized for all OS and all browsers so you can be sure that your users will have the best experience with default settings, however, if you wish, you can influence this engine selection algorithm by setting one or more of the following parameters:

- **enginepriority_java**
- **enginepriority_webrtc**
- **enginepriority_ns**
- **enginepriority_flash**
- **enginepriority_app**
- **enginepriority_p2p**
- **enginepriority_accessnum**
- **enginepriority_nativedial**

Possible values:

0: Disabled (never use this engine)
1: Lower (decrease the engine priority)
2: Normal (default)
3: Higher (will boost engine priority)
4: Highest (will use this engine whenever possible)
5: Force (only this engine will be used)

For example if you wish to prioritize the NS engine, just set: enginepriority_ns=3

The engines also have a built-in default priority number assigned which can range from 0 to 100. You can change also these values with the enginedefpriority_ENGINENAME settings. Default values:

enginedefpriority_java: 32
enginedefpriority_webrtc: 20
enginedefpriority_flash: 13
enginedefpriority_ns: 30
enginedefpriority_app: 10
enginedefpriority_p2p: 5
enginedefpriority_callback: 5
enginedefpriority_nativedial: 3

Even if you have a favorite engine, you should not disable the others. Just set your favorite engine priority to 3 or 4. This way even endusers which doesn't have a chance to run your favorite engine might be able to make calls with other engines.

autostart

(boolean)
Specify whether the webphone stack should be started automatically on page load.
If set to false then the start() method needs to be called manually in order for the webphone to start. Also the webphone will be started automatically on some other method calls such as register() or call().
Default is true.

webrtcserveraddress

(string)
Optional setting to indicate the domain name or IP address of your websocket service used for WebRTC if any. If not set, then the mizu webrtc service is used.
Examples:

ws://mydomain.com
wss://sip.mydomain.com:5062
ws://10.20.30.40:5065

Default value is empty (which means auto service discovery).
Note: latest Chrome and Opera require secure websocket (wss). You will need to install an SSL certificate for your WebRTC server for this and set this parameter with the domain name (not IP address). This is needed only if your VoIP server is WebRTC capable or you have your own WebRTC to SIP gateway. Otherwise no changes are required.
More details about webrtc can be found in the FAQ.

rtmpserveraddress

(string)
Optional setting to indicate the address (domain name or IP address + port number) of your flash service if any (flash media + RTMP). If not set, then the mizu flash to sip service might be used (rarely used in normal circumstances). Format: yourdomain.com:rtmpport
Example: 10.20.30.40:5678
Default value is empty.

stunserveraddress

(string)
STUN server address in address:port format (RFC 5389)
You can set to "null" to completely disable STUN.
Examples:

11.22.33.44:3478

mystunserver.com:3478
null

By default (if you leave this setting unchanged) the webphone will use the Mizutech STUN servers (unlimited free service for all webphone customers). You can change this to your own STUN server or use any public server if you wish.

Note: if you set an incorrect STUN server, then the symptoms are extra delays at call setup (up to "icetimeout").

turnserveraddress

(string)
TURN server address in address:port format (RFC 5766)
You can set to "null" to completely disable TURN.
Examples:

11.22.33.44:80
mystunserver.com:80
null

TURN is required only if the webphone cannot send the media directly to the peer (which is usually your VoIP server). For example if all UDP is blocked or only TCP 80 is allowed or you need peer to peer media via TURN relay

By default (if you leave this setting unchanged) the webphone can use the Mizutech TURN servers. If you wish, you can deploy your own TURN server using the popular open source coturn server. The MizuTech WebRTC to SIP gateway also has its own built-in TURN server.

turnparameters

(string)
Any TURN URI parameter.
Example: transport=tcp

turnusername

(string)
Username for turn authentication.

turnpassword

(string)
Password for turn authentication.

icetimeout

(number)
Timeout for ICE address gathering (STUN/TURN/others) in milliseconds.
Default is 3000 (3 seconds).
You might increase in special circumstances if you are using some slow STUN/TURN server or decrease if peer address is public (like if your SIP or WebRTC server is on public IP always routing the media, so calls will work also without STUN).

transport

(number)
Transport protocol for native SIP.
0: UDP (User Datagram Protocol. The most commonly used transport for SIP)
1: TCP (signaling via TCP. RTP will remain on UDP)
2: TLS (encrypted signaling)
3: HTTP tunneling (both signaling and media. Supported only by mizu server or mizu tunnel)
4: HTTP proxy connect (requires tunnel server)
5: Auto (automatic failover from UDP to HTTP if needed)
Default is 0.

Note: this will not affect WebRTC since webrtc transport is controlled by the browser: http/https, websocket/secure websocket (ws/wss) and DTLS/SRTP for the media.

mediaencryption

(number)
Media encryption method
0: not encrypted (default)
1: auto (will encrypt if initiated by other party)
2: SRTP
Default is 0.
Note: this will not affect WebRTC since webrtc always uses DTLS/SRTP for the media.

autodetectwebrtc

(number)
Try to auto-detect webrtc address if not set (if the active SIP server has built-in WebRTC capabilities)
0: no
1: yes
Default is 1.

offersoftphone

(boolean)
Offer native softphone to install if no suitable engine found.
Download links can be configured with "android_nativedialerurl" and "ios_nativedialerurl" listed below, otherwise the default will be used (auto provisioned apps from Mizutech with your branding, customization and settings as you define it for the webphone).
Default is true.

android_nativedialerurl

(string)
Android native softphone download URL if any. (Optional setting to allow alternative softphone offer on Google Play)
Default is empty.

ios_nativedialerurl

(string)
iOS native softphone download URL if any. (Optional setting to allow alternative softphone offer on Apple App Store)
Default is empty.

accessnumber

(string)
Set this if your server has an access number where users can call into from PSTN and it can forward their call on VoIP (IVR asking for the target number).
This can be used when no other engines are working (no suitable environment, no internet connection).
Default is empty.

callbacknumber

(string)
Set this if your server has a callback access number where users can ring into and will receive a call from your server (possibly with an IVR which might offer the possibility to specify the destination number via DTMF).
This can be used when no other engines are working (no suitable environment, no internet connection) and it is very useful in situation where call from the server is cheaper than user call to server.
Default is empty.

codec

(number)
List of allowed codec's separated by comma.
By default the webphone will automatically choose the best codec depending on available codec's, circumstances (network/device) and peer capabilities.
Set this parameter only if you have some special requirements such as forcing a specific codec, regardless of the circumstances.
Example: Opus,G.729,PCMU (This will disable Speex, GSM, iLBC, GSM and PCMA).
Default: empty (which means auto detection and negotiation)
Recommended value: leave it empty

Under normal circumstances, the following is the built-in codec priority:

I. Speex and Opus (These are set with top priority as they have the best quality. Likely used for VoIP to VoIP calls if the peer also has support for wideband)
II. G.729 (Usually the preferred codec for VoIP trunks used for mobile/landline calls because it's excellent compression/quality ratio for narrowband)
III. iLBC, GSM (If G.729 is not supported then these are good alternatives. iLBC has better characteristics but GSM is better supported by legacy hardware)
IV. G.711: PCMU and PCMA (Requires more bandwidth, but has the best narrowband quality. Preferred from WebRTC if Opus is not supported as these are present in almost any WebRTC and SIP endpoints and servers)

video

(number)
Enable/disable video.

-1: auto (default)
0: disable
1: enable
2: force always

video_bandwidth

(number)
Max bandwidth for video in kbits.
It will be sent also with SDP "b:AS" attribute.
Default is 0 which means auto negotiated via RTCP and congestion control.

video size parameters

(number)
You can suggest the size of the video (in pixels) with the following parameters:

- video_width
- video_height
- video_min_width
- video_min_height
- video_max_width
- video_max_height

aec

(number)
Enable/disable acoustic echo cancellation
0=no
1=yes except if headset is guessed
2=yes if supported
3=forced yes even if not supported (might result in unexpected errors)
Default is 1.

agc

(number)
Automatic gain control.
0=Disabled
1=For recording only
2=Both for playback and recording
3=Guess
Default value is 3

loglevel

(number)
Tracing level. Values from 1 to 5.
Log level 5 means a full log including SIP signaling. Higher log levels should be avoided, because they can slow down the softphone.
Loglevel above 5 is meant only for Mizutech developers and might slow down the webphone.
Do not set to 0 because that will disable also the important notifications presented for the users.
More details about logs can be found here.

logtoconsole

(boolean)
Specify whether to send logs to console.
true: will output all logs to console (default)
false: will output only level 1 (important events also displayed for the user)
The amount of logs depends on the "loglevel" parameter.
Default is: true

Call divert and other settings

These parameters are used for call auto-answer, forward, transfer, number rewrite and similar tasks:

normalizenumber

(number)
Normalize called phone numbers.
If the dialed number looks like a phone number (at least 5 number digits and no a-z, A-Z or @ characters and length between 5 and 20) then will drop all special characters leaving only valid digits (numbers, *, # and + at the beginning).
Possible values:
0: no, don't normalize
1: yes, normalize (default)

techprefix

(string)
Add any prefix for the called numbers.
Default is empty.

numpxrewrite

In case if you need to rewrite numbers after your dial plan on the client side, you can use the numpxrewrite parameter (although these kind of number rewrite are usually done after server side dial plan):
You can set multiple rules separated by semicolon.
Each rule has 4 parameters, separated by comma: prefix to rewrite, rewrite to, min length, max length
For example:

'74,004074,8,10;+,001,7,14;',

This will rewrite the 74 prefix in all numbers to 004074 if the number length is between 8 and 10.
Also it will rewrite the + prefix in all numbers to 001 if the number length is between 7 and 14.

enablepresence2

(number)
Enable/disable presence.
Possible values:
0: disable
1: auto (if presence capabilities detected)
2: always enable / force

blacklist

(string)
Block incoming communication (call, chat and others) from these users. (username/numbers/extensions separated by comma).
Default value is empty.

callforwardonbusy

(string)
Specify a number where incoming calls should be forwarded when the user is already in a call. (Otherwise the new call alert will be displayed for the user or a message will be sent on the JS API)
Default is empty.

callforwardonnoanswer

(string)
Forward incoming calls to this number if not accepted or rejected within 15 seconds.
Default is empty.

callforwardalways

(string)
Specify a number where ALL incoming calls should be forwarded.
Default is empty.

calltransferalways

(string)
Specify a number where ALL incoming calls should be transferred to.
This might be used if your server doesn't support call forward (302 answers) otherwise better to set this on server side because the call will not reach the webphone when it is offline/closed, so no chance for it to forward the call.
Default is empty.

autoignore

(number)
Set to ignore all incoming calls.
0: don't ignore
1: silently ignore
2: reject
Default value is 0.

autoaccept

(boolean)
Set to true to automatically accept all incoming calls (auto answer).
Default value is false.

beeponconnect

(number)
Will play a short sound when calls are connected
0: Disabled
1: For auto accepted incoming calls
2: For incoming calls
3: For outgoing calls
4: For all calls
Default value is 0

rejectonbusy

(boolean)
Set to true to automatically reject (disconnect) incoming call if a call is already in progress.
Default value is false.

automute

(number)
Specify if other lines will be muted on new call
0=no (default)
1=on incoming call
2=on outgoing call
3=on incoming and outgoing calls
4=on other line button click
Default is 0

autohold

(number)
Specify if other lines will be muted on new call
0=no (default)
1=on incoming call
2=on outgoing call
3=on incoming and outgoing calls
4=on other line button click
Default is 0

transfertype

(number)
Specify transfer mode for native SIP.
-1=default transfer type (same as 6)
0=call transfer is disabled
1=transfer immediately and disconnect with the A user when the Transf button is pressed and the number entered (unattended transfer)
2=transfer the call only when the second party is disconnected (attended transfer)
3=transfer the call when the VoIP Applet is disconnected from the second party (attended transfer)
4=transfer the call when any party is disconnected except when the original caller was initiated the disconnect (attended transfer)
5=transfer the call when the VoIP Applet is disconnected from the second party. Put the caller on hold during the call transfer (standard attended transfer)
6=transfer the call immediately with hold and watch for notifications (unattended transfer)

Default is -1 (which is the same as 6)
If you have any incompatibility issue, then set to 1 (unattended is the simplest way to transfer a call and all sip server and device should support it correctly)

transfwithreplace

(number)
Specify if replace should be used with transfer so the old call (dialog) is not disconnected but just replaced.
This way the A party is never disconnected, just the called party is changed. The A party must be able to handle the replace header for this.
-1=auto
0=no (will create a separate call)
1=yes (smooth transfer, but not supported by some servers)
Default is -1

ringtimeout

(number)
Maximum ring time allowed in millisecond.
Default is 90000 (90 second)

calltimeout

(number)
Maximum speech time allowed in millisecond.
Default is 10800000 (3 hours)

bargeinheader

(string)
You can barge-in or spy on the calls by sending a specific SIP header specified by the "bargeinheader" parameter available for NS and Java engines.
Default is empty (disabled).

voicerecupload

(string)
Voice upload URL.
If set then calls will be recorded and uploaded to the specified ftp or http address in mp3 or wave format.
The files can be uploaded to your FTP server (any FTP server with specified user login credentials) or HTTP server (in this case you need a server side script to save the uploaded data to file using http PUT or multipart/form-data POST)
Example:
ftp://user01:pass1234@ftp.foo.com/voice_DATE_CALLER_CALLED.wav
http://www.foo.com/myfilehandler.php/?filename=callrecord_CALLID.mp3
where the filename part can be set to any string ending with .mp3, .wav, .gsm, .ogg and you can use keywords which will be replaced accordingly.

