Mizu VoIP Server API

Overview

This document describes the Mizutech VoIP server 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 with admin rights 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.

Quick 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&requestid=1

Add credit: (from trusted IP only)
http://domain.com/mvapireq/?apientry=addcredit&authkey=KEY&authid=USRNAME&authmd5=MD5&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 (port 443 by default) 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 by the "format" query 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)
- WS (websocket. URI must contain "mvstwebsock". Then send the request in websocket as you would send in HTTP URI)
- WSS (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.
- optional “requestid” parameter for easier tracing. For this you can use a positive auto-increment number starting with a positive random number.

More details can be found here.

Supported data **format**s for requests:

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

By default the answer format (Content-Type) 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):

- json (will respond: application/json)
- xml (will respond: text/xml)
- text (will respond: text/plain)
- line (will respond: text/plain with the answer in one line)
- html (will respond: text/html)
- url (will respond: application/x-www-form-urlencoded)
- soap (will respond: application/soap+xml)
- jsonp (will respond: javascript callback code –requires enablejsonp)
- cleartext (will respond in clear text even for HTTP requests. Should not be used.)

Example to request json formatted answer:
http://domain.com/mvapireq/?apientry=apitest1&format=json

The response format is determined in the following way (in this priority order):

1. the format parameter (if set, as described above, usually submitted as URL query parameter)
2. the forcehttpcontenttype global config (if set)
3. guessed from the request (such as the “Accept” HTTP header)
4. the defhttpcontenttype global config (if can’t be guessed)

From JavaScript you might just JSON all time. For structured data you might use JSON and XML. For simple answers / simple parsing the text format might be also fine, especially if you except simple OK/ERROR answers.

Response format:
For simple text content-type the text will always begin with ERROR (for successful requests) or with OK (for failed requests)
With json, xml, soap and formatted answers you will always receive the followings properties in the main node:

- code: [number] 2xx means success, 4xx means failure (400: client error, 401: auth error, 403: forbidden, 404: not found, 422: other error), 5xx: server side error (500: error in processing, 501: feature disabled)
- status: [string] will be set either to “success” or “error”
- success: [bool] will be set to true on success, otherwise missing
- error: [bool] will be set to true on failure, otherwise missing
- data: [any] the response data if any, otherwise null

- message: [string] for successful answers it might be an informative text, such as “OK”. For failed requests it usually contains the error reason
- warning: [string] might be set to a string if the API request succeed but some sub-task failed
- frequest: [string] the requested API function name (what you passed as the “apientry” query parameter)
- requestid: [number] echoing back the requestid parameter from the request if any

- For successful requests the *code* will be 2xx, the *status* will be “success”, the *success* will be true, the *message* might be string or null and the *data* might contain the actual response data.
- For failed requests the *code* will be 4xx or 5xx, the *status* will be “error”, the *error* will be true, the *message* might contain the error reason and the *data* might be null.

Example for successful JSON answer:
Request: http:// 10.2.2.2/mvapireq/?apientry=apitest1&format=json
Answer:

```
{
  "code": 200,
  "status": "success",
  "success": true,
  "message": "OK: api test succeed",
  "data": null,
  "frequest": "apitest1",
  "requestid": 1
}
```

Example for failed JSON answer:
Request:
http://10.2.2.2/mvapireq/?apientry=getuser&param=all&format=json&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369
Answer:

```
{
  "code": 400,
  "status": "error",
  "error": true,
  "message": "ERROR: missing user input",
  "data": null,
  "frequest": "getuser",
  "requestid": 1
}
```

If the httpstrictresponse is set to 1, then the response code is sent as html answer code. Otherwise (by default) you will receive HTTP 200 OK for all requests and the real API response code will be in the message body as the *code* parameter or for simple text format the *message* might begin either with “OK” (for successful requests) or with “ERROR” (for failed requests).

Note: 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).

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
- requestid: you might append a requestid=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

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" or "httpapikey" global config
- 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**: Set as the "apiv2key" or "httpapikey" global config
- **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 to add more complexity for the MD5 (optional, depending if configured on the server)
- authsalt1: client side generated salt as sent by the "authsalt1" parameter (optional but highly recommended. It can be changed in every request or to be changed time to time)

Note:
You can also generate a (test) auth parameters for a user from MManage -> Users and devices -> right click popup menu -> Interact -> Generate API auth. However, the best practice is to generate these at runtime, using a different authsalt for every request or changing it periodically.

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 an optional custom action for this API if the result is success: 1=email, 2=sms, 3=call ring, 4=restart,5=mssqlrestart,6=reboot,7=run; or 5x if the result if error.
- custom_action_body: body text for the above custom_action if required (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)

API Endpoints

Most of 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}
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)
getphonenumber {return a phone number from the free pool)
lockphonenumber {reserve a phone number}
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,fformat,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}
sendpush [type, from_user, to_user, title, message, platform, package, to_token] {send push notification}
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) passed as “sql” parameter. The sql parameter should be URL encoded}
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 (days,fromdate,todate,usertype,parent,checkwarnings)
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 (UPDATE/INSERT/ETC) synchronously (otherwise the more efficient async will be used but that can’t give you a feedback whether the query have succeed or 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!

