DataChannel API Example

Applies to VoipNow 5.X.X and higher!

This page contains an example of how to get notifications about different events dispatched to your API customer by Hubgets via a DataChannel Event. These events are JSON-formatted WS messages sent to apps when such apps are connected to the DataChannel API. The consumers are never notified about the events that occurred in the past or prior to their connection to the DataChannel API.

What you should do before running the example

Requirements	
Node.js modules	readline, https, url, path, tools
socket.io	Socket.io, socket.io-client

The example in this article requires a valid token. To find out how to get a valid token, check the **Obtain Authorization** page.

Here's how to execute the example:

node <connector> <mode>
Where:

- `<connector>` is the script used for connection to DataChannel API;
- `<mode>` can be ***user***, ***app*** or ***bot*** depending on the authentication method of choice; each of them is configured under a key with the same name in the configuration file.

To keep things simple, we will save the configuration file as `ws-connector.config.json`. The configuration file will have the following content

ws-connector.config.json

```
{
	/*
		Place here a URL containing protocol, host and port only; a path in here has another effect
		than expected!
	*/

	"uri": "https://<FQDN of the VoipNow server>",

	"user": {
		/*
			These cookies can be extracted from a browser after authentication.
		*/
		"vn": "sessionId",
		"dev": "deviceId"
	},

	"app": {
	      "token": "<TOKEN>",
		/*
			If no OAuth token is specified above, you must provide the ID and secret of the
			Hubgets mobile app found by running this query on the VoipNow database:
				select * from app where platform = 'mobile' \G
		*/
		"id": "<APP KEY of the Hubgets mobile app>",
		"secret": "<APP SECRET of the Hubgets mobile app>",
		"username": "<username>",
		"password": "<password>"
	},

	"bot": {
		"token": "<VALID TOKEN FOR A BOT>"
	}
}
```

The configuration parameter for each mode is in the configuration file. In our example, we'll be using the ***app*** mode.

There are two methods to make this work:

Method #1

1. Use a valid token generated once you obtain the user authorization following the recommendations in the **Obtain Authorization** page.
2. Replace <FQDN of the VoipNow server> with a valid FQDN that points toward your VoipNow web server and install the valid, signed, SSL certificate.
3. It will not work with a self-signed SSL certificate. If you want it to work with an already generated token, replace <TOKEN> with the actual token.

Method #2

1. Use Hubgets Mobile App and get the <APP KEY> and <APP SECRET> from the database.
2. Replace the corespondent variables in the configuration file.
3. Add the username and password of the user where the APP will be authorized.
4. To get the <APP KEY> and <APP SECRET>, you need to connect to the SQL CLI and run the following query:

```
select app_id,app_secret from app where app_alias='HubgetsMobile';
```

5. Ensure the configuration file is correct, save the snippet below as ***ws-connector.js*** in the same folder as the configuration file saved earlier.

ws-connector.js

```
'use strict';

/**
 *
 * Make sure you specify a proper configuration in 'ws-connector.config.json'.
 *
 * Usage: <code>node ws-connector mode</code> where <code>mode</code> is one of 'user', 'app' or 'bot',
depending on
 * the desired authentication method. Each method is configured under a key with the same name in the
configuration
 * file.
 */

var io = require('socket.io-client');
var readline = require('readline');
var urlParser = require('url');
var https = require('https');
var pfs = require('path');
var tools = require('tools');
var config = tools.configuration.compileConfiguration(module);

if (process.argv.length < 3) {
        console.log("Usage:\n\tnode " + process.argv[1] + " <mode>\nwhere <mode> is one of [user, app,
bot].");
        process.exit(1);
}
if (!config) {
        console.log("No configuration file was found.");
        process.exit(1);
}

var mode = process.argv[2];
var socket;
var test=1;

var multiline = "";
var input = readline.createInterface({
        input: process.stdin
});

var STANDARD_PATH = "/datachannel";
var GRANT_TYPE = "mobile";
var nonce='';

function generateRandAlphaNumStr(len) {
  var rdmString = "";
```

```
  for( ; rdmString.length < len; rdmString  += Math.random().toString(36).substr(2));
  return  rdmString.substr(0, len);

}

function normalLine(line) {
        if (line === '"') {
                multiline = "";
                input.on('line', multiLine);
                input.removeListener('line', normalLine);
        }
        else if ((line = line.trim())) {
                try {
                        socket.send(JSON.parse(line));
                }
                catch (error) {
                        console.error("Invalid JSON. " + error);
                }
        }
}

function multiLine(line) {
        if (line === '"') {
                try {
                        socket.send(JSON.parse(multiline));
                }
                catch (error) {
                        console.error("Invalid JSON. " + error);
                }

                input.on('line', normalLine);
                input.removeListener('line', multiLine);
        }
        else multiline += line + "\n";
}


function setUpSocket(error, headers) {
        if (error)
                throw error;

        socket = io(config.uri, {
            transports: ['websocket'],
            path: pfs.join(config.path || STANDARD_PATH, config.instanceId || "xyz"),
            extraHeaders: headers
        });

        socket.on('message', function(message, callback) {
                console.log('RECEIVED', JSON.stringify(message) + "\n");
                if (callback)
                        callback();
        });

        socket.on('connect', console.log.bind(console, "CONNECTED"));
        socket.on('error', console.error);
        socket.on('disconnect', console.log.bind(console, "DISCONNECTED"));
        input.on('line', normalLine);
}

switch (mode) {
        case 'user':
                if (!config.user)
                        throw new Error("No authentication data found for user mode.");

                setUpSocket(null, {
                        cookie: "vn=" + config.user.vn + ";dev=" + config.user.dev
                });
                break;

        case 'app':
                if (!config.app) {
```

```
                        throw new Error("No authentication data found for app mode.");
                }

                if (config.app.token) {
                        setUpSocket(null, {
                                authorization: "Bearer " + config.app.token
                        });
                }
                else authClient(setUpSocket);
                break;

        case 'bot':
                if (!config.bot || !config.bot.token)
                        throw new Error("No authentication data found for bot mode.");

                setUpSocket(null, {
                        authorization: "Bearer " + config.bot.token
                });
                break;

        default:
                throw new Error("No such mode '" + mode + "'.");
}

function authClient(callback) {
        var uri = config.uri;
        var path = "hubgetsb/oauth/mobile/";

        if (uri.charAt(uri.length - 1) !== '/')
                uri += '/';
        uri += path;

        var options = urlParser.parse(uri);
        options.method = 'POST';

        options.headers = {
                "content-type": "application/x-www-form-urlencoded"
        };

        options.rejectUnauthorized = config.verifyCertificate;

        var req = https.request(options, function(res) {
                var result = {
                    status: res.statusCode,
                    content: ""
                };

                res.setEncoding('utf8');
                res.on('data', function(chunk) {
                        result.content += chunk;
                });
                res.on('end', function() {
                        try {
                                result.content = JSON.parse(result.content);
                        }
                        catch (err) {
                                err.message += "\nMalformed response from Voipnow:\n" + result.content;
                                callback(err);
                        }

                        if (result.status !== 200)
                                return callback(new Error(result.content.error + ": " + result.content.
error_description));

                        callback(null, {
                                authorization: "Bearer " + result.content.access_token
                        });
                });
        });

        req.on('error', function(err) {
```

```
                callback(err);
        });
        req.write('grant_type=' + GRANT_TYPE + '&client_id=' + config.app.id + '&client_secret=' +
config.app.secret
                + '&username=' + config.app.username + '&password=' + config.app.password);
        req.end();
}
```