The following keywords are defined:

- DATE: will be replaced to current date-time
- CALLID: will be replaced to sip call-id
- USER: will be replaced to local user name
- CALLER: will be replaced to caller party name
- CALLED: will be replaced to callee party name

Default is empty (no voice call recording).

NS and Java extra settings

With the NS and Java engines you can also use any parameters supported by the Mizu JVoIP SDK as listed in the JVoIP documentation.
(Unrecognized parameters will be skipped if the WebRTC engine is used)

User interface related

Most of these apply only to the Softphone user interface which is shipped with the webphone.

brandname

(string)
Brand name of the softphone.
Default is empty.

companyname

(string)
Your company name.
Default is empty.

logo

(string)
Displayed on login page.
Can be text or an image name, ex: "logo.png" (image must be stored in images/folder)
Default is empty.

colortheme

(number)
You can easily change the skin of the supplied user interfaces with this setting (softphone, click to call).
Possible values:

- 0: Default (android grey)
- 1: Light
- 2: Green
- 3: Green Light
- 4: Red
- 5: Red Light
- 6: Yellow
- 7: Yellow Light
- 8: Magenta
- 9: Magenta Light

Default is 0.

featureset

(number)
User interface complexity level.

0=minimal
5=reduced
10=full (default)
15=extreme

You might set to 5 for novice users or if only basic call features have to be used.
Default is 10.

showserverinput

(number)
This can be used to hide the server address setting from the user if you already preconfigured the server address in the webphone_api.js (“serveraddress” config option), so the enduser have to type only their username/password to use the softphone.
Possible values:

0: no (will hide the server input setting for the endusers)
1: auto (default)
2: yes (will shot the server input setting for the endusers)

useloginpage

(number)
Whether to use a simplified login page with username/password in the middle (instead of list style settings; old haveloginpage).
Possible values:

-1: auto (will auto set to 1 if featureset is Minimal, otherwise 0)
0: no
1: only at first login
2: always

chatsms

(number)
0: Auto guess or Ask
1: SMS only
2: Chat only
Default is 0.

conferencerooms

(number)
Enable/disable conference room feature.
0: disabled
1: enabled (if supported by the server)
Default is 1.

hasfiletransfer

(boolean)
Set to true to enable file transfer.

filetransferurl

(string)
HTTP URI used for file transfer. By default Mizutech service is used which is provided for free with the webphone.

displaynotification

(number)
Show notifications in phone notification bar (usually on the top corner of your phone).
0:Never
1:On event
2:Always
Default is 1.

displayvolumecontrols

(boolean)
Always display volume controls when in call.
Default is false.

displayaudiodevice

(boolean)
Always display audio device when in call.
Default is false.

displaypeerdetails

(string)
Specify where to display the information returned by scurl displaypeerdetails.
It can be used display details about the peers from your CRM such as full name, address or other details.
(Useful in call-centers and for similar usage)
Possible values:
0: show on call page (instead of contact picture)
1: on new page
div id: display on the specified DIV element

savetocontacts

(number)
Whether to (automatically) add new unknown called numbers to your contact list.
0:No
1:Ask
2:Yes (will not ask for a contact name)
Default is 1.

hasincomingcall

(boolean)
Whether to display a popup about incoming calls in certain engines.
Set to false to disable (in this case make sure that you handle the incoming call alert from your HTML/JS if required).
Default is true.

header

(string)
Header text displayed for users on top of softphone windows.
Default is empty.

footer

(string)
Footer text displayed for users on the bottom of softphone windows.
Default is empty.

version

(string)
Version number displayed for users.
Default is empty (will load the built-in version number)

messagepopup

(string)
Display custom popup for user once.
Default is empty.

showsynccontactsmenu

(number)
This is to allow contact synchronization between mobile and desktop.
-1=don't show
0=show Sync option in menu and Contacts page (if no contacts available)
1=show in menu only
Default is 1

defcontacts

(string)
Set one or more contacts to be displayed by default in the contact list.
Name and number separated by comma and contacts separated by semicolon:
Example: defcontacts: 'John Doe,12121;Jill Doe,231231'

extraoption

(string)
Custom parameters can be set in a key-value pair list, separated by semicolon Ex: displayname=John;
Default is empty.

logsendto

(number)
Specify allowed actions on the logs page.
0: no options (users will still be able to copy-paste the logs)
1: upload (default)
2: email launch (the email address set by the "supportmail" parameter or support@mizu-voip.com if not set)

Custom HTTP API and links

You can integrate the webphone with your server code using your custom HTTP (AJAX) API URI's. Just set the following settings to point to your API entries which will be called automatically as the webphone state machine changes:

- **scurl_onstart**: will be called when the webphone is starting
- **scurl_onoutcallsetup**: will be called on outgoing call init
- **scurl_onoutcallringing**: will be called on outgoing call ring
- **scurl_onoutcallconnected**: will be called on outgoing call connect
- **scurl_onoutcalldisconnected**: will be called on outgoing call disconnect with call details (CDR)
- **scurl_onincallsetup**: will be called on incoming call
- **scurl_onincallringing**: will be called on incoming call ring
- **scurl_onincallconnected**: will be called on incoming call connect
- **scurl_onincalldisconnected**: will be called on incoming call disconnect with call details (CDR)
- **scurl_oninchat**: will be called on incoming instant message
- **scurl_onoutchat**: will be called on outgoing instant message
- **scurl_setparameters**: will be called after "onStart" event(url) and can be used to provision the webphone from server API. The answer should contain parameters as key/value pairs, ex: username=xxx,password=yyy
- **scurl_displaypeerdetails**: will be called at the beginning of incoming and outgoing calls to return details about the peer from your server API (like full name, address or other details from your CRM). It will be displayed at the location specified by the "displaypeerdetails" parameter. You can return any string as clear text or html which can be displayed as-is.

For example: scurl_onoutcallsetup: http://mydomain.com/myapi/?user=USERNAME&called=CALLEDUMBER
(Your API will be called each time the webphone user makes an outgoing call)

You can integrate the included softphone user interface with your website and/or VoIP server HTTP API (if any) by using the following parameters:

- **advertisement:** Advertisement URL, displayed on bottom of the softphone windows.
- **supportmail:** Company support email address.
- **supporturl:** Company support URL.
- **newuser:** New user registration http request OR link.
- **forgotpasswordurl:** Will be displayed on login page if set.
- **homepage:** Company home page link.
- **accounturi:** Company user account page link.
- **recharge:** Recharge http request (pin code must be sent) or link.
- **p2p:** Phone to phone http request or link.
- **callback:** Callback http request or link (For example: http://yourdomain.com/callback?user=USERNAME)
- **sms:** SMS http request.

- **creditrequest:** Balance http request, result displayed to user.
- **ratingrequest:** Rating http request, result displayed for user on call page.
- **helpurl:** Company help link.
- **licenseurl:** License agreement link.
- **extramenuurl:** Link specifying custom menu entry. Will be added to main page (dialpad) menu.
- **extramenutxt:** Title of custom menu entry. Will be added to main page (dialpad) menu.

Example credit http request: http://domain.com/balance/?user= USERNAME

Parameters can be treated as **API requests** (specially interpreted) or **links** (to be opened in built-in webview). For http API request the value must begin with asterisk character: "*http://domain.com/...." For example if the "newuser" is a link, then it will be opened in a browser page; if it's an API http request (begins with *), then a form will be opened in the softphone with fields to be completed.

- The followings are always treated as API request: creditrequest, ratingrequest
- The followings can be links OR API http requests: newuser, recharge, p2p, callback, sms
- The rest will be treated always as links (opened in built-in webview or separate browser tab)

You can also use keywords in these settings strings which will be replaced automatically by the webphone. The following keywords are recognized:

- DEVICEID: unique identifier for the client device or browser
- SESSIONID: session identifier
- USERNAME: sip account username. preconfigured or entered by the user
- PASSWORD: sip account password
- CALLEDNUMBER: dialed number
- PEERNUM: other party phone number or SIP uri
- PEERDETAILS: other party display name and other available details
- DIRECTION: 1=outgoing call, 2=incoming call
- CALLBACKNR,PHONE1, PHONE2: reserved
- PINCODE: reserved. will be used in some kind of requests such as recharge
- TEXT: such as chat message
- STATUS: status messages: onLoad, onStart, callSetup, callRinging, callConnected, callDisconnected, inChat, outChat
- MD5SIMPLE: md5 (pUser + ":" + pPassword)
- MD5NORMAL: md5 (pUser + ":" + pPassword+":"+randomSalt)
- MD5SALT: random salt

For API request, the webphone will try to fetch the result using the following techniques (first available): AJAX/XHTTP, CORS, JSONP and websocket (if available).

Parameter security

The following methods can be used to secure the webphone usage:
-set the loglevel to 1 (with loglevel 5 also the password is written in the logs)
-don't hardcode the password if possible (let the users to enter it) or if you must hardcode it then use encryption and/or obfuscation
-restrict the account on the VoIP server (for example if the webphone is used as a support access, then allow to call only your support numbers)
-instead of password, use the MD5 and the realm parameters if possible (and this can also passed in encrypted format to be more secure)
-instead of preconfigured parameters you can also use the javascript api (start, register)
-use https (secure http / TLS)
-for parameter encryption/obfuscation you can use XOR + base64 with your built-in key, prefixed with the "encrypted__3__" string.
-secure your VoIP server (account limits, rate-limits, balance limits, fraud detection)

User interface Skin/Design

You can use the webphone with or without a user interface.

The webphone is shipped with a few ready to use open source user interfaces such as a softphone and click to call skins. Both of these can be fully customized or you can modify their source to match your needs. You can also create any custom user interface using any technique such as HTML/CSS and bind it to the webphone javascript API.

The default user interface for the softphone and other included apps can be easily changed by modifying parameters or changing the html/css. For simple design changes you don't need to be a designer. Colors, branding, logo and others can be

set by the settings parameters.
Also you can easily create your own app user interface from scratch with any tool (HTML/CSS or others) and call the webphone Java Script API from your code.

Quick/Basic skin change

Just use the "colortheme" parameter to make quick and wide changes.
The have a look at the "User interface related" parameters (described in the "Parameters" section) and change them after your needs (set logo, branding, others).

We can also send you a softphone with your preferred skin. For this just set your customization on the online designer form and send us the parameters.
We can also send you fully customized and branded web softphone with your preferences. For this just send us the customization details.

Advanced skinning

Web developers/designers can easily modify the existing skins or create their own.
For the softphone application all the HTML source code can be found in "softphone.html" file as a single-page application model. The style sheet structure is as follows:
- The jQuery mobile Theme Roller generated style sheet can be found in this file: "css\themes\wphone_1.0.css".
  Current jQuery mobile version is 1.4.2.
Using the Theme roller, you can create new styles: http://themeroller.jquerymobile.com/
- The style sheet which overrides the "generated" one (in which all the customizations are defined) is "css/mainlayout.css".
-You can also manually edit the html and css file with your favorite editor to change it after your needs
You can change the followings for easy color changes:

mainlayout.css: color to replace: #1d1d1d with urlparam: bgcolor
wphone_1.0.css: color to replace: #333 with urlparam: buttoncolor
wphone_1.0.css: color to replace: #373737 with urlparam: buttonhover
wphone_1.0.css: color to replace: #22aadd with urlparam: tabselectedcolor
mainlayout.css: color to replace: #31b6e7 with urlparam: fontctheme
mainlayout.css: color to replace: #ffffff with urlparam: fontcwhite
wphone_1.0.css: color to replace: sans-serif with urlparam: fontfamily

Note: If you are using the webphone as a javascript library then you can customize the "choose engine" popup in "css\pmodal.css".

