2015

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
10/12/2015

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 Session Initiation Protocol, it can inter-operate with any other SIP-based networks allowing people to make true VoIP calls directly from their browsers. Compatible with all SIP softphones, devices, gateways, ATA's, proxies, and PBX such as Asterisk, Trixbox, Mizu, 3CX, Voipswitch, Cisco, Avaya, SER, Huawei and 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":

- Java applet
- WebRTC
- Service
- Flash
- App
- P2P
- Callback
- Native Dial

The engine is automatically selected by default based on OS, browser and server availability. The engine to use can be also set manually from the configuration or their priorities can be changes.
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, replaced or removed.

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/online_demo.html
You might have to enable the .jar, .exe, .swf, .dll, .so and .dylib mime types in your webserver if not enabled by default.
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.

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. For more customizations see the "Parameters" chapter. Optionally the settings can be provided by URL parameters or via the API.

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).
At first start the webphone might offer to enable or download a native plugin if no 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.

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:

- **simple_example**.html: a basic usage example. You might change/extend it to fit your needs
- **online_demo**.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
- **click2call**.html: a ready to use browser to SIP click to call solution. You might further customize after your needs
- **custom**: you can easily create any custom browser VoIP solution by using the webphone java script library

VoIP Service providers / Endusers / Webmasters

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 the softphone.html via an iframe to your website.
If you are interested in a click to call button, you can use the “click2call.html”.

The web softphone can be configured in "webphone_api.js" API file, which can be found in the root directory of the package. Configuration parameters can be set in "parameters" Javascript object. 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.

Note: you might have to enable the following mime types in your browser: .jar, .swt, .dll

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 your static page, changed runtime from JavaScript or dynamically set by any server side script such as PHP, .NET, J2EE or Node.js.

Simple example:

```
<head>
        //include the webphone_api.js to your webpage and set the issdk parameter to true
        <script src="webphone_api.js"></script>
        <script>webphone_api.parameters.issdk = true;</script>
</head>
<body>
        //set parameters
        webphone_api.setparameter('serveraddress', serveraddress);
        webphone_api.setparameter('username', username);
        webphone_api.setparameter('password', password);
        //make a call
        webphone_api.call(number);
        //hangup
        webphone_api.hangup();
        //send IM
        webphone_api.sendchat(number, message)
</body>
```

For more details, see the “JavaScript API” section below in this documentation.

Designers

Modify any of the existing skins or create your own in any technology with JavaScript binding such as HTML/CSS, Flash or others.

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/mizulayout.css".

Please note that for simple design changes you don’t need to be a designer. Colors, branding, logo and others can be set by the settings.

Demo

You can also try the functionality from our online webphone demo page:
http://www.mizu-voip.com/Portals/0/Files/webphone_online_demo.aspx

User interface

The webphone is shipped with a few user interfaces to ease the usage, such as a softphone and click to call user interface. 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.

Features

- Standard SIP client for voice calls (in/out), chat, conference and others
- Maximum browsers compatibility. Runs in all browsers with Java, WebRTC or native plugin support (Chrome, Firefox, IE, Edge, Safari, Opera)
- Includes 8 different technologies to make phone calls (engines): Java applet, WebRTC, Service, Flash, App, P2P, Callback, Native
- SIP and RTP stack compatible with any standard VoIP servers and devices like Cisco, Voipswitch, Asterix, softphones, ATA and others
- Protocols: SIP/SIPS, RTP/SRTP. Transport: UDP, TCP, TLS, HTTP, websocket
- NAT/Firewall support: stable SIP and RTP ports ,keep-alive, rport support, proxy traversal, fast ICE/fast STUN protocols and auto configuration
- RFC's: 2543, 3261, 2976, 3892, 2778, 2779, 3428, 3265, 3515, 3311, 3911, 3581, 3842, 1889, 2327, 3550, 3960, 4028, 3824, 3966, 2663, 3022 and others
- Supported methods: REGISTER, INVITE, reINVITE, ACK, PRACK, BYE, CANCEL, UPDATE, MESSAGE, INFO, OPTIONS, SUBSCRIBE, NOTIFY, REFER
- Codec: PCMU, PCMA, G.729, GSM, iLBC, SPEEX, OPUS (including wide-band HD audio)
- Call divert: redial, mute, hold, transfer, forward, conference
- IM/Chat (RFC 3428), DTMF, Voicemail (MWI)
- JavaScript API
- Integration with any website or application using any server side stack such as PHP, .NET, J2EE, Node.js or simple static page
- Balance display, call timer, inbound/outbound calls, Caller-ID display
- Branding and customization: Use with your own brand. Customizable user interface, skins and languages (with ready to use, modifiable skins)
- Flexibility: all parameters/behavior can be changed/controlled by URL parameters, preconfigured parameters and/or from java script

