Apifonica WebSDK

User Guide

Version 2.2.4

2017

Copyright Information

Contact:

Address:	**Fredrikinkatu 55, 5th floor. OOIOO Helsinki**
Phone:	**+358 9 31 57 51-61**
E-mail:	**info@apifonica.com**
Home Page:	**http://www.apifonica.com**

Revision History

Date	Version	Description	Author
2016.07.10	1.0.0	First release (Version 1.0.0)	Grigory Tyschenko
2016.11.10	2.0.0	Updated function code	
2017.02.09	2.1.0	WebRTC demo project has been updated	
2017.04.24	2.2.0	WebSDK code has been updated to support push notification service between mobile devices and the Apifonica server	
2017.05.21	2.2.1	WebSDK code has been updated with API documentation	
2017.06.29	2.2.2	The WebSDK server-side code has been updated with a new version, 2.2.4.bld.2, to support the Apifonica service	
2017.09.11	2.2.3	The WebSDK server-side code has been updated. Proofreading for grammar and style	Grigory Tyschenko Alexey Tkachenko
2017.09.20	2.2.3 BLD.03	Document is edited and supplemented after testing WebSDK installation from scratch	Alexey Tkachenko
2017.09.28	2.2.4	WebPhone (WebRTC) demonstration has been updated	Grigory Tyschenko

CONTENTS

Introduction

The objective of this WebSDK documentation is to give any developer with iOS or Android experience the knowledge and understanding required to manage the functionality provided by the Apifonica platform.

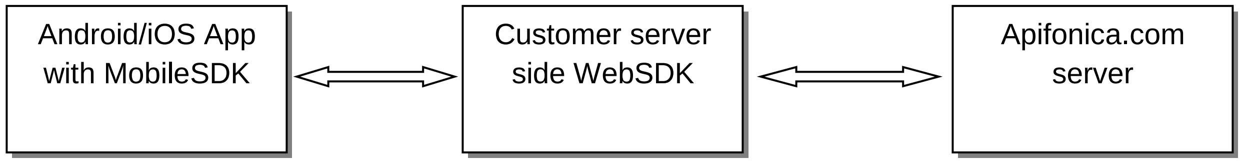
Particularly, the server-side WebSDK is required to operate the rented telephone numbers and incoming/outgoing VOIP calls through communication between the apifonica.com server and the client's iOS/Android/WebPhone applications with pre-installed MobileSDK packages. Interaction between services is performed via REST API requests.

The WebSDK package consists of two parts:

- the server-side service;
- the WebPhone demo project.

The Web SDK server-side service is used to establish connections among mobile (iOS/Android) and WebPhone devices and the Apifonica service.

The WebPhone (WebRTC) demo project contains functions to rent phone numbers and to make VoIP phone calls in the customer's web browser.



General Features

Server-side Features:

- Server management. Server interacts with MobileSDK apps and Apifonica server;
- Monitoring of rented phone numbers;
- Monitoring of incoming/outgoing calls via XML scenarios;
- Synchronization of rented numbers stored on your server side with rented numbers stored in your Apifonica account.

WebPhone Example Features:

- Renting phone numbers;
- Making incoming/outgoing VoIP calls based on the WebRTC service;
- Audio connection with audio codecs (OPUS/G722/G711(u/a-law)/SHA-256);
- Support for Google Chrome, Mozilla Firefox and Safari web browsers.

Requirements

- Node.js (ver.5.x+). Please refer to https://nodejs.org/en/download/package-manager/ for Node.js installation instructions;
- npm (package is automatically installed during Node.js installation);
- MySQL (ver.5.x+). Please refer to https://dev.mysql.com/doc/refman/5.7/en/installing.html for MySQL installation instructions.

Server-side Setup

1. Make sure you have all the required packages listed above installed.
2. Install Forever service in console mode. Forever will make your server persistent, so that once you log off it will still run. It will restart even if *node.js* throws an error.

```
sudo npm install forever –global
```