If you have different needs or don't like the default skins, just create your own from scratch and call the webphone JavaScript API from your code. Using the API you can easily add VoIP call capabilities also to existing website or project with a few function calls as described in the "Java Script API" section below.

Integration and customization

The webphone is a flexible VoIP web client which can be used for various purposes such as a dialer on your website, a click to call button for contacts or integrated with your web application (contact center, CRM, social media or any other application which requires VoIP calls)
The webphone can be customized by its numerous settings, webphone API's and by changing its HTML/CSS.

Deploy:
The webphone can be deployed as a static page (just copy the webphone file to your website), as a dynamic page (with dynamically generated settings) or used as a JavaScript VoIP library by web developers. You can embed the webphone to your website in a div, in an iFrame, as a module or as a separate page. The webphone settings can be set also by URL parameters so you can just launch it from a link with all the required settings specified.

VoIP platform:
All you need to use the webphone is a SIP account at any VoIP service provider or your own softswitch/PBX.
Free SIP accounts can be obtained from numerous VoIP service providers or you can use our service. (Note that free accounts are free only for VoIP to VoIP calls. For outbound pstn/mobile you will need to top-up your account).
If you wish to host it yourself then you can use any SIP server software. For example FreePBX for linux or the advanced / free VoIP server for windows by Mizutech. We can also provide our WebRTC to SIP gateway (for free with the Advanced or Gold license) if your softswitch don't have support for WebRTC and you need a self-hosted solution.

Technical settings:
The most important parameter that you will need to set is the "serveraddress" which have to be set to the domain or IP:port of your SIP server.
If you wish, you might change also other sip account, call-divert or VoIP engine related settings after your needs.

Integration:
You can integrate the webphone with your web-site or web-application:
-using your web server API
-and/or using the webphone client side JavaScript API to insert any business logic or AJAX call to your server API
The webphone doesn't depend on any framework (as it is a pure client side library) but you can integrate it with any server side framework if you wish (PHP, .NET, NodeJS, J2EE or any server side scripting language) or work with it only from client side (from your JavaScript)

Custom application:
For deep changes or to create your unique VoIP client or custom application you will need to use the JavaScript API

Branding:
Since the webphone is usually used within your website context, your website is already your brand and no additional branding is required inside the webphone application itself. However the softphone skin (if you are using this turn-key GUI) has its own branding options which can be set after your requirements. Additionally you can change the webphone HTML/CSS design after your needs if more modifications are required.

On request, we can send your webphone build already preconfigured with your preferences.
For this just answer the points from the voip client customization page (as many as possible) and send to us by email. Then we will generate and send your webphone build within one work-day. All the preconfigured parameters can be further changed by you via the webphone settings.

Of course, this is relevant only if you are using a skin shipped with the webphone, such as the softphone.html. Otherwise you can create your custom solution using the webphone library with your unique user interface or integrate into your existing website.

Click to call

You can use the webphone with
You can use the webphone library to implement your custom click-to-call solution or use one of the skin templates for click to call.

The webphone package contains a ready to use click to call solution.
Just copy the whole webphone folder to your website, set the parameters in the webphone_api.js file and use it from the click2call_example.html.
Rewrite or modify after your needs with your custom button image or you can just use it via a simple URI or link such as:
http://www.yourwebsite.com/webphonedir/clicktocall.html?wp_serveraddress=YOURSIPDOMAIN&wp_username=USERNAME&wp_password=PASSWORD&wp_callto=CALLEDNUMBER&wp_autoaction=1

This will launch the click to call page and will initiate the call automatically.
If you need to better integrate click to call into your website, then you just have a code to your website and adjust the serveraddress, username, password, callto parameters. For this you can use the click2call example from the webphone package (modify it after your needs) or follow the below guide:

Step by step guide to add click to call button to your web page

Put the below code in your web page's <head> section:

```
<link rel="stylesheet" href="css/click2call/click2call.css" />
   <script src="js/click2call/click2call.js"></script>
   <script>
      /**Configuration parameters*/
      webphone_api.parameters['serveraddress'] =  '';
      webphone_api.parameters['username'] = '';
      webphone_api.parameters['password'] = '';
      webphone_api.parameters['md5'] =  '';
      webphone_api.parameters['realm'] = '';
      webphone_api.parameters['callto'] = '';
      webphone_api.parameters['autoaction'] = 1;
</script>
```

Copy this html element in you page, where you want the click to call button to show up:

```
<div id="c2k_container_0" title=""><a href="tel://CALLTO" id="c2k_alternative_url">CALLTO</a></div>
```

Customize the button
Customization options can be found in click2call.js file located in js/click2call/ folder.
The following customizations are available:
- button color (for call and hang up states)
- text displayed on the button (for call and hang up states)
- button width, height and corner radius
- chat window default state: open or collapsed

The styling can be further customized from click2call.css located in css/click2call/ folder.

Use as a chat window
The click to call button can also be used as a chat window. This is controlled by the "autoaction" parameter: 1=call, 2=chat.
The chat window can also be opened by accessing the menu and selecting the Chat item.
The menu can be access by right clicking or by long clicking on the button.

Floating button
The click to call can also be used as a floating button on your page. The floating related configurations can be found in click2call.js file located in js/click2call/ folder.
To enable floating, set the "float_button" config to true and specify two direction coordinates for the floating. For example to have a floating button on the top right corner of your page, located from 100 pixels from the top and 10 pixels from the right:
var float_button = true;
var float_distance_from_top = 100;
var float_distance_from_right = 10;

Multiple instances
To add more than one click to call button to a page, include the script part in the <head> section once, and copy the container increasing the id index number for every instance.
ex:

These id indexes must be unique and increasing.
The callto parameter can be set as the title attribute of the element.

JavaScript API

About

You can use the webphone javascript library in multiple ways for many purposes:

- create your own web dialer
- add click to call functionality to your webpage
- add VoIP capability to your existing web project or website
- integrate with any CRM, callcenter client or other projects
- modify one of the existing projects to achieve your goal (see the included softphone and click to call examples) or create yours from scratch
- and many others

The public JavaScript API can be found in "webphone_api.js" file, under global javascript namespace "webphone_api".
To be able to use the webphone as a javascript VoIP library, just copy the webphone folder to your web project and add the webphone_api.js to your page.

Basic example

```
<head>
	//Include the webphone_api.js to your webpage
	<script src="webphone_api.js"></script>
</head>
<body>
<script>
	//Wait until the webphone is loaded, before calling any API functions
	webphone_api.onLoaded(function () {

		//Set parameters (Replace upper case worlds with your settings. Alternatively these can be also preset in your html or passed as URL parameters
		webphone_api.setparameter('serveraddress', SERVERADDRESS);
		webphone_api.setparameter('username', USERNAME);
		webphone_api.setparameter('password', PASSWORD);
		//See the “Parameters” section below for more options

		//Start the webphone (optional but recommended)
		webphone_api.start();

		//Make a call (Usually initiated by user action, such as click on a click to call button. Number can be extension, SIP username, SIP URI or mobile/landline phone)
		webphone_api.call(NUMBER);

		//Hang-up (usually called from “disconnect” button click)
		webphone_api.hangup();

		//Send instant message (Number can be extension, SIP username. Usually called from a “send chat” button)
		webphone_api.sendchat(NUMBER, MESSAGETEXT);
	});
	//You should also handle events from the webphone and change your GUI accordingly (onXXX callbacks)
</script>
</body>
```

See the webphone package for more examples. You should check especially the tech demo (techdemo_example.html / techdemo_example.js).

Note: If you don't have JavaScript/web development experience, you can still fully control and customize the webphone:

- *by its numerous configuration options which can be passed also as URL parameters*
- *from server side as described above*
- *we can also send ready to use fully customized web softphone with preconfigured settings, branding and integration with your web and VoIP server*

Functions

Use the following API calls to control the webphone:

setparameter (param, value)

Any additional parameters must be set before start/register/call is called.

getparameter (param)

Return type: string
Will return value of a parameter if exists, otherwise will return empty string.

start()

Optionally you can "start" the phone, before making any other action.
In some circumstances the initialization procedure might take a few seconds (depending on usable engines) so you can prepare the webphone with this method to avoid any delay when the user really needs to use by pressing the call button for example.
If the serveraddress/username/password is already set and auto register is not disabled, then the webphone will also register (connect) to the SIP server upon start.
If start is not called, then the webphone will initialize itself the first time when you call some other function such as register or call.
The webphone parameter should be set before you call this method (preset in the js file or by using the setparameter function). See the "Parameters" section for details.

register ()

Optionally you can "register". This will "connect" to the SIP server if not already connected by the start method.

unregister ()

Unregister SIP account from VoIP server.
Note: Unregister is called also automatically at browser close so usually there is no need to call this explicitly.

call (number)

Initiate call to a number, sip username or SIP URI.
Perhaps this is the most important function in the whole webphone API.

videocall (number)

Initiate a video call to a number, sip username or SIP URI.
(Will failback to a simple voice call if video is not supported by peer, by the server or gateway. It should always work between WebRTC endpoints if peers has a camera device)

hangup ()

Disconnect current call.

accept ()

Connect incoming call.

reject ()

Disconnect incoming call.
(You can also use the hangup() function for this)

ignore ()

Silently ignore incoming call.

forward (number)

Forward incoming call to the specified number (phone number, username or extension)

dtmf (dtmf)

Send DTMF message by SIP INFO or RFC2833 method (depending on the "dtmfmode" parameter).
Please note that the dtmf parameter is a string. This means that multiple dtmf characters can be passed at once and the webphone will streamline them properly. Use the space character to insert delays between the digits.
The dtmf messages are sent with the protocol specified with the “dtmfmode” parameter.
Example: API_Dtmf(-2," 12 345 #");

mute (state, direction)

Mute current call.
Pass true for the state to mute or false to un-mute.
The direction can have the following values:

0: mute in and out
1: mute out (speakers)
2: mute in (microphone)

hold (state)

Hold current call. This will issue an UPDATE or a reINVITE.
Set state to true to put the call on hold or false to un-hold.

transfer (number)

Transfer current call to number which is usually a phone number or a SIP username. (Will use the REFER method after SIP standards).
You can set the mode of the transfer with the “transfertype” parameter.

conference (number)

Add people to conference.
If number is empty than will mix the currently running calls (if there is more than one call in progress).
Otherwise it will call the new number (usually a phone number or a SIP user name) and once connected will join with the current session.
Note: you can also use the webphone with your server conference rooms/conference bridge. In this way, there is no need to call this function (just make a normal call to your server conference bridge/room access number)

dtmf (dtmfstring)

Send DTMF message by SIP INFO or RFC2833 method (depending on the "dtmfmode" parameter).
Please note that the dtmf parameter is a string. This means that multiple dtmf characters can be passed at once and the webphone will streamline them properly. Use the space character to insert delays between the digits.
Example: API_Dtmf(-2," 12 345 #");

sendchat (number, msg)

Send a chat message. (SIP MESSAGE method as specified in RFC 3428)
Number can be a phone number or SIP username/extension number.

sendsms (number, msg, from)

Send a SMS message if your provider/server has support for SMS.
The number parameter can be any mobile number.

