Mizu VoIP Server API

Overview

This document describes the Mizutech VoIP server public API.
The mizu softswitch API allows a client program (such as a browser, a web service or a SIP client) to connect to a server instance and perform various requests.

Primarily you will interact with the server using standard VoIP protocols such as SIP, H.323 or WebRTC. However there are a lot of functionalities which are not handled by these standard protocols, usually administration related. You can use the VoIP server API described in this document to cover such needs.

The API can be used by endusers or to implement various functionalities from:

- Any web application (from client side JavaScript using the HTTP transport or server side script such as PHP or .NET)
- Any mobile or desktop application
- Manually (for example via telnet, curl or browser)
- Test from your browser (just copy the API request to your browser URL input box and hit the Enter button)
- MManage -> Server Console form (all API commands can be sent also here, but without the authentication parameters as all commands from here will be authenticated as admin requests)

Quick start

You can generate valid API request examples for your server from MManage -> Config menu -> Client Config -> Generate API examples menu.
Copy-paste some of them into your browser URI box and experiment with them. Then search the API chapter for the API which covers your needs.

Examples

The examples below are for HTTP GET with URL parameters (however you can also use JSON, XML and other formats). Change domain.com to your server address:port and use valid values for authentication and for the parameters. In these examples we are using authkey + authid/authmd5 based authentication, however you might use others as described below in the "Authentication" section. KEY means the apiv2key global config setting.

Note: You can generate working examples for your server from MManage -> Config menu -> Client Config -> Generate API examples menu.

API test (this is the simplest example without authentication requirements):
http://domain.com/mvapireq/?apientry=apitest1

Request user credit if set as public with no apiv2key set (not recommended in production):
http://domain.com/mvapireq/?apientry=balance&authid=USRNAME

Request user credit with password in clear text (not recommended in production):
http://domain.com/mvapireq/?apientry=balance&authkey=KEY&authid=USRNAME&authpwd=PASSWORD

Request user credit with proper authentication (recommended):
http://domain.com/mvapireq/?apientry=balance&authkey=KEY&authid=USRNAME&authmd5=MD5

Initiate callback:
http://domain.com/mvapireq/?apientry=callback&authkey=KEY&authid=USRNAME&authmd5=MD5&anum=PHONE1&ivr=-1

Initiate phone to phone call:
http://domain.com/mvapireq/?apientry=p2p&authkey=KEY&authid=USRNAME&authmd5=MD5&anum=PHONE1&bnum=PHONE2

Sending SMS message:
http://domain.com/mvapireq/?apientry=sms&authkey=KEY&authid=USRNAME&authmd5=MD5&anum=PHONE1&bnum=PHONE2&txt=TEXT

Request rate to destination:
http://domain.com/mvapireq/?apientry=rating&authkey=KEY&authid=USRNAME&authmd5=MD5&bnum=PREFIX_OR_NUMBER

Recharge with PIN code:
http://domain.com/mvapireq/?apientry=charge&authkey=KEY&authid=USRNAME&authmd5=MD5&anum=PINCODE&now

Add credit: (from trusted IP only)
http://domain.com/mvapireq/?apientry=addcredit&authkey=KEY&authid=USRNAME&authmd5=MD5&pin=USERNAME&credit=AMMOUNT

With raw TCP or UDP you can send in the following format (terminated with a new line if TCP):
/mvapireq/?apientry=apitest&authkey=KEY&authid=USRNAME&authmd5=MD5

With websocket, you have to use the uri with the "mvstwebsock" parameter such as:
ws://yourserveraddress/mvstwebsock/
Then send the request within the websocket stream as it would be TCP (the rest of the request line).

Protocol

The API is listening by default on **port 80** for both UDP and TCP/HTTP/websocket (can be changed with the "mainaport" global config option or disabled by setting the "usemainaport" to false). Secure access is also available if you have set a domain and SSL certificate. More details can be found here.

The API can accept commands and send answers in various format defined by the request or the "format" parameter.

Transport protocols

Usually the API is accessed via HTTP or HTTPS however it also supports other methods which might be sometimes preferable for performance, accessibility or other reasons.

The following transport protocols can be used:

- UDP (up to 1490 bytes packet size)
- TCP (requests must be ended by CR/LF)
- TLS (if enabled)
- HTTP (1.0 or 1.1)
- HTTPS (if TLS is enabled)
- websocket (URI must contain "mvstwebsock". Then send the request in websocket as you would send in HTTP URI)
- secure websocket (on port 443 if TLS is enabled)
- Tunnel (if encryption/tunneling is enabled)
- SIP (via special header if enabled)
- SMS (some commands are available also through SMS: p2p, cb, balance, rating, charge, cli. The SMS text must contain the parameters)

Request/response format

A simple key/value protocol is utilized for the request with the following parts:

- the “mvapireq” entry point string has to be used with all requests. A fix string as “mvapireq”.
- apientry (name of the API function such as “sms”, “balance”). See the “API” section below.
- authentication parameters (“authid” and “authpwd” or “authmd5”/”authsalt”). See the “Authentication” section below.
- function parameters (such as “anum”, “bnum”, “txt”, “param1” , “param2”). See the “Parameters” section below.

More details can be found here.

Supported data **format**s:

- plain text
- URL parameters (usually with HTTP GET: http://SERVER/mvapireq/?apientry=function&parameter1=value1)
- parameters in http body (usually with HTTP POST)
- HTML form (submit)
- Ini format (key=value)
- XML
- SOAP (SOAP+XML)
- RDF
- JSON
- JSONP (JSON forward) works also for remote services if the “enablejsonp” global config option is set to 2 (http://SERVERADDRESS/mvapireq/jppget?url=...)

For the **response** a code and/or a text is sent:

- code: 200 means success, 499 means failure
- text: all answers are suffixed with either “ERROR” or “OK”

By default the answer format is guessed from the request, unless it is specified by the “format” parameter which can be set to one of the followings (string in lower case):

- text
- cleartext
- html
- xml
- soap
- json
- jsonp (will respond with javascript callback code)

If the httpstrictresponse is set to 1, then the response code is sent as html answer code (otherwise you will receive HTTP 200 OK for all requests and the real API response code will be in the message body).
You can also use the old API over this new (all functions in the old API works also via this new API with mapped parameters).