6. Make sure to enable the SIP notifications, edit */etc/voipnow/local.conf* and set the notification line as shown below.

```
SIP_NOTIFICATIONS 1
```

Restart Kamailio after enabling notifications.

How to run the example above

To connect to the DataChannel API, you need to run the following command:

#> node ws-connector.js app

If there is no error or exception returned, the notifications about the events that occur for a certain user, the one for which the APP was authorized, will start to flow as shown below.

CONNECTED

RECEIVED {"timestamp":1516707549,"class":"extension/register","from":"5@user","nonce":"ISft8PYKT1at","payload":{"extensionNumber":"0003*006","contact":"sip:0003*006@192.168.3.126:9463","registerExpire":"120","deviceIP":"192.168.3.126:9463","deviceId":""}}

RECEIVED {"timestamp":1516707642,"class":"phonecall/update","from":"5@user","nonce":"6fQyHG49AEet","payload":{"phoneCallId":"LjACJWyAY74XHTFvFqHMTBOg","view":"00","extension":{"extensionId":"7","number":"006","extendedNumber":"0003*006"},"status":"CREATED","pbxapp":"DIAL","hold":false,"parked":false,"recorded":false,"flow":"OUT","callerid":null,"dialed":"*52","answered":null,"started":"2018-01-23T11:40:42Z","nonce":null,"disposition":"UNKNOWN","author":{}}}

RECEIVED {"timestamp":1516707642,"class":"phonecall/update","from":"5@user","nonce":"8usAaaycxJTa","payload":{"phoneCallId":"LjACJWyAY74XHTFvFqHMTBOg","view":"00","extension":{"extensionId":"7","number":"006","extendedNumber":"0003*006"},"status":"UPDATED","pbxapp":"DIAL","hold":false,"parked":false,"recorded":false,"flow":"OUT","callerid":null,"dialed":"*52","answered":"2018-01-23T11:40:42Z","started":"2018-01-23T11:40:42Z","nonce":null,"disposition":null,"author":{}}}

RECEIVED {"timestamp":1516707645,"class":"phonecall/update","from":"5@user","nonce":"WhUtGrq4lud5","payload":{"phoneCallId":"LjACJWyAY74XHTFvFqHMTBOg","view":"00","extension":{"extensionId":"7","number":"006","extendedNumber":"0003*006"},"status":"DESTROYED","pbxapp":"DIAL","hold":false,"parked":false,"recorded":false,"flow":null,"callerid":null,"dialed":null,"answered":"2018-01-23T11:40:42Z","started":"2018-01-23T11:40:42Z","nonce":null,"disposition":"UNKNOWN","author":null}}

The entire list of the DataChannel Events is not available yet, but it will be soon published on our wiki.

There are events sent for extensions (delete, create,update, register), phone calls (new call, hangup call, transferred called, etc), new topics being added, deleted, updated or closed in Hubgets, the presence status of the users in Hubgets and many more.