The msg is the SMS text.
The from is the local user phone number and it is optional.
SMS can be handled on your server by:
-converting normal chat message to SMS automatically if the destination is a mobile number
-or via an HTTP API (you can specify this is to the webphone as the “sms” parameter)

voicerecord (start, url)

Start/stop voice recording.
Set the start parameter to true for start or false to stop.
The url is the address where the recorded voice file will be uploaded as described by the voicerecupload setting.

audiodevice()

Open audio device selector dialog (built-in user interface).

getaudiodevicelist(dev, callback)

Call this function and pass a callback, to receive a list of all available audio devices.
For the dev parameter pass 0 for recording device names list or 1 for the playback or ringer devices.

getaudiodevice(dev, callback)

Call this function and pass a callback, to receive the currently set audio device.
For the dev parameter pass 0 for recording device or 1 for the playback or ringer device.

getaudiodevice(dev, devicename, immediate)

Select an audio device. The devicename should be a valid audio device name (you can list them with the API_GetAudioDeviceList call)
For the dev parameter pass 0 for recording device or 1 for the playback or ringer device
The "immediate" parameter can have the following values:

0: default
1: next call only
2: immediately for active calls

getvolume(dev, volume)

Set volume (0-100%) for the selected device. Default value is 50% -> means no change
The dev parameter can have the following values:

0 for the recording (microphone) audio device
1 for the playback (speaker) audio device
2 for the ringback (speaker) audio device

setvolume(dev, volume)

Call this function, passing a callback and will return the volume (0-100%) for the selected device.
The dev parameter can have the following values:

0 for the recording (microphone) audio device
1 for the playback (speaker) audio device
2 for the ringback (speaker) audio device

setsipheader(header)

Set a custom sip header (a line in the SIP signaling) that will be sent with all messages.
Can be used for various integration purposes (for example for sending the http session id).
You can also set this with applet parameter (customsipheader)

getsipheader(header, callback)

Call this function passing a callback.
The passed callback function will be called with one parameter, which will be the string value of the requested sip header.

setline (line)

Will set the current channel. (Use only if you present line selection for the users. Otherwise you don't have to take care about the lines as it is handled automatically).Currently available only for Java and NS plugin.

getline ()

Return type: number
Will return the current active line number. This should be the line which you have set previously except after incoming and outgoing calls (the webphone will automatically switch the active line to a new free line for these if the current active line is already occupied by a call). Currently available only for Java and NS plugin.

isregistered ()

Return type: boolean
Return true if the webphone is registered ("connected") to the SIP server.
Note: you can track the phone state machine also with the events callbacks.

isincall ()

Return type: boolean
Return true if the webphone is in call, otherwise false.
Note: you can track the phone state machine also with the events callbacks.

ismuted ()

Return type: boolean
Return true if the call is muted, otherwise will return false.

isonhold ()

Return type: boolean
Return true if the call is on hold, otherwise will return false.

isencrypted ()

Check if communication channel is encrypted: -1=unknown, 0=no, 1=partially, 2=yes, 3=always

checkpresence (userlist)

Will receive presence information as events: PRESENCE, status,username,displayname,email (displayname and email can be empty)
Userlist: list of sip account username separated by comma.

setpresencestatus (status)

Function call to change the user online status with one of the followings strings: Online, Away, DND, Invisible, Offline (case sensitive)

getenginename ()

Returns the currently used engine name as string: "java", "webrtc", "ns", "app", "flash", "p2p", "natividial".
Can return empty string if engine selection is in progress.
Might be used to detect the capabilities at runtime (for example whether you can use the below jvoip function or not)

delsettings (level)

Delete stored data (from cookie, config file and local-storage).
For the level parameters the following are defined:

1: just settings file
2: delete everything: settings, contacts, call history, messages

jvoip(name, jargs)

If engine is Java or the NS Service plugin, then you can access the full java API as described in the JVoIP SDK documentation.
Parameters:
Name: name of the function
Jargs: array of arguments passed to the called function. Must be an array, if API function has parameters. If API function has no parameters, then it can be an empty array, null, or omitted altogether.
For example the API function: API_Call(number) can be called like this: webphone_api.jvoip('API_Call', [number]);

getlogs ()

Returns a string containing all the accumulated logs by the webphone (the logs are limited on size, so old logs will be lost after long run).
More details about logs can be found here.

getstatus ()

Returns the webphone global status. The possible returned texts are the same like for getEvenetsnotifications.
You might use the events described below instead of polling this function.

Events

The following callback functions can be used to receive event from the webphone such as the webphone state machine status (registered/call init/call connected/disconnected) and other important events and notifications:

getEvents (callback)

This function returns ALL events from the webphone including sip stack state, notifications, events and logs.
Call this function once and pass a callback, to receive important events, which should be displayed for the user and/or parsed to perform other actions after your software custom logic. For the included softphone and click to call these are already handled, so no need to change, except if you need some extra custom actions or functionality.
See the "Notifications" section below for the details.

Example:

```
webphone_api.getEvents( function (event)
{
   // For example the following status means that there is an incoming call ringing from 2222 on the first line:
   // STATUS,1,Ringing,2222,1111,2,Katie,[callid]
   // parameters are separated by comma(,)
   // the sixth parameter (2) means it is for incoming call. For outgoing call this parameter is 1.

   // example for detecting incoming and outgoing calls:

varevtarray = event.split(',');

if (evtarray[0] === 'STATUS' &&evtarray[2] === 'Ringing')
   {
if (evtarray[5] === '1')
      {
         // means it is an outgoing call
         // ...
      }
else if (evtarray[5] === '2')
      {
         // means it is incoming call
         // ...
      }
   }
});
```

You might also check the simple_example.html included in the package.
If you will use this function, then most probably you will catch everything here and don't need to use the other events functions described below.

If you don't wish to deal with notification strings parsing, then you can use the functions below to catch the important events from the webphone in which you are interested in. Call them once, passing a callback:

onStart (callback)

The passed callback function will be called when the webphone is initializing.

onLoaded (callback)

The passed callback function will be called when all the modules of the webphone are loaded in the page.
All the initialization of the webphone can be done here.

onRegistered (callback)

The passed callback function will be called on registered (connected) to VoIP server.

onUnRegistered (callback)

The passed callback function will be called on unregistered (disconnected) from VoIP server.
Note: If user closes the webpage, then you might not have enough time to catch this event.

onCallStateChange (callback)

The passed callback function will be called on every call state change.
Parameters:

- status: can have following values: callSetup, callRinging, callConnected, callDisconnected
- direction: 1 (outgoing), 2 (incoming)
- peername: is the other party username
- peerdisplayname: is the other party display name if any

onChat (callback)

The passed callback function will be called when chat message is received.
Parameters:

- from: username, phone number or SIP URI of the sender
- msg: the content of the text message

onCdr (callback)

The passed callback function will be called at each call disconnect. You will receive a CDR (call detail record).
Parameters:

- caller: the caller party username
- called: called party username
- connect time: milliseconds elapsed between call initiation and call connect
- duration: milliseconds elapsed between call connect and hangup (0 for not connected calls. Divide by 1000 to obtain seconds)

onLog (callback)

The passed callback function will receive all the logs in real time. It can be used for debugging or for log redirection if the other possibilities don't fit your needs.

Notifications

"Notifications" means simple string messages received from the webphone which you can parse with the getEvents(callback) to receive notifications and events from the webphone about its state machine, calls statutes and important events.
Skip this section if you are not using the getEvents() function. (You can use the functions such as onRegistered/onCallStateChange/others to catch the important events in which you are interested in and completely skip this section about notification strings handling).

If you are using the getEvents() function then you will have to parse the received notification strings from your java script code. Each notification is received in a separate line (separated by CRLF). Parameters are separated by comma ',’. For the included softphone and click to call these are already handled, so no need to change, except if you need some extra custom actions or functionality.

The following messages are defined:

STATUS,line,statustext,peername,localname,endpointtype

Where line can be -1 for general status or a positive value for the different lines.

General status means the status for the “best” endpoint.
This means that you will usually see the same status twice (or more). Once for general phone status and once for line status.
For example you can receive the following two messages consecutively:

STATUS,1,Connected,peername,localname,endpointtype,peerdisplayname,[callid]
STATUS,-1,Connected

You might decide to parse only general status messages (where the line is -1).
The following **statustext** values are defined **for general status (line set to -1):**

- Initializing
- Ready
- Register…
- Registering…
- Register Failed
- Registered
- Accept
- Starting Call
- Call
- Call Initiated
- Calling…
- Ringing…
- Incoming…
- In Call (xxx sec)
- Hangup
- Call Finished
- Chat

Note: general status means the “best” status among all lines. For example if one line is speaking, then the general status will be “In Call”.

The following **statustext** values are defined **for individual lines** (line set to a positive value representing the channel number starting with 1):

- Unknown (you should not receive this)
- Init (started)
- Ready (sip stack started)
- Outband (notify/options/etc. you should skip this)
- **Register** (from register endpoints)
- Subscribe (presence)
- Chat (IM)
    - **CallSetup** (one time event: call begin)
- Setup (call init)
- InProgress (call init)
- Routed (call init)
- Ringing (SIP 180 received or similar)
    - **CallConnect** (one time event: call was just connected)
- InCall (call is connected)
- Muted (connected call in muted status)
- Hold (connected call in hold status)
- Speaking (call is connected)
- Midcall (might be received for transfer, conference, etc. you should treat it like the Speaking status)
    - **CallDisconnect** (one time event: call was just disconnected)
- Finishing (call is about to be finished. Disconnect message sent: BYE, CANCEL or 400-600 code)
- Finished (call is finished. ACK or 200 OK was received or timeout)
- Deletable (endpoint is about to be destroyed. You should skip this)
- Error (you should not receive this)

You will usually have to display the call status for the user, and when a call arrives you might have to display an accept/reject button.
For simplified call management, you can just check for the one-time events (CallSetup, CallConnect, CallDisconnect)

Peername is the other party username (if any)
Localname is the local user name (or username).
Endpointtype is 1 from client endpoints and 2 from server endpoints.
Peerdisplayname is the other party display name if any
CallID: SIP session id

For example the following status means that there is an incoming call ringing from 2222 on the first line:

STATUS,1,Ringing,2222,1111,**2**,Katie,[callid]

The following status means an outgoing call in progress to 2222 on the second line:

STATUS,2,Speaking,2222,1111,**1**,[callid]

To display the “global” phone status, you will have to do the followings:

1. Parse the received string (parameters separated by comma)
2. If the first parameter is “STATUS” then continue

3. Check the second parameter. It "-1" continue otherwise nothing to do
4. Display the third parameter (Set the caption of a custom html control)
5. Depending on the status, you might need to do some other action. For example display your "Hangup" button if the status is between "Setup" and "Finishing" or popup a new window on "Ringing" status if the endpointtype is "2" (for incoming calls only; not for outgoing)

If the "jsscripstats" is on (set to a value higher than 0) then you will receive extended status messages containing also media parameters at the end of each call:
STATUS,1,Connected,peername,localname,endpointtype,peerdisplayname,**rtpsent,rtprec,rtploss,rtplosspercet,serverstats_if_received**,[callid]

PRESENCE,peername,presence

This notification is received for incoming chat messages.
Line: used phone line
Peername: username of the peer
Presence: presence status string; one of the followings:
CallMe,Available,Pending,Other,CallForward,Speaking,Busy,Idle,DoNotDisturb,Unknown,Away,Offline,Exists,NotExists,Unknown

CHAT,line,peername,text

This notification is received for incoming chat messages.
Line: used phone line
Peername: username of the sender
Text: the chat message body

CHATCOMPOSING,line,peername,composing

This notification might be received when the other peer start/stop typing (RFC 3994):
Line: used phone line
Peername: username of the sender
Composing: 0=idle, 1=typing

CHATREPORT,line,peername,status,text

This notification is received for the last outgoing chat message to report success/fail:
Line: used phone line
Peername: username of the sender
Status: 0=unknown,1=sending,2=successfully sent,3=failed to send
Text: failure reason (if Status is 3)

CDR,line,peername,caller,called,peeraddress,connecttime,duration,discparty

After each call, you will receive a CDR (call detail record) with the following parameters:
Line: used phone line
Peername: other party username, phone number or SIP URI
Caller: the caller party name (our username in case when we are initiated the call, otherwise the remote username, displayname, phone number or URI)
Called: called party name (our username in case when we are receiving the call, otherwise the remote username, phone number or URI)
Peeraddress: other endpoint address (usually the VoIP server IP or domain name)
Connecttime: milliseconds elapsed between call initiation and call connect
Duration: milliseconds elapsed between call connect and hangup (0 for not connected calls. Divide by 1000 to obtain seconds.)
Discparty: the party which was initiated the disconnect: 0=not set, 1=local, 2=peer, 3=undefined
Disconnect reason: a text about the reason of the call disconnect (SIP disconnect code, CANCEL, BYE or some other error text)

START,what

This message is sent immediately after startup (so from here you can also know that the SIP engine was started successfully).
The what parameter can have the following values:
"api" -api is ready to use
"sip" –sipstack was started

EVENT,TYPE,txt

Important events which should be displayed for the user.
The following TYPE are defined: EVENT, WARNING, ERROR
This means that you might receive messages like this:
WPNOTIFICATION,EVENT,EVENT,any text NEOL \r\n

POPUP,txt

Should be displayed for the users in some way.