Shared service port

All API and Web related services on the Mizu server can be accessed via a single unified port. This case be set in the MManage configuration wizard (Network page -> Access port) or via the mainaport global config option. Default value is 80 (the standard HTTP port).
You can access the server via its IP address or domain name (if a domain name or sub domain have been set).
You can use various transport methods for communication, including raw TCP, HTTP and WebSocket. UDP can be also used on the same port by default or via another port set by the mainaportudp global config value.
Most of the services can be also accessed via a secure port using the TLS protocol if you have configured a domain and SSL certificate. This can be set on the config wizard as the “Secure Port” or in the global config with the following sslportmain and sslportweb values. Default value is 443 (the standard HTTPS port).
Note: SIPS (TLS signaling) is listening on a separate socket and it can be set with the localtlsport parameter, defaulting to 5061.

The server will auto-detect the service to be used from the request (transport, protocol, headers, URI parameters and other heuristics) and the following services are accessible:

- API (URL containing the /mvapireq path. For example: http://myserver.com/mvapireq/?apientry=balance&authid=testuser). With WebSocket the /mvstwebsock path must be set and send the request parameters in the packet body.
- Enduser web portal (URL containing the /webvoip or /webvoipportal path. For example: http://myserver.com/webvoip)
- TURN and STUN (auto detected if the packet is a valid TURN or STUN request)
- SIP (auto detected if the packet is a valid SIP signaling message)
- WebSocket SIP signaling (URL containing the /mfstwebsock path)
- MMQ services (URL containing the /mmstwebsock path. Rarely used.)
- Built-in fast web service for static files (URL containing the /mvweb path)
- HTTPS -> HTTP forward (URL containing the hsforward parameter. This is a special service which is not enabled by default)
- Encrypted tunneling (if none of the above, it will be checked against some heuristics and if pass then it will treat as encrypted packets forwarded to tunnel decoder)

Note: Some of the services (such as the webportal or the WebRTC signaling) are running in a separate process and can be accessed also directly via their internal listener port

Security

Before execution, the server will authorize and authenticate each API request as described in the "Authentication" section below. Each API has a usage limit (rate-limit) to avoid overusing (DoS or brute-force) including for attacks tied to individual users, IP's and global usage.

Authentication

Each API function requires an **authentication** which is a combination of the followings:

- ip authentication: if client ip must be in the allowed list defined by the "apiv2ipauth" global config
- api key: usually a "authkey" parameter have to be sent which have to match the "apiv2key" global config option (old "httpapikey")
- user authentication: a valid username/password combination (or pin/did). For the password you can use the md5 hash and this is strongly recommended instead of the clean text password. TLS/HTTPS can be also used for better security.
- other optional validations (such as device registered state, enduser credit or enforcing nonce authentication)

The default API can be divided in the following **categories**:

- Sensitive API: IP and admin authentication is recommended for these. For example "newuseri"
- Admin API: for server administration, such as "info" to request server status details
- User API: API for users (such as "sms" to send SMS or "recharge" to add credit) by default with api key and username + password or md5 authentication.
- Public API: for example "status" to request a user online status (for these you might allow username based authentication)

The default API's are set with optimal authentication (sensitive API's requiring admin or ip access, local and same LAN IP's are trusted which you can change by adding them to tb_api and set custom authentication), however for custom API's (created by you) make sure that you set the proper authentication depending on the API sensitivity.

For example some API might request IP+apikey+user authentication, while there are some API's which you can set do require no authentication at all (for example you can set the "status" API to require only a valid username). Usually (and by default) each api require apikey and user authentication (some API's doesn't require these from trusted sources). For admin requests an admin user name/password has to be used (unless you change it).

For bulk operations you might set the "trustediplist" global config option to your IP. For example your web server IP. Multiple IP can be separated by comma.

The following **parameters** are defined:

- **authkey**: Defined by the "apiv2key" global config option (optional)
- **authid**: Account username (but can represent also the user id, name, pincode or did)

- **authpwd**: Account password in clear text (but can represent also the pin). This should be used only on trusted links such as on the localhost. Otherwise better to use the authmd5 parameter.
- **authsalt1**: Need to be sent if the authmd5 (below) is used (a short random number or string)
- **authsalt2**: Need to be sent if the authmd5 (below) and the nonce is used (a short random number or string)
- **nonce**: Optional. The first 16 character of the server supplied nonce if nonce auth was enabled
- **authmd5**: authv3_ + MD5(MD5(tU29m + authid + authpwd + authsalt1) + authsalt2 + serversalt + nonce)
    - MD5: message-digest algorithm
    - authv3_: hardcoded prefix string "authv3_"
    - tU29m: hardcoded string "tU29m"
    - authid: account user name
    - authpwd: account password
    - serversalt: salt defined by the "apiv2md5salt" global config option (optional, depending if configured on the server)
    - authsalt1: client side generated salt as sent by the "authsalt1" parameter (optional but highly recommended)
    - authsalt2: client side generated salt as sent by the "authsalt2" parameter (recommended if nonce is used)
    - nonce: only if nonce authentication was set ("preauth" was used)

The above string parameters just have to be concatenated (no colon or space between them).