Example custom API to return the users details for the submitted username:
fname: custom_get_user
customsql: select * from tb_users where username = '[SPARAM]'
example request:
http://10.1.1.20/mvapireq/?apientry=custom_get_user&SPARAM=john&authkey=AUTHKEY&authid=ADMIN&authmd5=ADMINPWD&authsalt=SALT&format=json

Example custom API to update the comment field for the specified user:
fname: custom_set_user_comment
customsql: *update tb_users set comment = '[SPARAM2]' WHERE username = '[SPARAM1]'
example request:
http://10.1.1.20/mvapireq/?apientry=custom_set_user_comment&sparam1=john&sparam2=yeees&authkey=AUTHKEY&authid=ADMIN&authmd5=ADMINPWD&authsalt=SALT

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).

Examples

List users

Use the "**listusers**" function to list the users.
Parameters:

- fields: fields to return. 0: id, username, 1: more, 2: all fields. Default: 0
- offset: page number. -1: no paging, 0+ page number. Default: -1 (no pagination)
- limit: max number of records to return. Set to 0 to no limit. Default: 1000
- Filters:
    - type: -1: all, 0: endusers, 5: traffic senders, 9: sip servers. Default is 0 without special users.
    - u_id: if set, then will list only the specified user (id)
    - parent: -2: list only children, -1: all (default), 0: exclude children, X: will list only users below the specified parent
    - parentinclusive: -2: list only children sub-endusers, 0: list only parent main endusers -1: all (default), X: will list only users below the specified parent and the parent user itself
    - enabled: -1: all, 0: only disabled users, 1: only enabled users. Default: 1
    - active: -1: all, 0: only inactive users, 1: only registered/active users. Default is -1.
    - where: custom SQL where clause (for example "credit > 100 and postpaid < 1"). Default is empty.
- order: order by: -1: auto, 0: no order, 1: id, 2: recently added, 3: username, 4: recently used. Default: -1
- pcr: set to true to list (format the answer) with parent- child hierarchy. Default is false
- Any parameters accepted by the "getuser" API if you wish to include user details

Examples:
List the SIP Servers:
http://10.2.0.72/mvapireq/?apientry=listusers&u_type=9&format=json&authkey=APIKEY&authid=ADMINUSERNAME&authpwd=ADMINPASSWORD

List registered users and their registrar records:
http://10.2.0.72/mvapireq/?apientry=listusers&limit=0&fields=1&u_type=0&active=1&details=basic-statustext-runtime&authkey=APIKEY&authid=ADMINUSERNAME&authpwd=ADMINPASSWORD

A more complicated example listing users below 1002 (including the 1002 user) in parent-child relationship format with the details including the most important fields, the active calls and the statistics for the last 3 days:
http://192.168.52.133/mvapireq/?apientry=listusers&parentinclusive=1002&pcr=true&details=basic-currentcalls-statistics&statdays=3&skipnullfields=true&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369&requestid=555

Note:
To list only main endusers, set the parentinclusive=0. To list only sub-endusers, set the parentinclusive=-2.
Other related API's: regusers, fastregusers, regstat

Add record

Use the **addrecord** API to insert a new record to a table.
The table name must be specified with the table parameter.
Specify any field to be set with its name prefixed with u_, like u_fieldname=value.
You can pass a sync parameter to specify how to execute the insert:

- not set or -1: auto (the execution mode will depend on the system load)
- 0: asynchronously (faster, but no feedback if fails)
- 1: normal/synchronously
- 2: return the new user id (tb_users.id)

Set field

Use the **setfield** API to set any field in a table.
The table name must be specified with the table parameter.
Specify the record to be modified with the rowid parameter (table.id).
Specify any field to be set with its name prefixed with u_, like u_fieldname=value.
You can pass a sync parameter to specify how to execute the update:

- not set or -1: auto (the execution mode will depend on the system load)
- 0: asynchronously (faster, but no feedback if fails)
- 1: normal/synchronously

Add user

Use the **adduser** API to create new users.
Specify any field to be set with its name prefixed with u_, like u_fieldname=value (this will set the tb_users.fieldname to the specified value)
Special fields:

- username: is mandatory (and it should not contain space or special characters)
- type: will be set to 0 (enduser) if not specified
- id: cannot be specified (auto-calculated auto-increment)
- password: if not specified, then a random password will be auto generated
- needauth: if not set, then it will be 1 (IP auth) for traffic senders, otherwise 4 (username/password digest auth)
- it is also possible to assign more numbers for endusers with the num1,num2 … numN parameters
- it is also possible to assign more source address (domain or IP) for IP auth users with the authip1,authip2 … authipN parameters
- if the userautoaddwithowner global config is set, then a parent reseller user will be also added if doesn't exists yet and if the parent field was not specified otherwise
- outbound trunks (such as SIP server users) can be also added to the routing automaticallt
    - routing_id: routing entry id or name (set to "null" if no routing entry is required; set to empty to auto-guess or add new).
    - routing_priority: routing priority if any (defaults to 7)

You can pass a sync parameter to specify how to execute the insert:

- not set or -1: auto (the execution mode will depend on the system load)
- 0: asynchronously (faster, but no feedback if fails)
- 1: normal/synchronously
- 2: normal and return the new user id (tb_users.id)
- If extra numbers or extra auth addresses or routing will have to be set, then the sync will be automatically set to 2