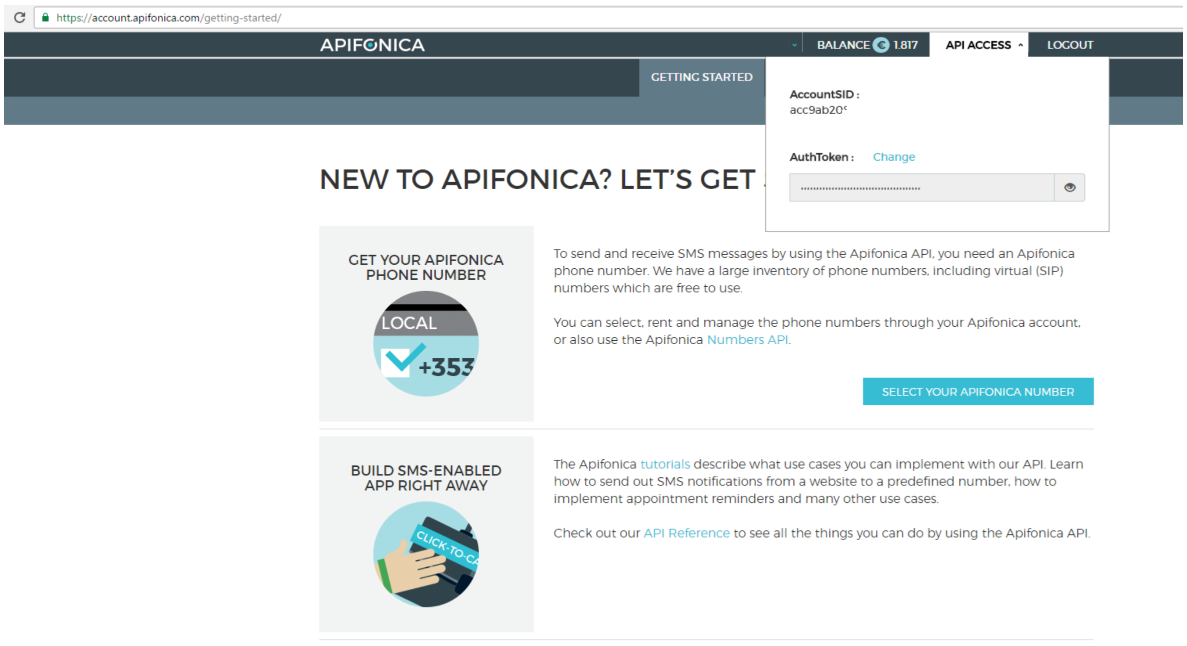
3. Copy and unpack the WebSDK archive package directly to the server where WebSDK needs to be deployed.
Note: If you are re-installing WebSDK over an existing installation, save the existing 'keys' folder and **config.js** file to a backup location. When re-installing WebSDK, copy the 'keys' folder and config.js file from the backup to the server's root folder.

4. Using console mode, open the project's root folder, enter the command **npm install** and wait until all the modules are installed:

```
cd apifonica_javascript_websdk
npm install
```

5. Register on the **apifonica.com** web site, copy your 'AccountSID' and 'AuthToken' codes from the web site's API access section when you are logged in (as shown in the figure below) and enter these values into the **config.js** file: sid (AccountSID), token (AuthToken):

```
apifo: {
            sid:      "accxx-xxxx-xxxx",
            token:   "autxx-xxx-xxxx"
       },
```



AccountSID :
acc9ab20‹

AuthToken : Change

NEW TO APIFONICA? LET'S GET

GET YOUR APIFONICA PHONE NUMBER

To send and receive SMS messages by using the Apifonica API, you need an Apifonica phone number. We have a large inventory of phone numbers, including virtual (SIP) numbers which are free to use.

You can select, rent and manage the phone numbers through your Apifonica account, or also use the Apifonica Numbers API.

BUILD SMS-ENABLED APP RIGHT AWAY

The Apifonica tutorials describe what use cases you can implement with our API. Learn how to send out SMS notifications from a website to a predefined number, how to implement appointment reminders and many other use cases.

Check out our API Reference to see all the things you can do by using the Apifonica API.

6. Set your domain name and ports for http and https connections in the **config.js** file:

```
// Deploy address
domain:develop?"http://127.0.0.1:8000":"http://websdk.yourserver.com:8000",
// port used to http
port:8000,
// port used to https
https_port:8080,
```

7. Open the **package.json** file and correct your domain name:

```
"url": "http://websdk.yourserver.com:8000/",
"sampleUrl": "http://websdk.yourserver.com:8000/",
```

8. For first-time installations, create a self-signed certificate from the project root using console mode:

```
cd apifonica_javascript_websdk
mkdir keys
cd keys
```

```
openssl genrsa -des3 -out server-tmp.key
openssl rsa -in server-tmp.key -out server.key
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 3650 -in server.csr -signkey server.key -out server.crt
```