The simplified formula for authmd5 without nonce looks like this:
MD5(ck5Gp + authid + authpwd + serversalt + authsalt1)

- MD5: message-digest algorithm
- ck5Gp: hardcoded string "ck5Gp"
- authid: account user name
- authpwd: account password
- serversalt: salt defined by the "apiv2md5salt" global config option (optional, depending if configured on the server)
- authsalt1: client side generated salt as sent by the "authsalt1" parameter (optional but highly recommended)

The authentication can be fine-tuned in the **tb_api** table. You can specify settings for existing API's (for example you can force the balance requests to be always authenticated) and you can also create new entries with custom SQL.
The following fields are defined:

- fname: function name (api entry such as "sms", "adduser" or "mynewcustomapi")
- req_key: 1 if authkey is required, otherwise 0 (default is 1)
- req_ip: 1 if ip is checked, otherwise 0. If 1, then the allowed IP list can be defined by the "apiv2ipauth" global config option (default is empty which means only local ip). Local IP is trusted by default. Private/LAN IP is also trusted by default unless you set the trustlanip global config to false.
- req_user: user authentication: 0=default,2=disabled,4=srvadmin,6=admins,8=owners and resellers, 10=support,11=SIP authenticated users,12=all,14=public (default is 6 for admin API and 12 for user API)
- req_user_mode: defines how to authenticate the user. 0=default, 2=user + pwd or pin (def), 4=user or pwd or pin (less), 6=username/pwd (most). The default is defined by the "apiv2auth" global config option. Will also check the authentication setting of the user (so it might result in username only or IP based authentication)
- req_user_credit: allow API access only if user has credit. 0 means default auto, 1 means no check, 2 means minimum credit defined by mincreditonroute, other means min credit value to check
- req_user_register: allow API access only if user is currently registered. null means def loaded from apidefneedregister, 0 means no (don't check registrar state), 1 means yes (user must be registered)
- req_user_login: allow API access only after login procedure. null means def loaded from apidefneedlogin, 0 means login is not required, 1 means yes (login required)
- customsql: custom api based on an SQL query (in case if you set this for a predefined API, then the predefined behavior will be lost and the API will use your custom SQL). You can write any query supported by the SQL engine, including stored procedure usage if needed
- custom_action: define a custom action for this API: 1=email, 2=sms, 3=call ring, 4=restart,5=mssqlrestart,6=reboot,7=run
- custom_action_body: body text for the above custom_action (such as SQL query or email body)
- enabled: set to 0 to disable the API or 1 (default) to enable

Example:
If you wish to set an existing API as public (with no authentication request) just create a new record in tb_api with api entry string for fname, 14 for req_user and 4 for req_user_mode. The fname field must set to the exact name of the existing API entry (for example "balance", "rating", "sms" or others) if you wish to overwrite the authentication for an existing API (Otherwise a new API entry will be created).
For example If you wish to make the balance request API fully public, then just add a new record into tb_api with the fname set to "balance" and the req_key to 0 and the req_user set to 14 (which means public). Once this is set the balance can be requested as simple as http://serveraddress.com/mvapireq/?apientry=balance&authid=USERNAME

Note: for the changes in the tb_api to be applied you must issue a "apireload" command (from the Server Console) or restart your server.

Hardened access

The above described parameters already provide good enough access restrictions for the API for normal use case, including access over the internet. By applying the following techniques described in this chapter, you can add even more security for your server API access, if required for your use case.

Global config options:

- **apiv2md5salt**: always set to a random value
- **apidefneedregister** (or **tb_api.req_user_register**): access will be granted to user API only for those user who are already authenticated over SIP or WebRTC. Possible values: 0: no (default), 1: for users, 2: check also the IP
- **allowtrustedip**: You might set to to avoid ip spoofing. Possible values: 0=no,1=my ip only,2=private ip's only,3=yes(default),4=everywhere(including admin console)
- **apitrustediponly**. Possible values: 0=no (default),1=allow api requests from trusted ip only. If this is set to 1, then users will not be access any API over the internet (not suitable for public services)
- **apideftrustadmin**. Specify how to recognize admin users. Possible values: 0: no, 1: admin if trusted ip (default), 2: always ip only, 3: always auth only, 4: always ip+auth
- **apialtauth**. Specify whether to allow also alternative authentication //-1: auto guess, 0= strict username/password only ,1=enable all mode (also clear text and pin)
- **apitrustfromconsole**. Specify whether to trust the admin console access. Possible values: 0=no,1=yes,2=as admin
- **apidefneedlogin** (or **tb_api.req_user_login**): Force nonce validated authentication. Possible values: 0: no, 1: require from not trusted sources, 2: yes, require, 3: yes, require with ip address verification
- **remoteadmin.** Specify whether to allow remote administration. Possible values: 0: disable, 1: from localhost/127.0.0.1, 2: from same machine, 3: from trusted sources, 4: from local LAN and subnet, 5: from everywhere. Default is 5.

Note: Some parameters are extra filtered by default for SQL injection and XSS attack preventions (the database operations performed by the mizu voip server are always parameterized to avoid SQL injections, however these extra validations might help for mitigating issues in third party application that you might use)

For additional security you should use the API over encrypted HTTPS (not over unsecure HTTP).
Optionally as a simple encryption/obfuscation, you can also encrypt the requests with XOR + Base64 encoding. The XOR key can be set by the httpapiencryptkey global setting. Replace the '=' char with '-',the '+' char with '_' and the '/' char with '.' in the resulting string if any. *An online test tool for this can be found here (select the XOR Base64 option from the list).*

- If you wish to encrypt the whole POST URI, then encrypt as described above and pass it as "mcrfs" parameter.
- If you wish to encrypt the parameters values, then encrypt as described above and prefix them with "oenc1_" string.

Using nonce for replay attach protection

Basic protection for replay attach is already enabled by default with the fix server salt (defined by apiv2md5salt) and client salt values (sent by the client with the authsalt parameter). Also by default the server will limit the API access after a few subsequent failed authentication requests.
For dynamic nonce verification, you must enable it with the **apidefneedlogin** global config option (set to 1 or 2) or individually for API entries by the **tb_api.req_user_login** field value.
When this is enabled, the client will always need to send a "**preauth**" API request first for which the server returns a nonce value ("OK: new nonce is:xxxEOF").