Requirements

Server side:

- Any Web server to host the files
- At least one SIP account at any VoIP service provider or your own SIP server (such as Asterisk, Trixbox, Voipswitch, 3CX, SER, Cisco and others)
- Optional: WebRTC capable SIP server or SIP to WebRTC gateway (free software is available and Mizutech also has a free service tier included)

Client side:

- Any modern browser supporting WebRTC, Java OR native plugins with JavaScript enabled (most browsers are supported)
- Audio device: headset or microphone and speakers (there are circumstances when this is not required, for example the user can receive a callback or P2P call from the VoIP server to its phone)

Compatibility:

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

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 our goal by implementing different "VoIP engines" targeting each OS/browser segment. 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 and OS/Browser support (this can be overridden by settings).

Java Applet
Based on our powerful JVoIP SDK, compatible with all JRE enabled browsers.
Pros:

- All SIP/media features are supported, all codecs including G.729 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 my ask the user permission to activate the Java plugin

WebRTC
A new technology for media streaming in modern browsers supporting common VoIP features.
Pros:

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

Cons:

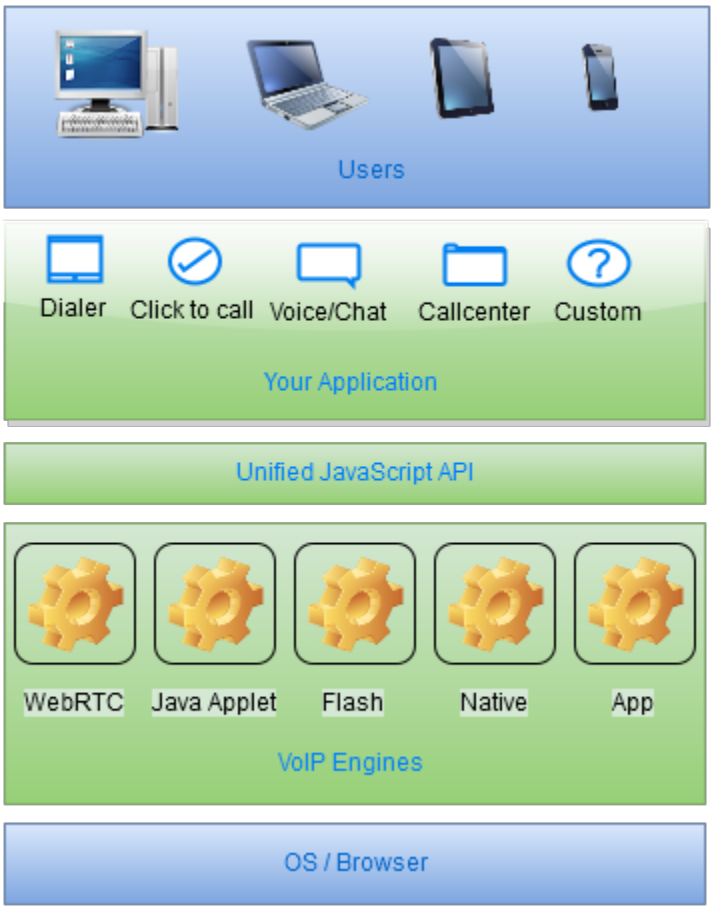
- It is a black-box in the browser with a restrictive API
- Lack of popular VoIP codec such as G.729 (this can be solved by CPU intensive server side transcoding)
- A WebRTC to SIP gateway may be required if your VoIP server don't have built-in support for WebRTC (there are free software for this and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

Flash
A browser plugin technology developed by Adobe with its proprietary streaming protocol. Rarely required.
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 and SIP/RTP
- Basic feature set



Service plugin
Our native VoIP engine implemented as a native service or browser plugin.
Pros:

- All features all supported, native performance

Cons

- Requires install (one click installer)

App
Some platforms don't have any suitable technology to enable VoIP in browsers. In these cases the webphone can offer to the user to install a native 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 as you defined by the webphone parameters.
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
P2P means server initiated phone to phone call. This is treated as a secondary (failback) engine if your VoIP server has such capability and no other engines are available.