Examples:
Simplest way to add an enduser by specifying only the username:
http://192.168.52.133/mvapireq/?apientry=adduser&u_username=atest1&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

A better way to add an enduser:
http://192.168.52.133/mvapireq/?apientry=adduser&u_type=0&u_name2=atest2&u_username=atest2&u_password=rand2&u_credit=1&u_postpaid=0&sync=2&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Add a child sub-enduser (below the parent user with id 888):
http://192.168.52.133/mvapireq/?apientry=adduser&u_type=0&u_isoperator=0&u_parentid=888&u_username=atest3&u_password=rand3&u_potpaid=0&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Add a traffic sender (allowed to send traffic from IP 5.5.5.5 and 6.6.6.6):
http://192.168.52.133/mvapireq/?apientry=adduser&u_username=in_trunk1&u_type=5&u_needauth=1&u_authip=5.5.5.5&u_authip2=6.6.6.6&u_credit=100&u_postpaid=0&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Add a SIP Server to send traffic to 7.7.7.7:5060 UDP:
http://192.168.52.133/mvapireq/?apientry=adduser&u_username=out_trunk1&u_type=9&u_ip=7.7.7.7&u_port=5060&u_protocol=0&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Set user details

Use the "**setuser**" function to modify any user parameter.
Specify the user with the u_id (users id from tb_users.id) or u_username (tb_users.username) query parameter.
Specify the field to be set with its name prefixed with u_, like u_fieldname=value
You can also change multiple fields at once by using u_fieldname1=value1, u_fieldname2=value2, … u_fieldnameN=valueN
See the tb_users table in the database for the possible fields.

Example to change the comment field to "changed" for user "john":
http://192.168.52.133/mvapireq/?apientry=setuser&u_username=john&u_comment=changed&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Access rights: When the API is invoked by a user (not with admin rights), the u_id is restricted to the authenticated user id and only certain fields can be modified

Get user details

Use the "**getuser**" function to list the users.
Specify the user with the u_id (users id from tb_users.id) or u_username (tb_users.username) query parameter.
The "details" parameter can be one or the combination of the followings separated by -:

- *basic*: the most important fields
- *allfields*: all fields (from tb_users)
- *statistics*: get past statistics: cdrc (call count), sl (speech length/call duration), asr (average success ratio), acd (average call length)
- *currentcalls*: list active calls
- *runtime*: registrar details
- *statustext*: calculate user main state
- *typetext*: type of the user as text
- *credittext*: credit amount as text
- *all*: all details
- *others*: the exact field names from tb_users (for example "telnumber", "credit", etc)
- Default: if no "details" parameter is specified, then it will return id,username

For example:
Single: details=basic
Multiple: details=basic-typetext-statistics

Set the "skipnullfields" parameter to avoid returning null, empty, 0, false, etc values.
If the "details" contains "statistics" then you can also pass a statdays parameter.
The statdays means for how much time to calculate the statistics. 1 means one day. 0.04166 means 1 hour (1/24). Default is 1.
If the "details" contains "statistics" or "currentcalls" then you can also pass a "direction" and "pcr" parameter.
With the "direction" parameter you can specify if to load the statistics as a caller or called. 0: default (caller for endusers and traffic senders, called for SIP servers, etc), 1: outgoing, 2: incoming, 3: both
If the "pcr" parameter is set to true, then the statistics and the currentcalls for parent users will include also sub-users data.

Example to return the basic fields and the main state for user with ID 5, then pass the following query parameters:
Relevant paramaters:

u_id=5
details=basic and statustext

http://10.1.1.1/mvapireq/?apientry=getuser&u_id=5&details=basic-statustext&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Example to list all the details for user "john" forced to JSON output:
http://10.1.1.1/mvapireq/?apientry=getuser&u_username=john&details=all&format=json&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Note:

- *With the "statustext" details you can get the estimated cumulative registrar state of a user. Possible strings are: Unknown, Not Used, Offline, Unregistered, Registered, In Call, Speaking*
- *If you are interested in the exact registrar record(s), then use the "runtime" details instead*
- *For more advanced statistics use the cdr api instead with its stats parameter as described at the Statistics section*
- *When the API is invoked by a user (not with admin rights), the u_id is restricted to the authenticated user id.*

Query user address/login status/equipment

If you are interested in the exact registrar record(s), then use the getuser API for this with the "runtime" parameter, as described above.
Example to query the login (register) state for user "1111":
https://yourserver.com/mvapireq/?apientry=getuser&u_username=1111&details=runtime&authkey=7799315&authid=voip_admin&authmd5=2db2eb29f59f413a2fd8d5056cba9b7d&authsalt=7858858

If the user is registered, then this API will return the runtime registrar record.
If the user is registered from multiple locations (from multiple apps / logins with same id), then it will return multiple registrar records.
The registration records will contain all your mentioned details (device type, device address, created-time, expire-time).
If the user is not registered, then the runtime/registrar answer parameter might be null.

Please note that the runtime registrar details are not stored in any table (for performance reasons, because it would require lots of SQL updates). It is kept only in the server memory, so you must use the getuser API and can't use any SQL query for this.
However, then main state of the user can be also loaded from the tb_user table (checking the status, statusdate,IP, port, equipment, enabled/temporarydisabled fields) if for some reason you prefer this way, but even in this case it is easier to just use the getuser API with the statustext parameter (example below).