ACTION,txt

Various custom messages. Ignore.

LOG,TYPE,txt

Detailed logs (may include SIP signaling).
The following TYPE are defined: EVENT, WARNING, ERROR

VAD,parameters

Voice activity.
This is sent in around every 2000 milliseconds (2 seconds) by default from java and NS engines (configurable with the vadstat_ival parameter in milliseconds) if you set the "vadstat" parameter to 3 or it can be requested by API_VAD. Also make sure that the "vad" parameter is set to at least "2".
This notification can be used to detect speaking/silence or to display a visual voice activity indicator.

Format:
VAD,local_vad: ON local_avg: 0 local_max: 0 local_speaking: no remote_vad: ON remote_avg: 0 remote_max: 0 remote_speaking: no

Parameters:
local_vad: whether VAD is measured for microphone: ON or OFF
local_avg: average signal level from microphone
local_max: maximum signal level from microphone
local_speaking: local user speak detected: yes or no

remote_vad: whether VAD is measured from peer to speaker out: ON or OFF
remote_avg: average signal level from peer to speaker out
remote_max: maximum signal level from peer to speaker out
remote_speaking: peer user speak detected: yes or no

Other notifications

Format: messageheader, messagetext. The followings are defined:

"**CREDIT**" messages are received with the user balance status if the server is sending such messages.
"**RATING**" messages are received on call setup with the current call cost (tariff) or maximum call duration if the server is sending such messages.
"**MWI**" messages are received on new voicemail notifications if you have enabled voicemail and there are pending new messages
"**PRESENCE**" peer online status
"**SERVERCONTACTS**" contact found at local VoIP server
"**NEWUSER**" new user request
"**ANSWER**" answer for previous request (usually http requests)

Version history

Major changes by release are listed here:

Version 0.2 (February12, 2015)

-internal beta version with basic skin and basic call functionality

Version 0.6 (July3, 2015)

-engines: WebRTC beta and Java Applet v.4.0
-more SIP settings
-early beta version with basic SIP call functionality

Version 0.9 (September 9, 2015)

-call-divert functionalities (voicemail, transfer, others)
-conference
-NS (Native Service or Plugin)
-examples and documentation

-better skinning
-better OS/browser handling
-automatic engine selection
-WebRTC stable incoming and outgoing calls
-upgrade to Java Applet v.5.6
-more JavaScript API

Version 1.0 (November 6, 2015)

-new: flash engine
-WebRTC improvements
-stable API, modules and file structure
-improved auto engine selection
-chat
-app engine
-secondary engines (p2p, native dial, callback)
-custom builds based on customer settings

Version 1.2 (January 18, 2016)

-callback API for simplified API (use simple call back instead of notification string parsing)
-server API for the webphone state machine (so you can easily catch all important events from server code)
-WebRTC engine upgrade to latest version
-presence (not fully standard compliant yet but working)
-added file transfer
-new missed call/chat notifications
-http vs https bug fixes
-NS engine availability from https
-reset setting parameter and API
-last call detailed statistics
-called number normalization
-one-way audio fix on WebRTC
-WebRTC fix for Android
-many other improvements and bug fixes

Version 1.3 (February 05, 2016)

-new: audio device selection
-new: favorite or block contact
-new: setsipheader/getsipheader
-improved: capability call special url's on events (server API integration)
-improved: number rewrite rules
-improved: feedback for file transfer
-fix: ns engine unregister on webpage close
-fix: increase cseq for re-invite
-other improvements and bug fixes

Version 1.4 (April 11, 2016)

-new: WebRTC to SIP gateway (free as both software and service for all our webphone customers)
-new: TURN (WebRTC works now even if all UDP is blocked and only port TCP 80 is allowed)
-new: auto codec convert when necessary (for example to G.729 from WebRTC)
-new: App engines for iOS and Android
-new: WebRTC on Android
-new: HTTP to HTTPS gateway (used automatically if hosting website is not secure which is required by Chrome for WebRTC)
-new: WebRTC caller-id
-improved: WebRTC NAT handling
-improved: STUN
-improved: end to end encryption
-improved: softphone skin
-fix: java freezing improvements
-fix: WebRTC caller-id

Version 1.5 (April 27, 2016)

-new: call recording (voicerecupload)
-new: 8 new call-divert related settings and API's
-new: callcenter integration
-improved: engine selection
-improved: VoIP over TCP using TURN only when necessary
-improved: usage on local LAN's

-improved: WebRTC (various fixes)
-fix: auto engine select related bugs, unnecessary java popups
-fix: NS engine discover issues
-fix: mute/unmute, hold/unhold
-fix: CTRL+C, CTRL+V in the softphone skin
-more than 20 other bug fixes and small improvements especially engine detect/choose related

Version 1.6 (July 1, 2016)

-new: video
-new: conference rooms (server assisted)
-new: audio device list, get, set functions
-new: web call-me
-new: peer to peer media auto discover
-new: call forward
-new: auto WebRTC discover (for example it can detect automatically if webrtc is enable in Asterisk and other servers)
-new: softphone skin now can be inserted also in a DIV (previously it was working only in iframe)
-new: callback (you can specify a callbacknumber parameter if your server has a callback access number)
-new: sip outbound proxy setting
-new: call transfer options
-new: CDR records after calls (can be easily posted to server API)
-new: group chat
-improved: click to call
-improved: presence
-improved: voicemail
-improved: conference
-improved: voice recording
-improved: android native dialer auto-configuration
-improved: themes (color theme/skinning)
-improved: TURN and STUN handling and auto-discovery
-improved: user interface integration (div, popup, flying, others)
-improved: chat (reliability, smiles, file-transfer, groups)
-improved: NS engine versioning and auto upgrade
-improved: webrtc engine
-fix: voip engine auto select related issues
-fix: settings save/restore
-fix: init delay
-fix: ns engine localhost certificate
-fix: https/wss issues
-fix: more than 44 bug-fixes mostly based on customer feedback and additional tests

Version 1.7 (July 20, 2016)

This is a bug-fix only release with 14 various bug fixes and minor improvements. No new features.

FAQ

How to get my own webphone?

1. Try from your desktop or webserver by downloading the webphone package or try the online demo.
2. If you like it, we can send your own licensed build within one workday on your payment.
   The pricing can be found here. For the payment we can accept PayPal, credit card or wire transfer.
   Contact Mizutech at webphone@mizu-voip.comwith the following details:
   -your VoIP and/or web server(s) address (ip or domain name or URL)
   -your company details for the invoice (if you are representing a company)

For the old “websipphone” (Java Applet based webphone) users:
Please note that this is a separate product and purchase or upgrade cost might be required. See the upgrade guide about the details. The old java applet based websipphone have been renamed to “VoIP Applet” and we will continue to fully support it as a separate product: https://www.mizu-voip.com/Software/Softphones/VoIPApplet.aspx
You can easily upgrade your old java applet websiphone to this universal webphone by following the steps described below in the “How to upgrade from the old java applet websipphone” FAQ.

What about support?

We offer support and maintenance upgrades to all our customers. Guaranteed supports hours depend on the purchased license plan and are included in the price.
Email to webphone@mizu-voip.com with any issue you might have.
Please include the followings with your message:

- exact issue description
- screenshot if applicable
- detailed logs
- optionally a description about how we can reproduce the problem with valid sip test account(s)

If the included support period with your license is expired, it can be increased by 2 years for around $600 (Note: This is completely optional. There is no need for any support plan to operate your webphone). For gold partners we also offer priority, phone and 24/7 emergency support.
Direct support is provided for the common features (voice calls, chat, dtmf, hold, forward and others) and common OS/browsers (Windows/Linux/Android/MAC OS IE, Firefox, Chrome, Opera) and not for extra features (such as presence, fax, video) and exotic OS/Browsers (such as FreeBSD, Chromium, Konqueror). The webphone should work also with these OS/browsers, but we are not testing every release against these exotic platforms.

What I will receive once I have made the payment for the webphone?

You will receive the followings:

- the webphone software itself (the webphone files including the engines, javascript API, html5/css skins and examples)
- the ready-to-use/turn-key softphone skin and click to call button
- latest documentations and code examples
- invoice (on request or if you haven't received it before the payment)
- support on your request according to the license plan

Can Mizutech do custom development if required?

Yes.
You can fully customize the webphone yourself by its numerous configuration options. However if you have some specific requirement which you can't handle yourself, please contact us at webphone@mizu-voip.com. Contact us only with webphone/VoIP specific requirements (not with general web development/design related requests as these can be handled by any web developer and we are specialized for VoIP).

Should I have programmer skills to be able to use the webphone?

No. The webphone can be deployed by anybody. If you already have a website, then you should be able to copy-paste and rewrite the example HTML codes. Some basic Java Script knowledge is required only if you plan to use the Java Script API (although there are copy-paste examples for the API usage also)

Web server requirements

You can use any kind of web server to host the webphone files. Just copy the webphone folder to your webhost and you are ready to go.

Some more details:
All the functionality of the webphone is implemented on client side (JavaScript running in users browser) so there is no any application specific requirements for the webserver. You can **use any web server** software (IIS, nginx, Apache, NodeJS, Java) on any OS (Linux, Windows, others). You can integrate the webphone with any server side framework if you wish (.NET, PHP, java servlet, J2EE, NodeJS and others). Integration tasks are up to you, and it can be done multiple ways such as dynamic webphone configuration per request, dynamic URL rewrite (since the webphone accepts parameters also in URL's), or add more server side app logic via your custom HTTP API which can be called from webphone (for example on call, on call disconnect or other events; the webphone has callbacks for these to ease this kind of integrations). All these are optional since you can implement any kind of app logic also on client side from JavaScript if you need so.
We recommend deploying the webphone to a secure site (**https**) otherwise the latest Chrome and Opera doesn't allow WebRTC.
If you can't enable https on your webhost for some reason, then we can host your webphone if you wish on a secure white-label domain for free.

Depending on the client browser and the selected engine, the webphone might have to download some platform specific binaries. (These are found in the "native" folder). Make sure that your web server allows the download of these resource types by allowing/adding the following **mime types** to your webserver configuration if not already added/allowed:

- extension: .mxml MIME type: application/octet-stream
- extension: .exe MIME type: application/octet-stream
- extension: .jar MIME type: application/java-archive
- extension: .jnilib MIME type: application/java-archive
- extension: .dll MIME type: application/x-msdownload

- extension: .so MIME type: application/octet-stream
- extension: .dylib MIME type: application/octet-stream
- extension: .swf MIME type: application/x-shockwave-flash

You can easily test if works by trying to download these files typing their exact URI in the browser such as:
http://yourwebsite.com/webphone/native/webphone.jar
(The browser should begin to download the file, otherwise the jar mime type is still not allowed on your webserver or you entered an incorrect path or webserver doesn't serve files from the specified folder)

Is it working with my VoIP server?

The webphone works with any SIP capable voip server including Asterisk, Trixbox, Huawei, Cisco, Mizu, 3CX, Voipswitch, Brekeke and many others. You don't necessarily need to have your own SIP server to use the webphone as you can use any SIP account(s) from any VoIP provider.

The webphone is using the SIP protocol standard to communicate with VoIP servers and sofswitches. Since most of the VoIP servers are based on the SIP protocol today the webphone should work without any issue. Some modules (WebRTC and Flash) might require specific support by your server or a gateway to do the translation to SIP, however these modules are optional, gateway software are available for free and also mizutech includes its own free tier service (usable by default with the webphone).
If you have any incompatibility problem, please contact webphone@mizu-voip.com with a problem description and a detailed log (loglevel set to 5). For more tests please send us your VoIP server address with 3 test accounts.

What are the main benefits?

Using the Mizu webphone you can have a single solution for all platforms with the same user interface and API. No individual apps have to be maintained anymore for different platforms such as a Windows Installer, a Web application, Google Play app for Android and other binaries.