Callback
Callback is a method to make cheap international calls triggering a callback from the VoIP server by dialing its callback access number. It is used only as a secondary engine if you set a callback access number in the settings and no normal VoIP call is possible.

Native Dial
This means native calls from mobile. This is a secondary "engine" 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.

The most important engines are: Java, WebRTC, Native 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.

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 above in the "Demo version" section) and are customized for your domain. All "mizu" and "mizutech" words 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 modifies)

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

Title, ownership rights, and intellectual property rights in the Software shall remain with MizuTech and/or its suppliers.
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.

Some audio codecs that can be used with the webphone (e.g. G.729) can be patented in your country and require you to pay royalties to their licensors (patent license per channel). Mizutech doesn't sell codec patent licenses (but the codecs are implemented in 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
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
Reverse engineer, decompile or disassemble the webphone
Modify the software in any way (except modifying the settings and the HTML/CSS skins)

Demo version

Demo version is downloadable which you can 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 in 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 minute after that you have to restart it or restart the browser
- can be blocked from Mizutech license service

Note: for the first few calls and in some circumstances the limitations might be weaker than described above, with less restrictions.
On request we can also send out demo with only the trial period limitation (will expire after several weeks of usage) and without the above demo limitations.

Download link: http://www.mizu-voip.com/Portals/0/Files/webphone.zip

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).

Parameters can be specified in the following ways:

- Preset in the "webphone_api.js" file, under "parameters" variable
- 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/window URL at startup)
- 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
- User input (You can let the use 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, 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. 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” applet parameter).
The rest of the parameters are optional and should be changed only if you have a good reason for it.

SIP account settings

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”
Default value is empty.

sipusername
(string)
This is the SIP username (A number/Caller-ID for the outgoing calls). The webphone will authenticate with this username on your server.
Default value is empty.

sippassword
(string)
SIP authentication password. The user will have to enter this password.
Default value is empty.

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

realm
(string)
Set the SIP realm if not the same with the serveraddress or domain.
Default 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 is set, otherwise no)
2: yes
Default value is 1.

Engine related

sdk
(boolean)
Must be "false" if used with the provided GUI (skin). Must be "true" if the softphone is used as SDK.
Can be configured ONLY in "webphone_api.js" parameters={ } variable.
Default value is false.

Engine priority:
By default the webphone will choose the “best” suitable engines based on OS/browser/server support. You can modify this with the following parameters:
enginepriority_java: 0=Disabled,1=Lower,2=Default,3=Higher,4=Force
enginepriority_webrtc: 0=Disabled,1=Lower,2=Default,3=Higher,4=Force
enginepriority_service: 0=Disabled,1=Lower,2=Default,3=Higher,4=Force
enginepriority_flash: 0=Disabled,1=Lower,2=Default,3=Higher,4=Force

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:80”
“wss://sip.mydomain.com:5062”
“ws://10.20.30.40:5065”
Default value is empty.

offeralternatebrowser
(boolean)
Offer alternate browser to install when user chooses engine.
Default is true.

pc_aletrnatebrowserurl
(string)
PC alternate browser download url.
Default is "https://www.mozilla.org/en-US/firefox/new/".

android_aletrnatebrowserurl
(string)
Android alternate browser download url.
Default is "https://play.google.com/store/apps/details?id=com.android.chrome".

offersoftphone
(boolean)
Offer native softphone to install when user chooses engine.
Download links can be configured with "android_nativedialerurl" and "ios_nativedialerurl", for Android and respectively iOS devices.
Default is true.

android_nativedialerurl
(string)
Android native softphone download url. If empty, then Mizutech's softphone will be offered:
https://play.google.com/store/apps/details?id=com.mizuvoip.mizudroid.app
Default is empty.

ios_nativedialerurl
(string)
iOS native softphone download url. If empty, then Mizutech's softphone will be offered: https://itunes.apple.com/us/app/mizuphone/id483685573?mt=8
Default is empty.

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.
Default is empty.

Extra settings

The java and the native plugins implements extra features. You can use any parameters supported by our JVoIP SDK as listed in the documentation:
http://www.mizu-voip.com/Software/JVoIP_Doc.pdf

User interface related

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

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.
Do not set to 0 because that will disable also the important notifications presented for the users.
Default is 1.