Other getuser API examples:

In case if you are interested only in the estimated cumulative registrar state of a user, then you might use the statustext parameter instead (this is simpler):
https://yourdomain.com/mvapireq/?apientry=getuser&u_username=1111&details=statustext&authkey=12345&authid=voip_admin&authmd5=2db2eb29f59f413a2fd8d5056cba9b7d&authsalt=7858858
Possible result statustext strings are the followings: Unknown, Not Used, Offline, Unregistered, Registered, In Call, Speaking

Or you can query both (statustext + runtime) like on the above screenshot:
https://yourdomain.com/mvapireq/?apientry=getuser&u_username=1111&details=statustext-runtime&authkey=12345&authid=voip_admin&authmd5=2db2eb29f59f413a2fd8d5056cba9b7d&authsalt=7858858

Or add also the basic details (basic + statustext + runtime):
https://yourdomain.com/mvapireq/?apientry=getuser&u_username=1111&details=basic-statustext-runtime&authkey=12345&authid=voip_admin&authmd5=2db2eb29f59f413a2fd8d5056cba9b7d&authsalt=7858858

Or get all the user details, like this:
https://yourdomain.com/mvapireq/?apientry=getuser&u_username=1111&details=all&authkey=12345&authid=voip_admin&authmd5=2db2eb29f59f413a2fd8d5056cba9b7d&authsalt=7858858

Delete user

Use the "**deluser**" function to delete a user.
Specify the user with the u_id (user id from tb_users.id) or u_username (tb_users.username) query parameter.
If you wish to delete also sub-users (which has the parent id as this user), then set the pcr parameter to true.

Example to delete user with ID 1234:
http://192.168.52.133/mvapireq/?apientry=deluser&u_id=1234&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Note:
This function should not be used if the user already had any activity, because it will result in orphaned data, such as CDR records without userid.

Use the disableuser function instead of deluser.

Disable user

Use the "**disableuser**" function to disable a user.
Specify the user with the u_id (users id from tb_users.id) or u_username (tb_users.username) query parameter.
Specify how to disable with the disabletype parameter. 0 means temporary, 1: full, 2: both, 3: also rename. Default is 0.
You might also set the unregister parameter to 0 if you wish to keep the existing sessions (by default it will disconnect).
If you wish to disable also sub-users (which has the parent id as this user), then set the pcr parameter to true.

Example to disable user with ID 1234:
http://192.168.52.133/mvapireq/?apientry=disableuser&u_id=1234&disabletype=3&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Note: other related API's: unreg, upperunreg, pushremove

Enable user

To (re)enable users, use the "**enableuser**" API.
It works like the "disableuser" in inverse.

List active calls

Get active calls with the "**currentcalls**" API.
You can filter it to a user id with the u_id query parameter.
If you pass an u_id parameter, then it is also possible to pass direction (0: default, 1: outgoing, 2: incoming, 3: both) and/or pcr (set to true if you wish to include child users)

Example to list all active calls:
http://10.2.0.72/mvapireq/?apientry=currentcalls&authkey=APIKEY&authid=ADMINUSERNAME&authpwd=ADMINPASSWORD

[OLD] NewUser API examples

The newuseri API have been deprecated by adduser.

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.yourdomain.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&requestid=5

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.yourdomain.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&requestid=6

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&requestid=7

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&requestid=8

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&requestid=9

[OLD] Addserver API parameters

Deprecated by adduser

- 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 if any
- 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&requestid=1

Credit -get

Request available amount with the "**balance**" or "**credit**"API.
This is allowed also for endusers to query their own credit and will return the credit as text like "Your credit is 123 USD" with the credit value as two digits precision.
The text is configurable with the following global settings:

- yourcredittext: default is "Your credit is"
- yourcredittextprefix: can be set to "CSIGN" to replace eur/usd simbols, empty to skip or to fix string. Default is CSIGN
- yourcredittextsuffix: can be set to "CURRENCY" to be replaced to the actual currency, empty to skip or to another fix string. Default is CURRENCY

If the API is called with admin right, then u_id or u_username parameter can be passed to specify the user.

Example (for enduser to request its own credit):
http://11.11.11.11:32658/mvapireq/?apientry=balance&authkey=66541228&authid=3037&authmd5=xxx

Example (admin to request credit for a user):
http://domain.com/mvapireq/?apientry=balance&u_username=john&authkey=KEY&authid=USRNAME&authmd5=MD5

Another API to request the credit is "**getcredit**".
This will return the following exact fields separately: credit, currency, postpaid
Example:
http://domain.com/mvapireq/?apientry=getcredit&u_id=55&authkey=KEY&authid=USRNAME&authmd5=MD5

Credit -set

While it is possible to set the credit directly by modifying the user credit field, it is recommended to use the following API's instead for proper bookkeeping and real-time upgrade.