The client then must use this nonce to calculate the authmd5 checksum (in addition to already mentioned field, it must concatenate also the nonce at the end) and also must send back the first 10 characters of the nonce with every request as the nonce parameter.
If the apidefneedlogin global config or the tb_api.req_user_login field is set to 2, then the client must be prepared to receive a "ERROR: new nonce is:xxxEOF" answer for any API request and in this case it must repeat the last request with the new nonce sent by the server (This is sent if the IP address of the client was changed or if there was no preauth request)

Parameters

Parameters can be URI encoded. By default the basic ASCII character set are used.
Parameters can be passed by the parameter names (as listed in the API section below []) or using param1, param2 … paramn.
Common parameters are the followings:

- authentication parameters: described below in the "Authentication" section.
- param1,param2,param3: this format can be also usually used. It must be in the same order as documented in the "API" section below. If you specify the parameters in this way, then the parameter order must be: functionname, apikey, username, password, function parameters (anum, bnum, text)
- anum/phone1/from: for first/from/originating phone number
- bnum/phone2/to: for second/to/target phone number
- txt/message: body for sms, email and others
- targetuser: admins can impersonate any other user by using this parameter
- now: you might append a new=x parameter at the end of the URL query list for API usage tracking
- others: as listed in the "API" section below. Possible parameters are listed in brackets [].

Keywords

The following keywords are automatically replaced by the Mizu VoIP client apps, so it is possible to set some API's (such as balance request and others) as SIP client configuration parameters which are then handled dynamically, at run-time:

- KEY: api key
- USERNAME: sip username
- PWD: sip password in clear text (should be used only with https)
- MD5CHECKSUM: mizutech old api: MD5("rD58s:" + username + ":" + pwd+":"+additional)
- MD5VALUE: mizutech new api: MD5("ck5Gp" + username + pwd + apiv2md5salt + usersalt) //the usersalt is optional
- MD5WEB: mizutech web MD5("C8y5:" + username + ":" + pwd+":"+additional);
- MD5NORMAL: MD5(username + pwd + salt)
- MD5SIMPLE: MD5(username + pwd)
- SALT: salt for md5 calculation
- PHONE1, PHONE2, PREFIX_OR_NUMBER, CALLEDNUMBER: caller/called numbers
- PINCODE: pin code
- AMMOUNT: credit

API

The supported commands (API entries / functions) are listed below.
Parameters in []
Short description in {}
The best way to learn the usage is to try out the requests quickly from your browser. Some valid examples can be generated from MMAnage -> Config menu -> Client config -> Generate API examples. For example try the balance request example first once that works, just replace the apientry parameter to any of the below API entries and add other URI parameters as needed.

User commands

Commands available for users (authenticated as enduser/reseller/other user type):

newuseru {public add user} [u_username,u_password,u_name,u_email,u_phone,u_currency,u_country,u_address]
balance {credit request}

callrating {call rating for an ongoing call returning credit, maxduration and rating for existing call} [username, callid]
rating [bnum=callednumber/prefix] { rate request}
status {self online status. 0=doesn't exists,1=offline,2=online,3=calling,4=speaking}
verifyuser [userusername] {request the status of this user. 0=doesn't exists,1=offline,2=online,3=calling,4=speaking}
presence_get [userusername/userlist,ptype,domain] { will return the presence status of one ore multiple users; ptype have to be set to 1 if called from timer and 0 when at start; answer: OK: puserlist: user1:status1,user2:status2 OR ERROR: errordescription }
presence_set [pstatus,ptype] {set presence status; pstatus means status string, ptype have to be set 0 when called at start, otherwise 1; answer: OK: xxx or ERROR: xxx (doesn't matter since we have nothing to do with it). Also offline messages: XOFFMESSAGE:fromuser:textXOFFMESSAGE:fromuser2:text2XOFFMESSAGE:}
sendim [bnum/buserlist,txt] {send chat message}
getconfroom {get/create conference room}
callback [num] {initiate callback call}
p2p [bnum, cnum, waitfor] {phone to phone call request. you can set the waitfor to 1 or 2 to wait for the call connect or failure}
sendfax [from,to,filename]
sendmail [from, to, subject, message] {from can be empty}
sms [from,to,txt] {send sms message}
sendsms [from,to,message] {send sms message asynchronously -fast}
sendsmssync [from,to,message] {send sms message synchronously –wait for delivery answer}
sendsmsunicode [from,to,message] {send sms message asynchronously -fast }
sendsmsunicodesync [from,to,message] {send sms message synchronously -wait for delivery answer}
cli {add pin less number (ANI)}
charge [pin] {recharge using recharge PIN}
cccharge [cardnum,expireyear,expmonth,ccv2,ammount]
transfercredit [username,credit] (user A can send credit to user B)
lockphonenumber {reserve a phone number}
getphonenumber {return a phone number from the free pool)
search {user search}
callbackaccessnumber {return the callbackaccessnumber}
addcallbacknumber {add custom callback number}
changepwd [newpassword,oldpassword]
forgotpasswordquestion [email]
forgotpasswordanswer [email, forgotpasswordanswer]
addcredit [pin] {add credit (Only from trusted IP by default)}
callforward [anum,bnum,cnum]
sendccinfo
voicerecdownload [cdrid,callid,userid,parentuserid,maxrecords,days,format,fromdt,todt,caller,called] {download recorded voice}
The format parameter can have the following values: 1: mp3, 2: wave, 3: mp3+zip, 4: wave+zip.
All other fields are used for CDR filtering (one or more files can be downloaded at once)
getdetails [mode=min/normal/max]
setfield [fieldtype=string/int/float, fieldname, fieldvalue,rowid, sync] {update a field in tb_users}
the following fields are allowed:
forwardonbusy,forwardonnoanswer,forwardalways,voicemail,musiconhold,displayname,noanswertimeout,forwardearlystart,changesptoring,convertdtmf,playadv,playdisc,sendsmsalert,sendemailalert,sendsmsreport,senddailyemail,sendlowcreditemail,sendmonthlyemail,sendotheremail,notifymissedsms,notifymissedemail,choosecodecs,needcodecconversion,discvoice,connectvoice,sendlowecreditemail,voicemailonbusy,voicemailonnoansw,voicemailalways
the following fields are allowed only if not set yet:
contactname,name,email,email2,phone,fax,address,billaddress,postaddress,country,regnumber,euregnumber,birthday,gender,language,utc
(These restrictions are applied only for user logins. Admins can change any fields in any table)
cdr [caller,called,maxcount=record count, days =today/lastweek/lastmonth/daysnumber dir=all/in/out type =all/connected/notconnected, prefix, fields=min/normal/all, summary=no/yes, by= all/day/caller/called, order= date,caller,called,user]

Admin commands

Commands available for administrators (authenticated as admin user and/or trusted IP address based authentication).