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

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.

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.

brandname
(string)

Brand name of the softphone.
Default is empty.

companyname
(string)
Your company name.
Default is empty.

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

footer
(string)
Footer text displayed for users on top of sofrphone windows.
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.

version
(string)
Version number displayed for users.
Default is empty.

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

advertisement
(string)
Advertisement url, displayed on bottom of the softphone windows.
Default is empty.

supportmail
(string)
Company support email address.
Default is empty.

supporturl
(string)
Company support url.
Default is empty.

newuser
(string)
New user registration http request OR link.
Default is empty.

homepage
(string)
Company home page link.

Default is empty.

accounturi
(string)
Company user account page link.
Default is empty.

recharge
(string)
Recharge http request (pin code must be sent) or link.
Default is empty.

p2p
(string)
Phone to phone http request or link.
Default is empty.

callback
(string)
Callback http request or link.
Default is empty.

sms
(string)
SMS http request.
Default is empty.

creditrequest
(string)
Balance http request, result displayed to user.
Default is empty.

ratingrequest
(string)
Rating http request, result displayed for user on call page.
Default is empty.

helpurl
(string)
Company help link.
Default is empty.

licenseurl
(string)
License agreement link.
Default is empty.

extramenuurl
(string)
Link specifying custom menu entry. Will be added to main page (dialpad) menu.
Default is empty.
extramenutxt
(string)
Title of custom menu entry. Will be added to main page (dialpad) menu.
Default is empty.

showsynccontactsmenu

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

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

log_mizu_email
(string)
The email account where the log messages will be sent by default.
Default: support@mizu-voip.com

Custom HTTP requests and links

You might integrate the webphone with your VoIP server HTTP API (if any).
Parameters can be treated as API requests or links. 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.
Parameters that can be API http requests are: newuser, recharge, p2p, callback, sms, creditrequest, ratingrequest.

Example credit http request:

http://domain.com/balance/?user= USERNAME

List of keywords that will be replaced automatically by the webphone:

USERNAME
PASSWORD
PINCODE
PHONE1
PHONE2
CALLEDNUMBER
CALLBACKNR
TEXT
MD5WEB: md5("C8y5:" + pUser + ":" + pPassword+":"+randomSalt);
MD5SIMPLE: md5 (pUser + ":" + pPassword);
MD5NORMAL: md5 (pUser + ":" + pPassword+":"+randomSalt);
MD5SALT: random salt
MD5CHECKSUM: md5("rD58s:" + pUser + ":" + pPassword+":"+randomSalt);
MD5VALUE: md5("ck5Gp" + pUser + pPassword + serversalt + randomSalt); serversalt can be empty string
SERVERAPIKEY: a server side specified authentication key

Parameter security

The following methods can be used to secure the webphone usage:
-don’t hardcode the password if possible (let the users to enter it) or if you must hardcode it then use encryption
-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)
-always pass the password parameter encrypted if you have to
-instead of password, use the MD5 and the realm parameters if possible (and this can also passed in encrypted format for the highest security)
-instead of applet parameters you can also use the javascript api (API_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.

JavaScript API

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.
Simple example:

```
<head>
        //include the webphone_api.js to your webpage and set the issdk parameter to true
        <script src="webphone_api.js"></script>
        <script>webphone_api.parameters.issdk = true;</script>
</head>
<body>
        //set parameters (alternatively these can be also preset in your html)
        webphone_api.setparameter('serveraddress', serveraddress);
        webphone_api.setparameter('username', username);
        webphone_api.setparameter('password', password);
        //see the “Parameters” section above 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)
        webphone_api.call(number);

        //hangup
        webphone_api.hangup();

        //send instant message
        webphone_api.sendchat(number, message)
</body>
```

getEvents (callback)

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.
   // You can find more detailed explanation about events in the documentation

   // example for detecting incoming and outgoing calls:

   var evtarray = 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.

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/sipusername/sippassword 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.

call (number)

Initiate call to a number, sip username or SIP URI.

hangup ()

Disconnect current call(s).

accept ()

Connect incoming call.

reject ()

Disconnect incoming call.

conference (number)

Add people to conference.
If number is empty than will mix the currently running calls (if there is more than one call)
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.

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.