Use the **setcredit** API to set a new credit value (requires admin access rights).
Specify the user with the u_id or u_username query parameter.
Specify the credit with the u_credit query parameter

Example (will set the credit to 10 to user 55):
http://domain.com/mvapireq/?apientry=setcredit&u_id=55&u_credit=10&authkey=KEY&authid=USRNAME&authmd5=MD5

Use the **changecredit** API to increase/decrease the credit with the specified value.
Specify the user with the u_id or u_username query parameter (or if the API caller doesn't have admin rights, then the user will be the current user executing the API call).
Specify the change value with the u_credit query parameter (if positive, then will increase/add credit, if negative then will deduct).
Instead of u_credit you can specify a pin parameter to recharge with pincode (from tb_prepaidcodes).
If the API caller doesn't have admin rights, then only pin code input will be accepted (can't specify u_credit parameter)

Example (will increase the credit with 8 for user 55 by admin):
http://domain.com/mvapireq/?apientry=changecredit&u_id=55&u_credit=8&authkey=KEY&authid=USRNAME&authmd5=MD5

Example (user recharge with pin code):
http://domain.com/mvapireq/?apientry=changecredit&pin=7777&authkey=KEY&authid=USRNAME&authmd5=MD5

Credit -old

These are old/deprecated credit set functions.

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

Recharge with PIN code:
http://domain.com/mvapireq/?apientry=charge&authkey=KEY&authid=USRNAME&authmd5=MD5&anum=PINCODE&requestid=1

Rating

Request call cost with the "**rating**" API:
Request rate to destination (user or app might call this function to request/display the pricing before a call):
Specify the called number with the bnum parameter.

Examples:
https://domain.com/mvapireq/?apientry=rating&authkey=14150857&authid=1111&authmd5=xxx&authsalt=3020822&bnum=3630

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

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

CDR API

Use the "**cdr**" API to query call detail records.
Possible query parameters:
Filters:

- User/number filters:
  - u_id: filter for one user (tb_users.id)
  - u_username: filter for one user (tb_users.username)
  - caller: caller id filter (tb_users.id)
  - called: called id filter (tb_users.id)
  - u_type: filter for user type (0 means endusers, 5 means traffic senders, 9 means SIP servers, etc)

        - prefix: prefix filter for called number
        - cdrid: filter for one single CDR (tb_cdrs.id)
        - callid: filter for one single SIP Call-ID (if stored)
    - Date-time filters:
        - days: lasthour/today/lastday/lastweek/thismonth/lastmonth/thisyear/lastyear or the exact number of days (or fraction)
        - fromdt: from date-time
        - todt: until date-time
        - dsql: custom SQL where clause (URL encoded)
        - if no date-time filters are set, then it will default to lastday (last 24 hours)
    - Other filters:
        - connected: all/yes/no. Defaults to all.
        - dir: all/in/out. Defaults to all. Applied only if userid is set.
        - pcr: true to include sub-users. Applied only if userid, caller or called is set.

Other parameters:

- limit: max number of records to return. Set to 0 to disable. Default is 1000
- fields: min/normal/all (fields returned). Defaults to “normal”.
- isreseller: true means resellers records. Defaults to false.
- by: see the Statistics section below
- stats: see the Statistics section below
- order: date,caller,called,user,null. Defaults to not datum.
- skipnullfields: set to true if you wish to omit null/0/false/empty fields. Default is false.

The generated SQL query will look like this (useful to know if you wish to use custom SQL WHERE clause with the dsql parameter):

```
SELECT
b.datum, u1.username as 'calleruser', b.callernumber, u2.username as 'calleduser', b.callednumber, b.realduration, b.costenduseru,
tb_reasoncodes.text as 'discreason',
(select top 1 tb_directions.name  from tb_directions with(nolock) where b.dirid = tb_directions.id) as 'dirname'
FROM tb_cdrs b WITH(NOLOCK)
left join tb_users u1 with(nolock) on (u1.id = b.callerid)
left join tb_users u2 with(nolock) on (u2.id = b.calledid)
left join tb_reasoncodes with(nolock) on (b.discreason = tb_reasoncodes.code)
WHERE
b.datum > getdate() - 30
ORDER BY b.datum desc
```

Examples:

List all calls from the last day:

http://192.168.52.133/mvapireq/?apientry=cdr&days=1&limit=0&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

List all outgoing calls from user with ID 55 in the last 30 days:

http://192.168.52.133/mvapireq/?apientry=cdr&u_id=55&days=30&dir=out&limit=0&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

List max 100 CDR’s with all fields from the last 30 days:

http://{host}/mvapireq/?apientry=cdr&authkey={authkey}&authid=admin&authmd5={authmd5}e&authsalt={authsalt}&maxcount=100&days=30&fields=all

Other/various example filters:

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

- list last 10 day calls from user id 123 to prefix 40: days=10&caller=123&prefix=40
- missed calls: days=lastweek&dir=in&connected=no

Access rights: when the API is invoked by a user (not with admin rights), the userid is restricted to the authenticated user id.

Statistics

Use the **cdr** API with statistics and by parameters to generate reports.

Specify the required statistics with the stats parameter.
This can be set to one or a combination of the following keywords (separated by -):

- cdrc: call count
- sl: speech length / total duration
- asr: average success ratio
- acd: average call duration / average call length
- endusercost: total enduser cost (our revenue paid by the users)
- providercost: total provider cost (our cost to be paid for call termination)
- profit: total profit (difference between endusercost and providercost)
- Advanced/rarely used statistics:
  - act: average connect time (call setup and ring time between incoming INVITE and 200 OK)
  - asrb: average call duration when success means call duration of at least 30 seconds
  - profit_percent: profit percent (for the difference between endusercost vs providercost)
  - salescost: sales margins if any
  - companycost: company cost or profit if sales diff
  - othercost: other costs (top reseller cost)
  - succ: number of successful calls
  - rtp: media statistics
  - avg_endusercost: estimated average enduser cost per minute
  - avg_providercost: estimated average provider cost per minute

You can set it to one value (for example &stats=asr) or to multiple values separated by - (for example &stats=cdrc-sl-asr-acd)
If nothing is set (but the by parameter is not empty) then it will load cdrc-sl.