Diagnostics

version {display software version number}
hearthbeat [mode=quick/full] {request server heartbeat}
info {show server status and important parameters}
regstat {registrar statistics}
reguser [usr] {show registered users/devices}
fastregusers [succ/all] {list fastreg module users}
endpoints {list sip endpoint info}
epreg [regtype, regstate] {list registrar endpoints}
routingtest [ip,caller,called] {will return the destination route as it would be for real call}
gencdr { generate cdr's: www.mizu-voip.com/portals/0/files/gencdr.pdf }
routingtest
deviceid {show device id}
fastusers {list cached users}
quickstat {quick statistics}
locks {list of pending loks}
threadlist {thread list}
threads {thread list}
threads2 {thread list}
querylist {list the guid for the current running queries}
fs {internal PBX}
lists
shortinfo
tunnelinfo
tunnelinfoex
tunnellist
call
showlog [main/sip/gk/sh/tcp,lines] {show the requested logfile}
log
file {will send the requested file}
logsearch
reports
showcfg {show the config files}

Commands

newuseri {private fast/unrestricted add user from trusted source admin}
  [u_username,u_password,u_name,u_email,u_phone,u_currency,u_country,u_address]
newuser {deprecated}
adduser {deprecated}
addserver {add sip server or traffic sender}
[u_type,u_name,u_username,u_password,u_domain,u_ip,u_port,u_proxy,u_transport,u_auth,u_email,u_country,u_currency,u_credit,u_routing,u_routingpriority,fieldnameX/fieldvalX]
deluser {delete a user by id or username} [u_id or u_username]
listusers {list users} [u_type, u_maxcount, u_extended, u_order]
cmd {exec windows shell command}
asendemail [from, to, subject, message. from can be empty]
smsreceive {for inbound SMS. See the SMS chapter in the admin guide for the details}
asendsms [from,to,message] {send sms message asynchronously for high performance}
asendsmssync [from,to,message] {send sms message and wait for the result}
sendamsg {SendAgentMessage: send message to agents (agentid-all,message) }
voicehere [dbcallid] {start realtime call listening}
voicehere2 [username] {start realtime call listening}
bannedlist {list of banned ip's}

delbanned [all] {remove ip from the banned list (ip)}
unreg [username,ip,mode] {unregister user or “all”. If mode if 1 then remove instead of unreg}
disccall [dbcallid]
gk {connect to h323 gatekeeper module}
client {switch to the requested client}
db {any database query (select,update,etc) only if authenticated user type is admin}
setip {set new ip (cardname,ip,mask,gateway,gwmetric)}
tbackup { launch a database backup to the second server}
timezoneset {set timezone}
putfile [filename, filedata] {will save a file under the server directory. The filedata should be base64 encoded }
getfile [filename] {will load a file from under the server directory }

Reset

reload {reload current configuration}
vsiprst {restart sip-h323 router}
sipreset {will reset the sip stack}
maintreset {will restart the maintanance thread}
ftprst
webrst
resetp
dbreset {reset database connections}
restart {what=service/server/database/force (server means OS restart!)}
gwrst { restart all gateways }
dbrsttgs
apireload {useful if you added/modified an entry in tb_api)
rulesreload
reloaddialplan
reloadapikeys
reloadnumdircache
logrename
rebill
reindexncache { reindex ccampclient }
calcncache { recalc cache size and mode }
clearnccache { clear next client cache [operatorid] }
upgradeall {selfupgrade all clients (0-vclientgw,1-vclientsrv,2-all)}
dbbackup {create database backup} [reseverd,path,ival,type]
dbrestore {restore database} [reserved,path]
dbfailover {force db failover} [reserved,forcedbnum,checkonly,withrestore,needrestart]

Various operations

help { show console command list; deprecated }
keepalive {keep alive the connection}
close/exit/quit {close current connection}
predectivet [start,stop,restart] {start-stop predective thread}
syncmedia {syncronize media files}
stopcanner {stop number scanner threads}
checkscanner {check number scanner threads}
backupdb
backup {BackupToOtherServer: launch a database backup to the second server}
finetune {FinetuneService}
normalizechk { check number normalization (called,caller,country) }
syncusers [oop] {syncroonize fast users list with the database}
delobj [ep/logic/sqlupd/router/pred/check/simall/maint/logger/dbobj/all] {free objects}

monthlymaint { run monthly maintanance tasks }
weeklymaint { run weekly maintanance tasks }
dailymaint { run daily maintanance tasks }
hourlymaint { run hourly maintanance tasks }
dailyreport { launch the daily report builder thread }
unittest [testtype,threads,testtime,parameter] {start unit tests}
savesettings {write uncommitted settings from db to file}
setini [section,key,value,file] {write to ini file}
getini [section,key,file]] {read from ini file}
saveinifiles {write uncommitted inikeys from db (module,file -or all, notsaved) }
loadinifiles {load ini to db (module,file -or all)}
toclient { send message to client service (clientip or username, message) }
togw { send message to gw (clientip or username, message) }
call { sim call each other (origsimid,telnum,duration) }
toip { send message to the specified (sims,simid,credit,callnum,dialnum) ip }
at { send direct message to the specified engine (gwname,line,ATcommand) }
updcdrdir
createcdrinfo
setfastregpwd
newuserevent
newusersevent
checkcredits {check charges and creditrequests need}
checkpins {will recheck pincodes}
capcheck { check free capacity (direction -3330630) }
parsegsmcdr { reparse cdrr from the specified logfile }
willstop { no more creditcheck }
checkmails {CheckEmailJobs: check email jobs}
db [txt] {any sql select/insert/update command}
setfield [table, fieldtype, fieldname, fieldvalue,rowid, sync] {update a field}
getfield [table, retfieldname, filterfieldtype, filterfieldname, filterfieldvalue] {request a field}
def values:
table: tb_users
retfieldname: id
filterfieldtype: auto guess
filterfieldname: auto guess (username/callerid/userid)
filterfieldvalue: mandatory
pputfile [filename, filedata] {will save a file under the mvweb\filestorage directory. The filedata should be base64 encoded }
pgetfile [filename] {will load a file from under the mvweb\filestorage directory }