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 (mute, direction)

Mute current call. The direction can have the following values:
0: mute in and out
1: mute out (speakers)
2: mute in (microphone)
3: mute in and out (same as 0)

hold (hold)

Hold current call. This will issue an UPDATE or a reINVITE.
Set the second parameter to true for hold and false to reload.

sendchat (number, msg)

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

audiodevice()

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

setvolume(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

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.

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).
Currently available only for Java and Service plugin.

getline ()

Return type: int
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 Service plugin.

isregistered ()

Return type: boolean
Return true if the webphone is registered ("connected") to the SIP server.

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(statustring)

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

isencrypted()

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

delsettings()

Delete all stored data (from cookie and localstorage): settings, contacts, callhistory, messages.

getenginename()

Returns the currently used engine name as string: "java", "webrtc", "service", "app", "flash", "p2p", "natividial".
Can return empty string if engine selection is in progress.

jvoip(name, arguments)

If engine is Java or Service plugin, then you can access the full java API as described in the JVoIP SDK documentation.
name=name of the function, arguments=array of arguments passed to the called function
ex: API function: API_Call(number) can be called like this: webphone_api.jvoip('API_Call', [number]);

See the webphone_api.js for more details.

Notifications

Use the getEvents(callback) to receive notifications and events from the webphone about its state machine, calls statutes and important events.
You will have to parse the received strings from your java script code. The 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):**

- 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

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 (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

Version 0.2 (February 12, 2015)

-internal beta version with basic skin and basic call functionality

Version 0.6 (July 3, 2015)

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

Version 0.9 (September 9, 2015)

-call-divert functionalities (mute, hold, forward, transfer, conference)
-native plugin
-examples and documentation
-better skinning
-better OS/browser handling
-automatic engine selection
-WebRTC stable
-upgrade to Java Applet v.5.6
-JavaScript API improvements

Version 0.98 (October 9, 2015)

-new: flash engine
-60+ bug fixes and improvements
-WebRTC improvements
-final stable API, modules and file structure

Planned for Version 1.0 (~October, 2015)

-chat
-file transfer
-app engine
-secondary engines (p2p, native dial, callback)
-custom builds based on customer settings
-MAC and iOS related improvements
-30+ minor improvements are planned
-improvements and bug fixes based on v.0.98 users feedback

Planned for Version 1.1 (~November, 2015)

-presence
-Android and iOS App and service engine
-more WebRTC features
-others

FAQ

How to get my own webphone?

1. Download and try the demo from http://www.mizu-voip.com/Portals/0/Files/webphone.zip
   The pricing can be found at http://www.mizu-voip.com/Support/Webphonepricing.aspx
2. Contact Mizutech at webphone@mizu-voip.com with 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)
3. Mizutech support will send your own webphone build within one workday on your payment.
   The payment options (paypal, credit card, wire transfer) can be found here: http://www.mizu-voip.com/Company/Payments.aspx

For the old “websipphone” 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: http://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 free basic (email based) support to all our customers. Maintenance upgrades are also available regarding to your license plan.
For “gold partners” we offer priority, phone and 24/7 emergency support.
Email to webphone@mizu-voip.com with any issue you might have.
Guaranteed supports hours depend on the purchased license plan and are included in the price.
Once your support period expires, it can be increased by 2 years for around $600 (This is optional. There is no need for any support plan to operate your webphone).

Direct support is provided for the common features (voice calls, chat, dtmf, transfer, 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).

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

You will receive the followings:

- the webphone files itself (engines, javascript API, html/css skins)
- latest documentations and 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, please contact us at webphone@mizu-voip.com. Please contact us only with webphone related or VoIP specific projects.

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

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)

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.

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 and gateway software are available also for free.
If you have any incompatibility problem, please contact webphone@mizu-voip.com with a problem description and a detailed log (loglevel set to 4). 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
- 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
- Social networking websites , facebook phone
- 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
- 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
- Java Script phone
- 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
- 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 "mizulayout.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?

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's 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>webphone_api.parameters.issdk = true;</script>
<script type="text/JavaScript" src="oldapi_support.js"></script>
<script src="js/jquery-1.8.3.min.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.

Please note that this 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, connecting to your VoIP server directly (Java, Service and App), via WebRTC or via Flash so you will have a solution fully owned/controlled/hosted by you.