Specify the required grouping with the by parameter.
This can be set to one or a combination of the following keywords (separated by -):

- day: month-day
- hour: hour
- calleruser: caller username
- caller: caller number
- callerid: caller user id
- calleduser: called username
- called: called number
- calledid: called user id

You can set it to one value (for example &by=calleruser) or to multiple values separated by - (for example &by=day-hour)
If nothing is set (but the stat parameter is not empty) then it will load the statistics with no grouping applied (total).

Examples:
Main system statistics in the last day:
http://192.168.52.133/mvapireq/?apientry=cdr&days=1&by=all&statistics=cdrc-sl-asr-acd&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Main system statistics by day for the last month:

http://192.168.52.133/mvapireq/?apientry=cdr&days=lastmonth&by=day&statistics=cdrc-sl-asr-acd&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Basic statistics for the calls made by user 78922 in the last 30 days:
http://192.168.52.133/mvapireq/?apientry=cdr&u_id=78922&dir=out&days=30&statistics=cdrc-sl-asr-acd-endusercost&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

SIP servers (outbound trunks) statistics for the last 10 days:
http://192.168.52.133/mvapireq/?apientry=cdr&u_type=9&days=10&statistics=cdrc-sl-asr-acd-providercost-profit&by=calleduser&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

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&requestid=1
    - 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&requestid=1
    - DB SQL example: insert into tb_users (type, username, password, name, email, credit, postpaid) values (0,…

Unregister user

Use the “unreg” function to force unregister a user.
Specify the user with the u_id (users id from tb_users.id) or u_username (tb_users.username) query parameter.
Example:
http://SERVERADDRESS/mvapireq/?apientry=unreg&u_username=john&authkey=APIKEY&authid=ADMINUSERNAME&authpwd=ADMINPASSWORD&requestid=ANYNUMBER

Wsuser

https://webrtc.domain.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&requestid=141

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.

Note: You might use the getuser function instead of this.

Voice recording download

http://11.11.11.11/mvapireq/?apientry=voicerecdownload&authkey=95822812&authid=MizuTech_support&callid=sipcallid&maxrecords=1&requestid=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&requestid=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&requestid=777

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

Callback

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

Various

Initiate callback call (for endusers):
*http://serverdomain:8080/mvapireq/?apientry=callback&authkey=31796042&authid=101&authmd5=be23e68277b494ad1680f3ff764ac59e&authsalt=530586&authsalt2=232079&anum=PHONENUMBER&requestid=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&requestid=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&requestid=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&requestid=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&credit=5&requestid=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&requestid=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&requestid=415

Rebill

Use the “rebill” API to recalculate CDR pricing (for example if you made changes to the billing which should be applied for previous calls).
You might just send this command from the Server Console.
Possible parameters:days,fromdate,todate,usertype,parent,checkwarnings
Example command to rebill last month CDR records:
rebill,31

Assign phone number to users

You can store phone numbers in the tb_telnumbers table which can be also managed from MManage Phone Numbers form.
See the “DID numbers” chapter in the admin guide for the details.

During the onboarding process, you might present a list of phone numbers the user from the free pool, so the users will be able to choose their preferred one.
This can be achieved with the **getphonenumber** API.
This API will list free-to-use phone numbers and will temporary lock them (so other new users will not take it during the onboard process)
Use the limit parameter to specify the number of free-to-use phone numbers to return (default is 1).
Use the ttype parameter to specify the number type to return (default is 0).
If you call the API with admin rights, then you might also pass a u_id or u_username (the user for which you wish to allocate the number if already known)

Example to list 5 free-to-use phone numbers and temporary lock them:

http://192.168.52.133/mvapireq/?apientry=getphonenumber&limit=5&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

When the user selects a phone number, then you can use the **lockphonenumber** API to reserve the number (prevent allocating it to other users).
Specify the phone number with the num parameter.
If you call the API with admin rights, then you might also pass a u_id or u_username (the user for which you wish to allocate the number, if already known). If you don't specify a num parameter, then a free number will be automatically loaded, locked and returned.

Example to lock phone number 01888888:
http://192.168.52.133/mvapireq/?apientry=lockphonenumber&num=01888888&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369

Use the **assignphonenumber** API to actually allocate a phone number to a user (tb_users.telnumber or tb_users.username)
Specify the user with the u_id or u_username parameter.
Specify the phone number with the num parameter. If you don't specify a num parameter, then a free number will be automatically loaded and that number will be returned from this API.
The assignas parameter means the followings: 0: save as username if missing, otherwise as telnumber, 1: save as username, 2: save as telnumber. Default is 0.
You can also set a verify parameter to one of the following values: 0: don't verify even the user, 1: num is not from tb_telnumbers, 2: num is from tb_telnumbers, but don't reject if not, 3: full/normal verify. Default is 3.