Custom

You can create your custom API by adding rows to tb_api.
For this, just create a new record:

- set the name of the API call with the fname parameter (for example “myrequest”)
- set what it does with the customsql and/or custom_action/custom_action_body fields
- define access by the req_user and req_user_mode
- optionally add more restriction for the access with the req_ip, req_user_credit, req_user_register, req_user_login

The “customsql” field can contain any sql statement: for SELECT it will display the rows (max 100) and for UPDATE/INSERT it will execute the query.
Make sure to set the proper values for the authentication fields and set any name with the “fname” field.

You might prefix the sql with * to execute synchronously (otherwise the faster async will be used which can’t tell if the query have failed). The following keywords can be used:

- [USERID]: will be replaced with the currently logged in user id (tb_users.id)
- [USERNAME]: will be replaced with the currently logged in username
- [SPARAM1]: will be replaced with the supplied sparam1 URL parameter
- [SPARAM2]: will be replaced with the supplied sparam2 URL parameter
- …
- [SPARAM99]: will be replaced with the supplied sparam99 URL parameter

You can also use any user or input parameter enclosed with "{" and "}".
For example: {username}, {email}.
These will be replaced at runtime to their corresponding value.

Be aware of SQL injections. Don't pass these parameters from user input!

Actions

Upon API execution, you can define a specific extra action to be executed such as sending an email to admin when a specific API have been called.
This can be done by setting the custom_action and custom_action_body fields in tb_api.
The custom_action field can have one of the following values:
For successful API requests:

- 1: email
- 2: SMS
- 4: restart
- 6: reboot
- 7: command (execute arbitrary command line)
- 8: sql (execute arbitrary SQL)

For failed API requests:

- 51: email
- 52: SMS
- 54: restart
- 56: reboot
- 57: command (execute arbitrary command line)
- 58: sql (execute arbitrary SQL)

The custom_action_body is a string which can represent the action body depending on the custom_action field, such as email body, SMS message, SQL or command line.
The body can contain keywords enclosed in {} which will be replaced at runtime with the API input parameters or with the current authenticated user fields. For example {username}, {id} or others.
You can also insert a custom SQL like {select top 1 callednumber from tb_cdrs with(nolock) where callerid = USERID}.
(USERID is a special keyword which will be replaced with the currently authenticated user id: tb_users.id)
*The processing is similar to Scheduled Tasks so you can use the ***TO and ***SUBJECT also.*

Notes

Access Control and Custom API

The default access control offers a reasonable balance of usability and security. You can fine tune it with several global config settings and/or by overwriting individual API access from the tb_api and you can also add new functions here.

Basically with tb_api table you can do two things:

1. Change the authentication for an existing API as described here.
2. Create new API entries with custom action as described here.

CLI

By default all API's are available also via the server console (from MManage "Server Console" form).
The CLI is listening on port defined by the adminport global config option.
For login it will accept admin and support user credentials (users can be added from MManage -> Users and devices form).
You might IP filter the access with the remoteadmin global config option.
Possible values:

0: disable
1: from localhost/127.0.0.1
2: from same machine
3: from trusted sources
4: from local LAN and subnet
5: from everywhere (default)

Implementing a webportal based on the API

You can implement any website or any responsive webpage based purely on this API only, using XHTML/AJAX requests.
For example:

- login page: use the login API
- displaying a status page: use the getdetails API (here for example you can display the user phone status and balance)
- load/save user details: use the getdetails and setfield API
- display call history: use the cdr API
- price list display: use the rating API

Have a look at the "API" chapter for the short description of these API's.

Direct database operations

Instead of direct SQL connection to the database, you can also use the API to perform various database operations.
The related API entries are the followings: db, setfield, getfield (described above in the "API" chapter).
You can also define custom API's to perform various SQL operations as described here.
(You should use admin authentication with these API calls. Admin access is not recommended from untrusted source such as a JavaScript requests from a webpage!)

Other possibilities

There are also other ways to interact with the Mizu VoIP server, as described in the "Integration Guide" (for example working directly with the SQL database).

Extra comments and examples

Addserver API parameters

- u_type: 0: both, 5: only as traffic sender, 9: only as SIP server (use 0 to add upper servers)
- u_name: name (either the name or the username must be provided)
- u_username: username (either the name or the username must be provided)
- u_password: will be auto generated if empty
- u_domain: domain name (either the domain or the ip must be provided)
- u_ip: IP address. it can be also IP:port (either the domain or the ip must be provided)
- u_port: port number if not set as part of the domain or IP (default is 5060 if not provided)
- u_proxy: outbound proxy (optional)
- u_transport: udp, tcp or tls (default is udp if not set)
- u_auth: the needauth db field (default is 1 which means IP based auth; check the documentation for other possibilities)
- u_email: optional
- u_country: optional
- u_currency: optional

- u_credit: optional
- u_routing: routing entry id or name (set to "null" if no routing entry is required; set to empty to auto-guess).
- u_routingpriority: routing priority
- fieldnameX / fieldvalX: optional extra fields to be set in tb_users (X can go from 0 to 99)

Simple example:
http://10.2.0.72/mvapireq/?apientry=addserver&authkey=23879556&authid=voip_admin&authmd5=ccb612764b35204ed0d212bb80ef931a&authsalt=8394206&u_name=NAME&u_domain=xxx.tokuvoip.com&now=711