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_java 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) or you might just need to add a module for your server to enable it, 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. Setup a WebRTC to SIP gateway: There are many free software which is capable to do this task for you, including Asterisk and Dubango.
4. Use the fee 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)
5. Use the Mizutech WebRTC paid service. We are going to provide dedicated WebRTC gateways for companies with such needs

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 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 enduser bypassing your server. This is up to your server settings and NAT handling features)

What kind of licensing options are available?

We have separate pricing options to fulfill all of our customers' needs and budget: http://www.mizu-voip.com/Support/Webphonepricing.aspx

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.

ERROR and WARNING messages in the log

If you set the applet 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.

The webphone is not loading/starting

Make sure that either Java or WebRTC is available in your browser or you can use the native plugin (Android and Windows) or app (Android and iOS).

Failed outgoing calls

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 applet parameters set to 2 or 3 (where xxx is the name of the codec: use_pcma=2, usgsm=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)

Calls are disconnecting

If the calls are disconnecting after a few second, then try to set the "invrecordroute" applet 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 applet parameter (you need to call the API_HTTPKeepAlive function periodically from a javascript timer)

Limitations

-A few features are not implemented yet and planned for v.1.0 (~October, 2015), including better chat, better mobile support and a lot of other improvements
-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
-Not all the listed features are available from all engines (the webphone automatically handle these differences internally)

OS/browser related issues

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" applet 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.

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 file. Navigate to "Safari -> Preferences -> Security (tab) -> Internet plug-ins (Manage Website Settings)". Find the site in question and modify the dropdown to "Run In Unsafe Mode". You will be asked to accept the site's certificates. Alternatively, simply use the latest version of the FireFox browser.

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.

Java in latest Chrome is not supported anymore (the webphone will select WebRTC by default).
If you 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

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” applet 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)

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” applet parameter to at least “5” than the important rtp and media related events are also stored in the logs.

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.

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_rport 9
-use_ fast_stun: 0
-optionally you can also set the “udpconnect” applet 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 records to implement failover or load balancing.

I have call quality issues

Call quality is influenced primarily by the followings:

- The engine used (Java and native plugin 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 use_xxx parameters where xxx have to be replaced with the codec name)
- 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)
- wireshark log (check missing or duplicated packets)

I have one way audio

1. Review your server NAT related settings
2. Set the "setfinalcodec" applet parameter to 0 (especially if you are using Asterisk or OpenSIPS)
3. 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.
4. If you are using MizuVoIP server, set the RTP routing to "always" for the user(s)
5. Make sure that you have enabled all codec's
6. Make a test call with only one codec enabled (this will solve codec negotiation issues if any)
7. Try the changes from the next section (Audio device cannot be opened)
8. 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" applet 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 applet 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

You can verify this by using this applet.

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 applet 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" applet parameter to 0 and check if this will solve the problem).

If these doesn't help, you might set the "cancloseaudioline" applet 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 applet parameter that can be used in this situation:

- set the "changesptoring" applet parameter to 3
- set the "natopenpackets" applet parameter to 10
- set the "earlymedia" applet parameter to 3
- change the use_fast_stun applet 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).

Subsequent chat messages are not sent reliably

Set the “separatechatdiag” applet 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” applet 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_stun parameter 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 applet loglevel parameter to 5) and also from the caller (your server or remote SIP client)

What is the best codec?

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.
This depends mainly on the circumstances.
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) are 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 number display 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 applet parameters to influence the caller id display at the remote end:

- -username
- -authusername/sipusername
- -displayname

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).

For incoming calls the webphone will use the caller username, name or display name to display the Caller ID. (SIP From and Contact fields).
You can also use headers such as preferred-identity to control the Caller ID display.

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. Browsers can cache applets so there is a change that you are still using the old version.
Refresh the browser cache by pressing F5 or Ctrl+F5. Make sure that you don’t have some caching proxy on the path.

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

The demo examples are not working on my PC

This issue can happen in some circumstances only if you launch the applet from local file for testing (not from a web server).
Make sure that the file path is not too large for java. Copy the example directory to the C:\ directory and try again.

How to use via URL parameters?

The webphone can load its settings also from the webpage URL and perform various actions such as initiate a call. Example:
http://www.yourwebsite.com/webphonedir/webphone.htm?serveraddress=sipserverdomain.com&username=USERNAME&password=PASSWORD&haveloginpage=false& callto=CALLEDNUMBER&autocall=true
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.

Resources

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