In case if your onboarding process doesn't require the above described phone number select and lock, then you can just use this function alone.
This API requires admin access by default.

Example to assign number 01888888 to user 55 (which should be already locked):
http://192.168.52.133/mvapireq/?apientry=assignphonenumber&u_id=55&num=01888888&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369&format=json

Example to automatically load a number and assign it to user 55:
http://192.168.52.133/mvapireq/?apientry=assignphonenumber&u_id=55&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369&format=json

If you don't wish to use the tb_telnumbers/Phone numbers form to manage your DID numbers, then you can just assign any number to a user like this:
http://192.168.52.133/mvapireq/?apientry=assignphonenumber&u_id=55&num=0199999&verify=1&authkey=1393185105&authid=voipadmin&authmd5=e486c239327e8d1c49c6e693674129fe&authsalt=8661369&format=json

If you wish to implement a full onboarding process with user selectable number, then:
1. first use the getphonenumber API to present a list of available numbers to the user
2. once the user select a number, call the lockphonenumber API (to prevent assigning it to some other user during the onboard)
3. once the new user have been added/saved, call the assignphonenumber to actually assign the previously loaded phone number

Get one or more (a list of) phone numbers from the free pool
getphonenumber {return a phone number from the free pool)
lockphonenumber {reserve a phone number}

http://192.168.154.136/mvapireq/?apientry=getphonenumber&authkey=1393185105&count=3&requestid=101
http://192.168.154.136/mvapireq/?apientry=lockphonenumber&authkey=1393185105&num=3456&requestid=102

getphonenumber {return a phone number from the free pool)
lockphonenumber {reserve a phone number}

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=777&authid=ddd&authmd5=02900aa52dbadd1e389931b3e24f12b2&authsalt=8617671&requestid=555

Note: other related API: listusers

File upload

File uploads are treated specially.
See the “Built-in webserver” FAQ point in the Admin Guide for the details.

Announcements

The **announce** API can be used to send a message (predefined voice file, text message or pin code) to the specified destination (usually by calling the destination number but SMS and email is also supported)

Typical use-case:

1. Customer make an announce API request passing the destination number with the file/message/pin code to be sent
2. The server will make a call to the destination and will play the message, then hang-up

Parameters:

- target: destination phone number or username (or email address if email should be used)
- content: message to be sent (name of the voice file, text or pincode)
- language: optional short language code (such as en, fr, hu, etc)
- source: optional source number or email address (can be disabled which means that it will be ignored from the API; defaults to current user TelNumber or Username)
- callid: optional SIP Call-ID (globally unique identifier for this call; by default it will be automatically generated)
- type: optional delivery method: 1: call, 2: sms, 3: email. Default is global setting announcement_type.
- mode: announcement mode: 1: file/text, 2: PIN. Default is global setting announcement_mode.

Examples:

Simple example to make a call to +36308153374 and playback file "announcement.wav" (assuming that both announcement_type and announcement_mode is set to 1 and announcement.wav file exists in the voice folder):
https://sip.example.com/mvapireq/?apientry=announce&authkey=XXX&authid=USERNAME&authmd5=MD5&authsalt=XXX&target=+36308153374&content=announcement

Same as above, but also specify the source number (e.g. +36308153373) and the language set to hu (assuming that announcement_hu.wav file exists in the voice folder):
https://sip.example.com/mvapireq/?apientry=announce&authkey=XXX&authid=USERNAME&authmd5=MD5&authsalt=XXX&target=+36308153374&content=announcement.wav&source=+36308153373&language=hu

Pin2Speech (assuming that the announcement_mode is set to 2 or empty)

https://sip.example.com/mvapireq/?apientry=announce&authkey=XXX&authid=USERNAME&authmd5=MD5&authsalt=XXX&target=+36308153374&content=123&mode=2&format=json

Sending SMS (assuming that the announcement_type global config is set to 2 or it is empty)
https://sip.example.com/mvapireq/?apientry=announce&authkey=XXX&authid=USERNAME&authmd5=MD5&authsalt=XXX&target=+36308153374&content=hello&type=2

Announcements related global configurations:

- announcement_type: specify default type for the announce API. -1: not set (will accept from API), 0: disable, 1: call, 2: sms, 3: email
- announcement_mode: specify default mode for the announce API. -1: not set (will accept from API), 0: undefined, 1: file (the "text" parameter in the API will be what we play), 2: pin2speech (the "text" parameter in the API will be the pin code)
- announcement_permission: 0: the global announcement_type/announcement_mode setting will overwrite the API type/mode parameters. 1: allow the API to specify type/mode, even if already set in global settings. default is 1.
- announcement_sourcenumber: specify a fix source number for the announcement. "0" is a special value meaning don't allow passing via API
- announcement_result: the ability to generate call id and then store the results in the CDR record (as tb_cdrs.simid) with the announce_result API. -1: from the “aid” true/false parameter, 0: disable, 1: enable
- announcement_result_maxduration: for how much time should we lookup for the result expressed in days (1 means 1 day, 0.1 means 2 hours and 24 minutes, 0.04 means around 57 minutes. Default is empty/infinite, but highly recommended to set for a reasonable value for performance optimization)
- announcement_maxduration: max call time for announcements in seconds. Default is 120.
- localcallcodecs: codecs to offer for locally initiated calls in priority order. 115,18,0,8,3 which means OPUS-wideband,G.729,PCMU,PCMA,GSM
- mincreditonroute: minimum credit required for the client to proceed with this API call (the exact pricing will be checked again at routing)