- Unlike traditional softphones, the webphone can be embedded in webpages
- Single unified JavaScript API and custom web user interface
- Easy customization for all kind of use-case (by the parameters and optionally by using the API)
- Compatible with all browsers (IE, Firefox, Safari, Opera, Chrome, etc) and all OS (Windows, Linux, MAC, Android, etc)
- Compatible with your existing VoIP server or any SIP service
- Works also behind corporate firewalls (auto tunnel over TCP 80 if needed)
- Easy to use and easy to deploy (copy-paste HTML code)
- Easy integration with your existing infrastructure since it is using the open SIP/RTP standards
- Easy integration with your existing website design
- Proprietary SIP/RTP stack guarantees our strong long term and continuous support
- Support for all the common VoIP features

Usage examples

- As a browser phone
- Integration with other web or desktop based software to add VoIP capabilities
- A convenient dialer that can be offered for VoIP endusers since it runs directly from your website
- Callcenter VoIP client for agents/operators (easy integration with your existing software)
- Embedded in VoIP devices such as PBX or gateways
- Click to call functionality on any webpage
- VoIP conferencing in online games
- Buy/sell portals
- WebRTC SIP client or WebRTC softphone
- Salesforce help button
- Social networking websites , facebook phone
- Integrate SIP client with jQuery, Drupal, joomla, WordPress, angularjs, phpBB, vBulletin and others as a plugin, module or API
- As an efficient and portable communication tool between company employees
- VoIP service providers can deploy the webphone on their web pages allowing customers to initiate SIP calls without the need of any other equipment directly from their web browsers
- Customer support calls (VoIP enabled support pages where people can call your support people from your website)
- VoIP enabled blogs and forums where members can call each other
- VoIP enabled sales when customers can call agents (In-bound sales calls from web)
- Java Script phone or WebRTC SIP client
- Turn all phone numbers into clickable links on your website
- Integrate it with any Java applications (add the webphone.jar as a lib to your project)
- HTTP Call Me buttons
- Remote meetings

- HTML5 VoIP
- Asterisk integration (or with any other PBX)

Folders and file structure

- "css" folder: - style sheets used in skin (GUI). The style of the skin can be changed by editing "mainlayout.css" file
- "css/themes" folder: - jQuery mobile specific cascading style sheets and images
- "flash" folder: - flash engine
- "images" folder:- images used in skin (GUI)
- "js/softphone" folder: - GUI files. For every jQuery mobile "page" there is an equivalent JavaScript file, which handles the behavior of the page. Also there is a string resource file (stringres.js) which contains all the text displayed to the user.
- "js/lib" folder: - softphone JavaScript library files
- "lib" folder: - softphone java and native plugin library files
- "old_skin " folder: - old webphone skin, which is used only in old browsers, ex: IE 6
- "raw" folder: -raw resources, mainly audio files
- the root folder contains the following files:
    - "favicon.ico": - web page favicon
    - "index.html": - simple integration example of the softphone
    - "oldapi_support.js": - backward compatibility with old skin. Useful for cases where the webphone was integrated using the "old" JavaScript API.
    - "simple_example.html": - simple usage example of softphone SDK
    - "softphone.html": - GUI html file
    - "webphone_api.js": - public Javascript API of the softphone

How to upgrade from the old java applet websipphone?

This new webphone has an easy to use API, however if you wish to keep your old code, you can do so with minimal changes as we created a compatibility layer for your convenience. Follow the next steps to upgrade to our new webphone:

1. The root folder of the new webphone is the folder, in which "webphone_api.js" and "softphone.html" files are located.
2. Copy the contents of the new webphone root folder, in the same folder where the old webphone's .html file is (merge "images" and "js" folders, if asked upon copy process).
3. In the <head> section of the .html file, where the old webphone is, replace line:
<script type="text/JavaScript" src="js/wp_common.js"></script>"
with the following lines:
<script type="text/JavaScript" src="webphone_api.js"></script>
<script type="text/JavaScript" src="oldapi_support.js"></script>

Note: Don't remove or add any webphone related Javascript file imports.
"jquery-1.8.3.min.js " file will be imported twice, but that is how it supposed to be, in order for the upgrade to work correctly.

For old webphone customers: please note that this new webphone is a separate product and purchase or upgrade cost might be required. The old java applet webphone have been renamed to “VoIP Applet” and we will continue to fully support it. More details can be found in the wiki.

Is the webphone depends on Mizutech services?

No, the webphone can be used on their own as a fully self-hosted solution, connecting to your VoIP server directly (Java, NS and App), via WebRTC or via Flash so you will have a solution fully owned/controlled/hosted by you without dependency on our services.

By default the webphone might use some of the services provided by mizutech to ease the usage and to make it a turn-key solution without any extra settings to be required from your side. None of these are critical for functionality; all of them can be turned off or changed. The following services might be used:

- Mizutech license service: demo, trial or free versions are verified against the license service to prevent unauthorized usage. This can be turned off by purchasing a license (Your final build will not have any DRM and will continue to work even if the entire mizutech network is down).
- WebRTC to SIP gateway: if your server doesn’t have WebRTC capabilities but you enable the WebRTC engine in the webphone then it might use the Mizu WebRTC to SIP gateway service. Other possibilities are listed here.
- Flash to SIP gateway: rarely used (only when there is no better engine then Flash). Just turn it off (by setting the “enginepriority_flash” parameter to 0) or install your own RTMP server and specify its address.
- STUN server: by default the webphone might use the Mizutech STUN service. You can change this by changing the “stunserveraddress” to your server of choice (there are a lot of free public STUN services or you can run your own: stable open source software exists for this and it requires minimal processing power and network bandwidth as STUN is basically just a simple ping-pong protocol sending only a few short UDP packets and it is not a critical service).
- TURN server: by default the webphone might use the Mizutech TURN service which can help firewall/NAT traversal in some circumstances (rarely required). You can specify your own turn server by setting the “turnserveraddress” parameter (if TURN is required at all).

- JSONP: if you set some external API to be used by the softphone skin (such as for user balance or call rating requests) and your server can't be contacted directly with AJAX requests due to CORS, then the API calls might be relayed by the Mizutech JSONP or websocket relay. To disable this, make sure that the domain where you are hosting the webphone can access your domain where your API is hosted.
- HTTPS proxy: with the WebRTC engine if you are using the webphone from Chrome and your website is not secured (not https) then the webphone might reload itself via the Mizu HTTPS proxy. To disable this, host your webphone on HTTPS if you wish to use WebRTC from Chrome.
- Tunneling/encryption/obfuscation: In some conditions the webphone might use the Mizu tunneling service to bypass VoIP blockage and firewalls. This is usually required in countries where VoIP is blocked or behind heavy firewalls with DPI and you can turn it off by setting the "usetunneling" parameter to 0.

Note: if you are using the webphone on a local LAN then these services are not required and are turned off automatically (so the webphone will not try to use these if your VoIP and/or Web server are located on local LAN / private IP).

What software do I need to be able to use/deploy the webphone?

- Webserver (rented, hosted) to host the webphone files
- Some server side scripts if more customization/changes are necessary that is permitted by the parameters of from java script
- Voip access (one of the followings):
  - account(s) at any VoIP service provider OR
  - buy a VoIP server software or hardware (Cisco, Mizu, Brekeke, others)
  - a free or open source VoIP server (asterisk, Trixbox, OpenSIPS, others) OR
  - rent a softswitch (SaaS)
- Optionally a WebRTC and/or Flash service

How to handle WebRTC?

WebRTC is one of the important engines built into the webphone. You have several options to deal with WebRTC:

1. Don't use WebRTC at all. There are other engines built into the webphone which can be used most of the time. There are only a few circumstances when the only available engine would be WebRTC. (Although WebRTC is convenient for enduser since it doesn't need any browser plugin in browsers where it is supported). To completely disable WebRTC, set the enginepriority_webrtc setting to 0.
2. Check if your VoIP server already has WebRTC support. Most modern VoIP server already has implemented WebRTC (including mizu VoIP server, Asterisk and others) or you might just need to add/enable a module on your server for this, so chances are high that your VoIP server can handle WebRTC natively. Just set the webrtcserveraddress setting to point to your server websocket address
3. Use the free Mizutech WebRTC to SIP service tier. This is enabled by default and it might be suitable for your needs if you don't have too much traffic over WebRTC (the webphone will automatically start to boost the priority for other engines when you are over the free quote)
4. Use the mizutech WebRTC to SIP gateway software. We are providing this software for free for our webphone customers. (You just have to setup this near your SIP server)
5. Use any third party WebRTC to SIP gateway: There are few free software which is capable to do this task for you, including Asterisk and Dubango. (However if you don't have any of these installed yet, then we recommend our own gateway as mentioned above)
6. Use the Mizutech WebRTC to SIP paid service. We provide dedicated WebRTC to SIP conversion services for a monthly fee if required.

The webphone can be used as a WebRTC softphone by increasing the enginepriority_webrtc to 3 or 4 (in this case it will use the other engines only when WebRTC is not supported by the browser).
Note: Latest Chrome and Opera browsers requires secure connection to allow WebRTC for both your website (HTTPS) and websocket (WSS).

How to handle Flash?

Chances are high that you don't need Flash at all. In the rare circumstances when the only available engine is Flash, the webphone can automatically use the Mizutech Flash to SIP free service. In case if somehow you wish to drive all your traffic over Flash, then you can install a Red5 server (open source free software) to handle the translation between RTMP and SIP/RTP, then set the rtmpserveraddress to point to your flash media server and increase the value of the enginepriority_flash setting.

How to handle Java, Native and App engines?

These engines doesn't need any special server side support and they works with old legacy SIP (all SIP servers) without any extra settings or software. When the webphone uses one of these engines, there is a direct connection between the engine (running in the user's browser) and your VoIP server, without involving any intermediary relay (RTP can also flow directly between the endusers, bypassing your server. This is up to your server settings and its NAT handling features)

What are the advantages over pure WebRTC solutions?

WebRTC is becoming a trendy technology but it has a lot of disadvantages and problems:

- It is a moving target. The standards are not completed yet. Lots of changes are planned also for 2016. Edge just start to add a different “ORTC” implementation
- Incompatibility. WebRTC has known incompatibility issues with SIP and there are a lot of incompatibilities even between two WebRTC endpoint as browsers has different implementation and different codec support
- Not supported by all browsers. No support in Edge, IE and Safari. No support on iOS and MAC. No support on older Android phones.
- Lack of popular VoIP codec such as G.729 which can be solved only by expensive server side transcoding
- It is a black-box in the browser with browser specific bugs and a restrictive API. You have little control on what is going in the background
- A WebRTC to SIP gateway required if your VoIP server don’t have built-in support for WebRTC
- Adds unneeded extra complexity. The server has to convert from the websocket protocol to clear SIP and from DTLS to RTP

Luckily the Mizu webphone has some more robust engines that can be used without these limitations and by default will prioritize these over WebRTC whenever possible, depending on available browser capabilities and user willingness. (Small non-obtrusive notification might be displayed for the enduser when a better engine is available or if a user can upgrade with one-click install).

One of the main advantages of the Mizu webphone is that it can offer alternatives for WebRTC, so you can be sure that all your VoIP users are served with the best available technology, regardless of their OS and browser.

However we do understand that WebRTC is comfortable for the endusers as it doesn’t require any extra plugin if supported by the user browser. The mizu webphone takes full advantage of this technology and we provide full support for WebRTC by closely following the evolution of the standards.

With a WebRTC only client you would miss all the benefits that could be offered by a standard SIP/RTP client connecting directly to your VoIP server with native performance, full SIP support with all the popular VoIP codecs and without the need for any protocol conversion, directly from enduser browser.

The webphone is not loading/starting

Make sure that either Java or WebRTC is available in your browser or you can use the NS engine (Android and Windows) or app (Android and iOS). Also make sure that the .jar and .exe mime types are allowed on your webserver so the browsers are able to download platform specific native binaries.