NewUser API examples

If you wish to create a new user from an untrusted location (such as from client side or from browser initiated by the user), then use the newuseru API.
Example:
https://sip.autoservice.ltd/mvapireq/?apientry=newuseru&authkey=97275263&u_username=NEWUSERNAME&u_password=PASSWORD&u_name=NAME&u_email=EMAIL&u_phone=PHONE&u_currency=USD&u_country=USA&u_address=x&deviceid=DEVICEID&now=555

If you wish to create a new user from a trusted location (such as your server side web service), then use the newuseri API.
Example:
https://sip.autoservice.ltd/mvapireq/?apientry=newuseri&authkey=97275263&authid=voip_admin&authmd5=c2b9d08390a1d653478d42dc1abd7439&authsalt=9457612&u_username=NEWUSERNAME&u_password=PASSWORD&u_name=NAME&u_email=EMAIL&u_phone=PHONE&u_currency=USD&u_country=USA&u_address=x&credit=5&now=555

Other examples:
http://11.11.11.11:8088/mvapireq/?apientry=newuseri&authkey=75739802&authid=MizuTech_support&authmd5=633b935c465a83054449575db1de4c9e&authsalt=8275681&authsalt2=6406731&u_username=HE&u_password=testpassword12234&u_name=HE&u_email=EMAIL&u_phone=PHONE&u_currency=AUD&u_country=AU&u_address=x&credit=5&now=555

http://11.11.11.11:8080/mvapireq/?apientry=newuseri&authkey=1797368488&authid=fjurajanibiotechcom&authmd5=f3ce3b8cbbf1249e23c5c418ab653118&authsalt=3554881&u_username=NEWUSERNAME&u_password=PASSWORD&u_name=NAME&u_email=EMAIL&u_phone=PHONE&u_currency=USD&u_country=USA&u_address=x&credit=5&now=555

http://11.11.11.11:8020/mvapireq/?apientry=newuseri&authkey=1900228177&authid=voipadmin&authmd5=4efc31ad0fac90a51e6a5192fe29cab7&authsalt=9750607&u_username=NEWUSERNAME&u_password=PASSWORD&u_name=NAME&u_email=EMAIL&u_phone=PHONE&u_currency=USD&u_country=USA&u_address=x&credit=5&now=555

Credit

https://sip1.webvoipphone.com/mvapireq/?apientry=rating&authkey=14150857&authid=1111&authmd5=53233b8a2526e709b7fee4c90bd2c3f3&authsalt=3020822&authsalt2=7597691&bnum=3630&now=1

http://11.11.11.11:32658/mvapireq/?apientry=balance&authkey=66541228&authid=3037&authmd5=55D1734C848443FDC0E4F750D79A33DF&now=555

Cdr API examples

enduser today call count,duration:
days=today&by=all
enduser this month costs:
days=thismonth&by=all
admin todayprofit:
days=today&by=all&fields=all
enduser today call list:
days=today
enduser last 7 days call list:
days=lastweek
enduser last 30 days call list:
days=lastmonth
enduser this month call list with summary:

days=thismonth&summary=yes

admin list last 10 day calls from user id 123 to prefix 40:

days=10&caller=123&prefix=40

enduser missed calls:

days=lastweek&dir=in&type=notconnected

Example:
http://{host}/mvapireq/?apientry=cdr&authkey={authkey}&authid=admin&authmd5={authmd5}e&authsalt={authsalt}&maxcount=1000&days=365&dir=all&type=all&fields=all

http://11.11.11.11/mvapireq/?apientry=cdr&authkey=19729184&authid=telbee_admin&authpwd=xxx&fields=all

Common Tasks

- Login:
  - API: as you can see in the API documentation, each API call have to pass the credentials (it is also possible to trust your web service IP address)
  - DB SQL example: select id from tb_users with(nolock) where username = :username and password = :password and type = 0 and enabled <> 0 and temporarydisabled <> 1
- Dashboard, Single contact view:
  - You should have the important details in your database and the rest can be queried via the webphone API (like the register state) or from our server.
  - For example to get the credit:
    - API example: http://148.251.28.185/mvapireq/?apientry=balance&authkey=11633357&authid=1111&authmd5=2e99ca4d0cd322116771777d569ba1d9&authsalt=7359618&authsalt2=8451828&now=555
    - DB SQL example: select credit, postpaid from tb_users where id = X
- Recent calls, Call history:
  - You can store the call history as you wish from the webphone onCdr callback (and you can also query the CDR records from the server)
- Contact list
  - This should be not closely related with the VoIP server. You can implement any contact list management as you wish.
  - (But if needed, both the webphone softphone skin has a contact API and we can also store the contacts in a voip server side address book if needed)
- Create contact:
  - API example: http://148.251.28.185/mvapireq/?apientry=newuseri&authkey=11633357&authid=ever_admin&authmd5=a0f4f8ca8a6a012dfd540d62bfc92b9d&authsalt=7359618&authsalt2=8451828&u_username=NEWUSERNAME&u_password=PASSWORD&u_name=NAME&u_email=EMAIL&u_phone=PHONE&u_currency=USD&u_country=USA&u_address=x&credit=5&now=555
  - DB SQL example: insert into tb_users (type, username, password, name, email, credit, postpaid) values (0,…

Unreg example

http://SERVERADDRESS/mvapireq/?apientry=unreg&param1=alma&param3=1&authkey=APIKEY&authid=ADMINUSERNAME&authpwd=ADMINPASSWORD&now=ANYNUMBER

Wsuser

https://webrtc.clsa-elcv.ca/mvapireq/?apientry=wsuser&authkey=XXXXXX&authid=USERNAME&authpwd=PASSWORD&brandname=brand&companyname=COMPANY&upperserver=IP&upperserverdomain=DOMAIN

User registered state

Using the API, you can use the reguser request to get the register state of a user like this:
http://SERVERADDRESS/mvapireq/?apientry=reguser&usr=USER_TO_QUERY&authkey=APIKEY&authid=ADMINUSERNAME&password=ADMINPASSWORD&now=555

You can also request the register state of a user from the database using the following SQL query:
select top 1 id from tb_users where username = 'USER_TO_QUERY' and status > 0 and statusdate > getdate() - 0.1 and Enabled <> 0 and temporarydisabled <> 1
This will return the user id if the user is registered, or null/no record if the user is not registered, disabled or doesn't exists.