Considerations:

- The related voice files must be present in the voices folders as described also in the IVR documentation “Voices” chapter.
- The language can be loaded also from the global or user settings as described also in the IVR documentation “Language and localization” chapter.
- If the language is set, then make sure that there is a corresponding filename_languagecode.wav for the required files (for example your_pincode_is_fr.wav). Otherwise the server will load the default (your_pincode_is.wav) which are usually in English.
- The source number is loaded in the following order (items below can overwrite the source set from items above):
    1. The source API parameter (if announcement_sourcenumber is not set to “0”)
    2. The API client TelNumber if set (the Other Number field from MManage -> Users and Devices form)
    3. The announcement_sourcenumber global config (specify if you need the same source number for all calls)
    4. Any Caller-ID settings at the routing time. See the “Caller ID Settings” chapter in the Admin Guide for the details.
- You can disable this API by setting the announcement_type global config to 0.

Pincode announcement setup:

Pincode to call/sms is a specific use-case for the announce API, when the mode is set to 2.

The related global settings are the followings:

- announcement_mode: set to 2 (or loaded from the announce API mode parameter)
- pin2speech_play_prefix: voice file to play before the pincode. Default is: “your_pincode_is”
- pin2speech_play_suffix: voice file to play before the pincode. Default is empty
- pin2speech_txt_prefix: text to send before the pincode if type is 2 (sms) or 3 (email). Default is: “Your pincode is:”
- pin2speech_txt_suffix: text to send after the pincode if type is 2 (sms) or 3 (email). Default is empty
- pin2speech_txt_subject: email subject if the type is 3 (email). Default is: “Pincode”
- pin2speech_play_repeat: how many times to repeat the pincode in the call. Defaults is 1

- pin2speech_play_delay: delay between rounds and at the end in seconds. Defaults to 1

Here is a typical configuration:
- announcement_mode: 2
- announcement_type: 1
- pin2speech_play_repeat: 2 (or whatever you wish)
- The pin2speech_play_prefix/ pin2speech_play_suffix settings and related files.
- If multiple languages are required, then the files in the required languages should be also created and copied to the voices folder.
- announcement_result: set to 1 only if you wish to use also the announce_result API
- announcement_result_maxduration: 0.04 (if the announce_result API will be called within one hour after the initial announce API)

The **announce_result** API can be used to submit the operation result (such as success or failure).
This is useful to track the API usage results with CR (conversion rate) statistics.
If enabled (aid/ announcement_result is set to 1), then the announce API will return a session ID in the answer data.
Then you can call the announce_result API to submit the result with the same passed as aid parameter and a aresult parameter with the operation result. The value for the aresult must be between 1 and 99 (inclusive). For example you can use 1 for fail and 2 for success.
These result codes will be save to the CDR records (tb_cdrs.simid) for later queries or to generate CR statistics.

Example:
https://sip.example.com/mvapireq/?apientry=announce_result&authkey=XXX&authid=USERNAME&authmd5=MD5&authsalt=XXX&aid=1234&aresult=2

Custom SQL commands

Custom SQL commands can be executed with the "**db**" API.
Pass the sql query as the sql parameter (URL encoded!).
For select statements the API will return the data. Otherwise will return if the exec succeed or failed.
Optional parameters:
- isopen: set to true if the sql is a statement which should return data (such as select). Set to true if the sql needs to be executed (such as update). If not set, then it will be guessed from the sql text.
- sync: for exec statement set the sync to 0 to execute asynchronously (faster, but no error feedback) or to 1 to execute synchronously (you will know if the exec succeed or failed). Default (if not specified) will be auto set depending on system load.

Example:
http://11.11.11.11/mvapireq/?apientry=db&sql=select top 5 * from tb_users&authkey=777&authid=ddd&authmd5=02900aa52dbadd1e389931b3e24f12b2&authsalt=8617671&requestid=555

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

Custom API example

Custom API's can be added into **tb_api** as described in this document.
For example you can see/edit it from MManage -> Direct query form -> select * from tb_api.
(Double click on the memo fields if you wish to edit them)

Example custom API to return the users details for the submitted username:
fname: custom_get_user
customsql: select * from tb_users where username = '[SPARAM]'

example request:
https://yourserverdomain.com/mvapireq/?apientry=custom_get_user&SPARAM=1111&authkey=7799315&authid=voip_admin&authmd5=2db2eb29f59f413a2fd8d5056cba9b7d&authsalt=7858858

Example custom API to update the comment field for the specified user:
fname: custom_set_user_comment
customsql: *update tb_users set comment = '[SPARAM2]' WHERE username = '[SPARAM1]'
example request:
https://yourserverdomain.com/mvapireq/?apientry=custom_set_user_comment&sparam1=1111&sparam2=yeees&authkey=7799315&authid=voip_admin&authmd5=2db2eb29f59f413a2fd8d5056cba9b7d&authsalt=7858858

These API's are already set in the tb_api table and you can try them as described at the above mentioned example requests.
You can modify them as you wish or add more API's after your needs.

For more details use the Admin Guide or contact our support

Copyright © MizuTech