Can’t connect to SIP server

Make sure that:
-you have set your SIP server address:port correctly (from the user interface or “serveraddress” parameter in the webphone_api.js file)
-make sure that you are using a SIP username/password valid on your SIP server
-make a test from a regular SIP client such as mizu softphone or x-lite
-send us a detailed client side log if still doesn’t work with loglevel set to 5 (from the browser console or from softphone skin help menu)

Failed outgoing calls

Make a test call first from a simple SIP client such as mizu softphone or x-lite

By default only the PCMU,PCMA, G.729 and the speex ultra-wideband codec’s are offered on call setup which might not be enabled on your server or peer UA. You can enable all other codec’s (PCMA, GSM, speex narrowband, iLBC and G.729) with the use_xxx parameters set to 2 or 3 (where xxx is the name of the codec: use_pcma=2, use_gsm=2, use_speex=2,use_g729=2,use_ilbc=2). Some servers has problems with codec negotiation (requiring re-invite which is not support by some devices). In these situations you might disable all codec’s and enable only one codec which is supported by your server (try to use G.729 if possible. Otherwise PCMU or PCMA is should be supported by all servers)

If still doesn’t work send us a detailed client side log with loglevel set to 5 (from the browser console or from softphone skin help menu)

Calls are disconnecting

If the calls are disconnecting after a few second, then try to set the “invrecordroute” parameter to “true” and the “setfinalcodec” to 0.
If the calls are disconnecting at around 100 second, then most probably you are using the demo version which has a 100 second call limit.
If the calls are disconnecting at around 3600 second (1 hour) and you are using the java scrip API then please check the httpsessiontimeout parameter (you need to call the API_HTTPKeepAlive function periodically from a javascript timer)

Known limitations

-Not all the listed features are available from all engines (the webphone automatically handle these differences internally)
-Some platforms currently have very limited VoIP support available from browsers. The most notable is iOS where the default browser (Safari) lacks any VoIP support. The webphone tries all the best to work around about these by using its secondary engines offering call capabilities for also for users on these platforms
-WebRTC registers also with incorrect user/pwd when used with Mizutech service. The WebRTC to SIP Mizutech service doesn’t check user authentication for the REGISTER requests at this moment (the proper authentication is left for your SIP server, so the users will not be able to make calls with invalid user/pwd)
-Some features might not work correctly between WebRTC and SIP. This is not a webphone limitation, but it depends completely on server side (your softswitch or gateway responsible for WebRTC-SIP protocol conversion)

-Some features require also proper server side support to work correctly. For example call hold, call transfer and call forward. See your VoIP server documentation about proper setup.

OS/browser related issues

There are many browser and OS related bugs either in the browser itself or in the plugins used for VoIP (native/webrtc/java/flash). Most of the issues are handled automatically by the webphone by implementing workarounds for a list of well-known problems. Rarely there is no any way to circumvent such issues from the webphone itself and needs some adjustment on server or client side.

Some chrome versions only use the default input for audio. If you have multiple audio devices and not the default have to be used changing on chrome, Advanced config, Privacy, Content and media section will fix the problem.

Some linux audio drivers allow only one audio stream to be opened which might cause audio issues in some circumstances. Workaround: change audio driver from oss to alsa or inverse. Other workarounds: Change JVM (OpenJDK); Change browser.

If the java (JVM or the browser) is crashing under MAC at the start or end of the calls, please set the “cancloseaudioline” parameter to 3. You might also set the "singleaudiostream” to 5. If the webphone doesn’t load at all on MAC, then you should check this link.

Incoming calls might not have audio in some circumstances when the webphone is running in Firefox with the WebRTC engine using the mizu WebRTC to SIP gateway.

One way audio problem on OSX 10.9 Maverick / Safari when using the Java engine: Safari 7.0 allows users to place specific websites in an "Unsafe Mode" which grants access to the audio recording. Navigate to "Safari ->Preferences -> Security (tab) and tick "Allow Plug-ins" checkbox. Then depending on safari version:
-from the Internet plug-ins (Manage Website Settings)" find the site in question and modify the dropdown to "Run In Unsafe Mode".
-or go to Plug-in Settings and for the option "When visiting other websites" select "Run in Unsafe Mode". A popup will ask again, click "Trust"
You will be asked to accept the site's certificates or a popup will ask again, click "Trust". Alternatively, simply use the latest version of the Firefox browser.

Java in latest Chrome is not supported anymore (the webphone will select WebRTC by default).
If for some reason you still wish to force Java, then in versions prior September 1, 2015 it can still be re-enabled:
Go to this URL in Chrome: chrome://flags/#enable-npapi (then mark activate)
Or via registry: reg add HKLM\software\policies\google\chrome\EnabledPlugins /v 1 /t REG_SZ /d java

(By default the webphone will handle this automatically by choosing some other engine such as WebRTC unless you forced java by the engine priority settings)

Using the webphone on local LAN

The webphone can be used also on local LAN’s (when your VoIP server or Web server or both are on your private network).
-The NS and Java engines will connect directly to your server as a normal SIP softphone does.
-For WebRTC to work you will need a WebRTC to SIP gateway on your LAN or your PBX have to support WebRTC, otherwise this engine will not be picked up (this is handled automatically).
-The webphone could use the Mizutech STUN, TURN, JSONP and HTTPS gateway services by default, however these are not required on local LAN’s (the webphone will detect this automatically and will not try to use these services while on local LAN).

Using the webphone without internet connection

The webphone can be used also without internet connection with some limitations:
-WebRTC in Chrome needs https (secure http), which will work only with a local policy, otherwise the browser will not be able to verify the SSL certificate against public CA. If you can’t setup a local CA or browser rule for this, just disable WebRTC (or use Firefox instead of Chrome if you need WebRTC without certificate).
-Java applets need to be signed and on startup the JVM will have to pass the code verification signature. Workaround: Just disable the Java engine.
-The NS engine can be used from unsecured http in local LAN’s with no issues.
These circumstances will be automatically handled by the webphone, always selecting the best suitable engine unless you change the engine priority related settings.

Using the webphone in controlled environment

If you are using the webphone in a controlled environment (where you have control over the clients, such as call-centers) then you might force the NS or Java engines by disabling or lowering the priority for the WebRTC engine (enginepriority_webrtc = 1). This is because NS and Java are more native for SIP/RTP and might have better quality, more features and lower processing costs on your server. The big advantage of WebRTC is that it can work without any extra plugin download/install, however in a controlled environment you can train your users (such as the callcenter agents) to allow and install the NS engine when requested and this one-time extra action will reward with long term quality improvement.

RTP statistics

For RTP statistics increase the log level to at least 3 and then after each call longer than 7 seconds you should see the following line in the log:
EVENT, rtp stat: sent X rec X loss X X%.
If you set the "loglevel" parameter to at least "5" than the important rtp and media related events are also stored in the logs.
You can also access the details about the last call from the softphone skin menu "Last call statistics" item.

NAT settings

In the SIP protocol the client endpoints have to send their (correct) address in the SIP signaling, however in many situations the client is not able to detect it's correct public IP (or even the correct private local IP). This is a common problem in the SIP protocol which occurs with clients behind NAT devices (behind routers). The clients have to set its IP address in the following SIP headers: contact, via, SDP connect (used for RTP media). A well written VoIP server should be able to easily handle this situation, but a lot of widely used VoIP server fails in correct NAT detection. RTP routing or offload should be also determined based in this factor (servers should be always route the media between 2 nat-ed endpoint and when at least one endpoint is on public IP than the server should offload the media routing). This is just a short description. The actual implementation might be more complicated.

With the WebRTC engine make sure that the STUN and TURN settings are set correctly (by default it will use mizu services which will work fine if your server is on the public internet).

For NS and Java engines you may have to change the webphone configuration according to your SIP server if you have any problems with devices behind NAT (router, firewall).
If your server has NAT support then set the use_fast_stun and use_rport parameters to 0 and you should not have any problem with the signaling and media for webphone behind NAT. If your server doesn't have NAT support then you should set these settings to 2. In this case the webphone will always try to discover its external network address.

Example configurations:
If your server can work only with public IP sent in the signaling:
-use_rport 2 or 3
-use_fast_stun: 1 or 2

If your server can work fine with private IP's in signaling (but not when a wrong public IP is sent in signaling):
-use_rport9
-use_fast_stun: 0
-optionally you can also set the "udpconnect" parameter to 1

Asterisk is well known about its bad default NAT handling. Instead of detecting the client capabilities automatically it relies on pre-configurations. You should set the "nat" option to "yes" for all peers.
More details:
http://www.voip-info.org/wiki/view/NAT+and+VOIP
http://www.voip-info.org/wiki/view/Asterisk+sip+nat
http://www.asteriskguru.com/tutorials/sip_nat_oneway_or_no_audio_asterisk.html

Server failover/fallback

Use the following settings if you have 2 voip servers:

- serveraddressfirst: the IP or domain name of the first server to try
- serveraddress: the IP or domain name of the next server
- autotransportdetect: true
- enablefallback: true

In this way the webphone will always send a register to the first server first and on no answer will use the second server (the "first" server is the "serveraddressfirst" at the beginning, but it can change to "serveraddress" on subsequent failures to speed up the initialization time)

Alternatively you can also use SRV DNS records to implement failover or load balancing, or use a server side load balancer.

I have call quality issues

Call quality is influenced primarily by the followings:

- The engine used (Java and NS tends to have the best quality)
- Codec used to carry the media (wideband has better quality)
- Network conditions (check your upload speed, packet loss, delay and jitter)

- Hardware: enough CPU power and quality microphone/speaker (try a headset, try on another device)
- AEC and denoise availability

If you have call quality issues then the followings should be verified:

- whether you have good call quality using a third party softphone from the same location (try X-Lite for example). If not, than the problem should be with your server, termination gateway or bandwidth issues.
- make sure that the CPU load is not near 100% when you are doing the tests
- make sure that you have enough bandwidth/QoS for the codec that you are using
- change the codec (disable/enabled codec's with the "codec" parameter)
- deploy the mediaench module (for AEC and denoise). (Or disable it if it is already deployed and you have bad call quality)
- webphone logs (Check audio and RTP related log entries. Also check the statistics after call disconnect.)
- wireshark log (Check missing or duplicated packets)

I have one way audio

1. Review your server NAT related settings
2. Set the "setfinalcodec" parameter to 0 (especially if you are using Asterisk or OpenSIPS)
3. Check stun and turn settings (might be used for WebRTC if your server is not on the public internet, doesn't route the RTP or you need peer to peer media routing)
4. Set use_fast_stun, use_fast_ice and use_rport to 0 (especially if you are using SIP aware routers). If these don't help, set them to 2.
5. If you are using Mizu VoIP server, set the RTP routing to "always" for the user(s)
6. Make sure that you have enabled all codec's
7. Make a test call with only one codec enabled (this will solve codec negotiation issues if any)
8. Try the changes from the next section (Audio device cannot be opened)
9. If you still have one way audio, please make a test with any other softphone from the same PC. If that works, then contact our support with a detailed log (set the" loglevel" parameter to 5 for this)

Audio device cannot be opened

If you can't hear audio, and you can see audio related errors in the logs (with the loglevel parameter set to 5), then make sure that your system has a suitable audio device capable for full duplex playback and recording with the following format:
PCM SIGNED 8000.0 Hz (8 kHz) 16 bit mono (2 bytes/frame) in little-endian

If you have multiple sound drivers then make sure that the system default is workable or set the device explicitly from the webphone (with the "Audio" button from the default user interface or using the "API_AudioDevice" function call from java-script)

To make sure that it is a local PC related issue, please try the webphone also from some other PC.

You might also try to disable the wideband codec's (set the use_speexwb and use_speexuwb parameters to 0 or 1).

Another source for this problem can be if your sound device doesn't support full duplex audio (some wrong Linux drivers has this problem). In this case you might try to disable the ringtone (set the "playring" parameter to 0 and check if this will solve the problem).

If these doesn't help, you might set the "cancloseaudioline" parameter to 3 and/or the "singleaudiostream" to 5.

No ringback tone

Depending on your server configuration, you might not have ringback tone or early media on call connect.
There is a few parameter that can be used in this situation:

- set the "changesptoring" parameter to 3
- set the "natopenpackets" parameter to 10
- set the "earlymedia" parameter to 3
- change the "use_fast_stun" parameter (try with 0 or 2)

One of these should solve the problem.

Chat is not working

Make sure that your softswitch has support for IM and it is enabled. The webphone is using the MESSAGE protocol for this from the SIP SIMPLE protocol suite as described in RFC 3428.

Most Asterisk installations might not have support for this by default. You might use Kamailio for this purpose or any other softswitch (most of them has support for RFC 3428).

If subsequent chat messages are not sent reliably, set the "separatechatdiag" parameter to 1.