Voice recording download

http://11.11.11.11/mvapireq/?apientry=voicerecdownload&authkey=95822812&authid=MizuTech_support&callid=sipcallid&maxrecords=1&now=1

https://serverdomain.com/mvapireq/?apientry=voicerecdownload&cdrid=737075&userid=&parentuserid=&maxrecords=1&days=lastmonth&format=1&fromdt=&todt=&caller=&called=&authkey=14150857&authid=voipadmin&authpwd=apwd520185786fr&now=1558422718

You can match the CDR record after the SIP Call-ID.

- On the client/webphone side you can easily get the Call-ID (It is in the CDR which you get with the onCdr callback or you can also request the SIP header with the getsipheader() function).
- On the server you have a “sipcallid” field in the CDR table with the same.

Here is an example SQL query in case if you need it:
SELECT TOP 1 * FROM tb_cdrs WITH(NOLOCK) WHERE datum > getdate() – 1 AND sipcallid = 'xyz'

P2P Call

https://serverdomain.com/mvapireq/?apientry=p2p&authkey=33591933&authid=USERNAME&authpwd=PASSWORD&anum=PHONENUMBER1&bnum=PHONENUMBER2&now=777

http://11.11.11.11/mvapireq/?apientry=p2p&authkey=54742345&authid=550000782542&authmd5=9db1671101a81da6fb2315ba45c27ac2&authsalt=2334557&anum=01199925&bnum=08007774004&now=1

Callback

https://serverdomain.com/mvapireq/?apientry=callback&authkey=33591933&authid=USERNAME&authpwd=PASSWORD&anum=PHONE1&now=111

Various

User credit request (public by default):
*http://serverdomain:8080/mvapireq/?apientry=balance&authkey=31796042&authid=101&authmd5=be23e68277b494ad1680f3ff764ac59e&authsalt=530586&authsalt2=232079&now=555

Rating request:
*http://serverdomain:8080/mvapireq/?apientry=rating&authkey=31796042&authid=USERNAME&authmd5=MD5VALUE&authsalt=MD5SALT&anum=CALLEDNUMBER&now=415

Initiate callback call (for endusers):
*http://serverdomain:8080/mvapireq/?apientry=callback&authkey=31796042&authid=101&authmd5=be23e68277b494ad1680f3ff764ac59e&authsalt=530586&authsalt2=232079&anum=PHONENUMBER&now=555

Send SMS (for endusers):
*http://serverdomain:8080/mvapireq/?apientry=sms&authkey=31796042&authid=101&authmd5=be23e68277b494ad1680f3ff764ac59e&authsalt=530586&authsalt2=232079&anum=MYNUMBER&bnum=TARGETMOBILE&txt=TEXT&now=555

Account signup (for endusers):

*http://serverdomain:8080/mvapireq/?apientry=newuseru&authkey=31796042&u_username=NEWUSERNAME&u_password=PASSWORD&u_name=NAME&u_email=EMAIL&u_phone=PHONE&u_currency=USD&u_country=USA&u_address=x&deviceid=DEVICEID&now=555

Create new user (admin access by default, from trusted IP only):
*http://85.195.93.126:8080/mvapireq/?apientry=newuseri&authkey=31796042&authid=vfoxadmin&authmd5=34723d90476e19758e99110fc4e31d3b&authsalt=530586&authsalt2=232079&u_username=NEWUSERNAME&u_password=PASSWORD&u_name=NAME&u_email=EMAIL&u_phone=PHONE&u_currency=USD&u_country=USA&u_address=x&credit=5&now=555

Add credit to user (admin access by default, from trusted IP only):
*http://85.195.93.126:8080/mvapireq/?apientry=addcredit&authkey=31796042&authid=vfoxadmin&authmd5=34723d90476e19758e99110fc4e31d3b&authsalt=530586&authsalt2=232079&pin=101&credit=5&now=555

List registered users (for admins):
*http://85.195.93.126:8080/mvapireq/?apientry=reguser&usr=x&authkey=31796042&authid=vfoxadmin&authmd5=34723d90476e19758e99110fc4e31d3b&authsalt=530586&authsalt2=232079&now=555

Create new user from untrusted devices:
*http://serverdomain:8080/mvapireq/?apientry=newuseru&authkey=31796042&u_username=USERNAME&u_password=PASSWORD&u_name=NAME&u_email=MAIL&u_phone=PHONE&u_currency=CURRENCY&u_country=COUNTRY&u_address=ADDRESS&deviceid=DEVICEID&now=415

DB query

http://11.11.11.11/mvapireq/?apientry=db&authkey=1393185105&authid=voipadmin&authmd5=6366577df9d9ec703da23ec6aa710aa9&authsalt=6031972&txt=select%20top%201%20username%20from%20tb_users&now=555

Get a free phone number

http://192.168.154.136/mvapireq/?apientry=getphonenumber&authkey=1393185105&count=3&now=555
http://192.168.154.136/mvapireq/?apientry=lockphonenumber&authkey=1393185105&num=3456&now=555

List registered users

Use the “reguser” API to list the registered users: reguser [usr] {show registered users/devices}

It has an optional “usr” parameter, which can have the following values:
-empty: for normal listing
-all: detailed list for all users
-username: an exact user

Example:
http://11.11.11.11/mvapireq/?apientry=reguser&usr=all&authkey=TellfreeApiV2&authid=Duarte&authmd5=02900aa52dbadd1e389931b3e24f12b2&authsalt=8617671&pin=9935274&credit=5&now=555

For more details use the Admin Guide or contact our support