The webphone doesn't receive incoming calls

To be able to receive calls, the webphone must be registered to your server by clicking on the "Connect" button on the user interface (or in case if you don't display the webphone GUI than you can use the "register" parameter with supplied username and password, or from the JavaScript API)

Once the webphone is registered, the server should be able to send incoming calls to it.

The other reason can be if your server doesn't handle NAT properly.
Please try to start the webphone with use_fast_stunparameter set to 0 and if still not works then try it with 2.

If the calls are still not coming, please send us a log from the webphone (set the loglevel parameter to 5) and also from the caller (your server or remote SIP client)

What is the best codec?

This depends on the circumstances and there is no such thing as the "best codec". All commonly used codec's present in the webphone are well tested and suitable for IP calls with optimized priority order by default, regarding to environment (client device, bandwidth, server capabilities).
This means that usually you don't need to change any codec related settings except if you have some special requirement.
Between webphone users (or other IP to IP calls) you should prefer wideband codec's (this is why you just always leave the speex wideband and ultra wideband with the highest priority if you have calls between your VoIP users. These will be picked for IP to IP calls and simply omitted for IP to PSTN calls).
Otherwise G.729 provides both good quality and low bandwidth if this codec is available for you.
G.711 (PCMU/PCMA) is always supported and they offer good call quality using some more bandwidth then G.729.
To calculate the bandwidth needed, you can use this tool. You might also check this blog entry: Codec misunderstandings
With the webphone you don't need to change the codec settings except if you need some special settings. With the default settings the webphone is already optimized and will always choose and negotiate the "best" available codec.

Caller ID display

For outgoing calls the Caller ID (CLI/A numberdisplay) is controlled by the server and the application at the peer side (be it a VoIP softphone or a pstn/mobile phone).

You can use the following parameters to influence the caller id display at the remote end:

- username (this is used for both SIP username and authentication username if sipusername is not set)
- sipusername (if this parameter is set, then the "sipusername" will be used for authentication and the "username" parameter as the SIP username)
- displayname (SIP display name)

If you set all these parameters, then it will be sent in the SIP signaling in the following way (see the uppercase worlds):

```
INVITE sip:called@sipdomain.com SIP/2.0
From: "DISPLAYNAME" <sip:USERNAME@sipdomain.com>;tag=xyz
Contact: "DISPLAYNAME"<sip:USERNAME@sipdomain.com>
Remote-Party-ID: "DISPLAYNAME" <sip:USERNAME@88.150.183.87>;party=calling;screen=yes;privacy=off
Authorization: Digest username="SIPUSERNAME",realm="sipdomain.com" …
```

Some VoIP server will suppress the CLI if you are calling to pstn and the number is not a valid DID number or the webphone account doesn't have a valid DID number assigned (You can buy DID numbers from various providers).
The CLI is usually suppressed if you set the caller name to "Anonymous" (hide CLI).
If required by your SIP server, you can also set a Caller Identity header as a "customsipheader" parameter. (P-Preferred-Identity/P-Asserted-Identity/Identity-Info)

For incoming calls the webphone will use the caller username, name or display name to display the Caller ID. (SIP From , Contact and Remote-Party-ID fields).

I got an upgrade for my feature/issue request, but nothings seems to be changed

Make sure that you are actually using the new version. Refresh the browser cache by pressing F5 or Ctrl+F5. Make sure that you don't have some caching proxy on the path. A sure way to bypass all caching is to change the server folder (deploy in a "test2" directory and launch from there). If you are using the NS engine, then you might need to upgrade the webphone service.

If your webphone is using the NS engine, then it might be possible that the PC is running an old version. This can be updated in the following ways:
-manually as described below
-set the minserviceversion parameter. If higher than the current installed version then it will ask the user to upgrade (one click install)

-auto-upgrade: the core of the ns engine is capable to auto upgrade itself if new versions are found (you can disable this by setting the “autoupgrade” parameter to 6)
(In the NS service there is a built-in SSL certificate for localhost. This is also capable for auto-upgrade when new certificates are found)

How to uninstall or (re)install the webphone service

In some situation under Windows OS the webphone might install an NT service named “Webphone” (This is the NS service plugin and it is installed only on user opt-in)

- Disabling: If you don’t wish to use the NS engine, you can just disable the service (set startup type to Manual and Stop the service) or set the enginepriority_ns to 0
- Uninstalling: The service has its own uninstaller, so you can easily uninstall it from the Add/Remove Programs control panel. It can be also removed with the –uninstall parameter. Example: C:\Program Files (x86)\WebPhoneService\WebPhoneService.exe –uninstall.
- Re(installing): The install can be done from the softphone skin by just going to menu -> settings -> advanced settings -> sip settings -> voip engine -> select the NS engine. That should offer the download of the new version (if the service is not already running, so if you need to install a new version, then you should uninstall or stop it first).
  You can also (re)install/upgrade manually by running the “WebPhoneService_Install.exe” from the webphone\native folder. (You can also download it from your webserver: http://yourdomain.com/path_to_webphone/native/WebPhoneService_Install.exe). Just run the executable and it will install the NS engine automatically (this should work even if the service is already running as it will automatically update your old version)

How to keep the webphone call between page loads?

There is no way to keep the webphone session alive between page loads. For this some different technique should be used, for example:

- run the webphone in an frame
- load your content dynamically (Ajax)
- run the webphone in a separate window/page

How to use the webphone via URL parameters?

The webphone can load its settings also from the webpage URL and perform various actions such as initiate a call. All the listed parameters can be used, prefixed with “wp_”.
Example to trigger a call with the softphone by html url parameters:
http://www.yourwebsite.com/webphonedir/softphone.html?wp_serveraddress=YOURSIPDOMAIN&wp_username=USERNAME&wp_password=PASSWORD&wp_callto=CALLEDNUMBER
Example to trigger a call with the click to call by html url parameters:
http://www.yourwebsite.com/webphonedir/clicktocall.html?wp_serveraddress=YOURSIPDOMAIN&wp_username=USERNAME&wp_password=PASSWORD&wp_callto=CALLEDNUMBER
Example trigger chat by html parameters
http://www.yourwebsite.com/webphonedir/softphone.html?wp_serveraddress=YOURSIPDOMAIN&wp_username=USERNAME&wp_password=PASSWORD&wp_sendchat=TEXT&wp_to=DESTINATION
Note: you should use clear password only if the account is locked on your server (can’t call costly outside numbers). Otherwise you should pass it encrypted or use MD5 instead.

Click to call from email signature

Just set your phone number in your email signature as a link (URL anchor) to the webphone click to call:
http://www.yourwebsite.com/webphonedir/clicktocall.html?wp_serveraddress=YOURSIPDOMAIN&wp_username=USERNAME&wp_password=PASSWORD&wp_callto=YOURNUMBER
In this way the phone number in your email signature will become a clickable link which will trigger the webphone and will call your number automatically on SIP.
Instead of the clicktocall.html, you can also use the softhone.html (or your custom webphone html).

For account username/password you should just create a special extension on your SIP server which is not authenticated and allows unrestricted calls to local extensions only (not to outbound/paid).

How can I set the engine to be used?

The best engine is selected by the webphone automatically based on circumstances (client device, OS, browser, network, server):
However the preferred engine can influenced on 3 levels:
-Choice presented to the user in some circumstances on startup (This is not always presented. The webphone will go with the best engine when there is a definitive winner, without asking the user)
-Engine settings in the user interface, so the enduser might change its own preferred engine
-Engine priority options in the configuration. You can set this in the “webphone_api.js” (enginepriority_xxx settings as discussed in this documentation Parameters section)

There should be very rare circumstances when the default engine selection algorithm should be changed. The webphone always tries to select the engine which will disturb the user the less (minimizing required user actions) and offers the best performance.
For example don't be inclined to disable Java for the sake of its age. Users will not be alerted to install Java by default. However if Java is already enabled in the user browser then why not to use it? Java can offer native like VoIP capabilities and there should be no reason to disable it.
We spent a considerable amount of work to always select the best possible engine in all circumstances. Don't change this unadvisedly, except if you have a good reason to use a particular engine in a controlled environment.

What are the "best" settings?

This is a question often asked by our customers about how to optimize the webphone for best call quality. The answer is rather simple for this question:
The best settings are the default settings. The default settings are optimized and should be preferred in almost all use cases except if you have some uncommon needs. You should change the default settings only if you have a good reason to do so.

How to set the webphone parameters dynamically?

If you need to integrate the webphone with your server (for example with a CRM) you might have to set different parameters regarding the session (for example different user credentials based on the currently logged-in user). There are 3 ways to do this:

1. With the client side JavaScript using the webphone setparameter API (get the parameters from you webapp or via ajax requests)
2. Just generate the URL (iframe or link) dynamically from your server side scripts with the parameters set as required (wp_username, wp_password and other URL parameters).
3. Set the "scurl_setparameters" setting to point to your server side http api which will have to return the details once called by the webphone. This will be called after "onStart" event and can be used to provision the webphone from server API. The answer should contain parameters as key/value pairs, ex: username=xxx,password=yyy.

How to get the logs?

The webphone can generate detailed logs for debugging purposes.
For this just set the "loglevel" setting to 5 (or enable logs from the user interface if any; this is already set to 5 by default in the demo versions).
Once enabled, you can see the logs in the browser console or in the softphone skin help menu (if you are using this GUI). If the Java engine is being used, then the logs will appear also in the Java console. You can also use the API: getlogs() and the onLog(callback) functions.
When contacting Mizutech support with any issue, please always attach the detailed logs: just send the output of the browser console (or you can find the same from the softphone skin help menu if you are using the softphone.html).
On Firefox and Chrome you can access the logs with the Ctrl+Shift+J shortcut (or Cmd+Shift+J on a Mac). On Edge and Internet Explorer the shortcut key is F12.

- *If the webphone is using the WebRTC engine then the browser console output will contain the most important logs.*
  If you are using the softphone skin, then better if you check the logs from the skin help menu because the number of lines are limited in the browser console.
  If you have voice issues (no voce, one side voice, delays) then you should get a detailed log. With Chrome this can be done by launching it like:
  *"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --enable-logging --v=4 --vmodule=*libjingle/source/talk/*=4 --vmodule=*media/audio/*=4*
  Then you can find the logs at: C:\Users\USER\AppData\Local\Google\Chrome\User Data\chrome_debug.log
- *If the webphone is using the Java engine, then a log window will appear if the "loglevel" is set to "5" and the "canopenlogview" to "true". Grab the logs also from this window (Ctrl+A, Ctrl+C, Ctrl+V) or from the Java console.*
- *If the webphone is using the NS engine on Windows, then some more detailed logs can be obtained from C:\Program Files (x86)\WebPhone_Service\WebPhone_Servicelog.dat and C:\Program Files (x86)\WebPhone_Service\content\native\webphonelog.dat. (C:\Program Files (x86)\WebPhone_Service is the default data directory which might be different on your PC. It might be located in the C:\Users\secondaryacc\AppData\Roaming\WebPhone_Service directory if the account doesn't have write access to Program Files).*

When sending logs to Mizutech support, please attach them as text files (don't insert in email body).

ERROR and WARNING messages in the log

If you set the loglevel higher than 1 than you will receive messages that are useful only for debug. A lot of ERROR and WARNING message cannot be considered as a fault. Some of them will appear also under normal circumstances and you should not take special attention for these messages. If there are any issue affecting the normal usage, please send the detailed logs to Mizutech support (webphone@mizu-voip.com) in a text file attachment.

Why I see RTP warning in my server log

The webphone will send a few (maximum 10) short UDP packets (\r\n) to open the media path (also the NAT if any).
For this reason you might see the following or similar Asterisk log entries: "WARNING[8860]: res_rtp_asterisk.c:2019 ast_rtp_read: RTP Read too short" or "Unknown RTP Version 1".
These packets are simply dropped by Asterisk which is the expected behavior. This is not a webphone or Asterisk error and will not have any negative impact for the calls. You can safely skip this.

You might turn this off by the "natopenpackets" parameter (set to 0). You might also set the "keepaliveival" to 0 and modify the "keepaliveival" (all these might have an impact on the webphone NAT traversal capability).

Resources

Mizu WebPhone homepage: https://www.mizu-voip.com/Software/WebPhone.aspx
Download: https://www.mizu-voip.com/Portals/0/Files/webphone.zip
Pricing: https://www.mizu-voip.com/Support/Webphonepricing.aspx
For help, contactwebphone@mizu-voip.com