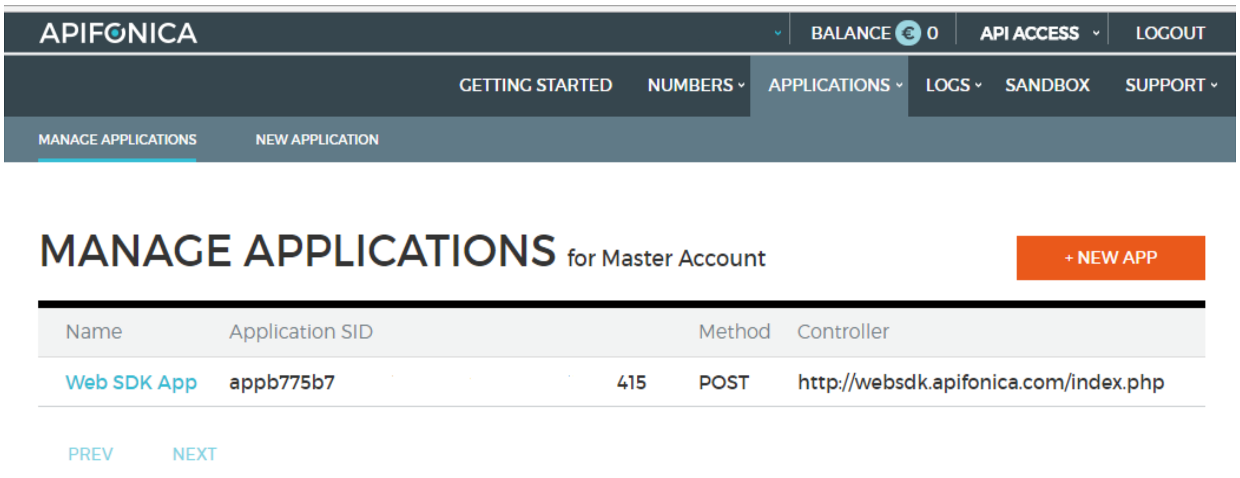
9. Set up the MySQL database.
   9.1. Using console mode, run these commands:

```
mysql –uUSERNAME –pPASSWORD;

CREATE DATABASE IF NOT EXISTS websdk_example;

USE websdk_example;

CREATE TABLE IF NOT EXISTS `websdk_sessions` (
        `session_id` VARCHAR(64) NOT NULL PRIMARY KEY,
        `data` LONGTEXT NOT NULL,
        `expires` integer
    );

CREATE TABLE IF NOT EXISTS `websdk_number` (
        `id` SERIAL PRIMARY KEY NOT NULL,
        `number` varchar(16) NOT NULL UNIQUE,
        `password` varchar(64) NOT NULL,
        `number_sid` varchar(64) NOT NULL,
        `user_id` integer REFERENCES number ON DELETE SET NULL,
        `token` varchar(128),
        `push_id` varchar(512)
    );

CREATE TABLE IF NOT EXISTS `websdk_users` (
        `id` SERIAL PRIMARY KEY NOT NULL ,
        `login` varchar(64) NOT NULL UNIQUE,
        `pass` varchar(64) NOT NULL,
        `app_id` varchar(128)
    );
```

9.2. Add one user with his/her account to the 'websdk_users' table:

```
INSERT INTO websdk_users (login,pass,app_id) VALUES ( "login", "password",
"Application_SID_code");
```

where "Application_SID_code" is your real Application SID value. The Application SID identifier can be created in the Applications section of the Apifonica account web site (https://account.apifonica.com) by clicking the "New App" button, entering the new Application Name and controller URL and saving the entered values. The new Application SID will appear in the list of applications on the Manage Applications page (https://account.apifonica.com/applications/manage/):



MANAGE APPLICATIONS for Master Account

Name	Application SID	Method	Controller
Web SDK App	appb775b7 415	POST	http://websdk.apifonica.com/index.php

9.3. Exit the MySQL client:

```
exit;
```

Note: You can also create the database with the tables from the ready-to-use SQL schema in the **my_schema.sql** file:

```
mysql –uUSERNAME –pPASSWORD < my_schema.sql;
```

Remember: You must add at least one user with his/her account to the 'websdk_users' table.

10. Edit the **config.js** file and insert your MySQL settings:

```
 db:{
      host     : 'localhost',
      user     : 'USERNAME',
      password: 'PASSWORD',
      database: 'websdk_example'
      },
```

11. Edit the **config.js** file and set the "default_user" value to the ID of the active user you have added to the ‘websdk_users’ table in Step 10. This ID is used to identify the user whose rented numbers will be synchronized with the Apifonica server:

```
default_user:develop?1:2,
```

Starting the Server-side Service

1. Start your server using these commands in console mode:

```
cd apifonica_javascript_websdk
forever start app.js
```

Note: You can also use the **node app.js** command to start the server, but this command does not guarantee server restart after it goes offline:

```
node app.js
```

2. Use this command to preview the log files generated by running *forever* processes:

```
forever logs
```

3. Use **forever list** to verify that the server process is running. You need to make sure that no other process with the same name is running at the same time:

```
forever list
```

4. To restart your server, use this command:

```
forever restart {process uid}
```

where the 'process_uid' can be seen by running the **forever list** command.

5. To stop the previously started server processes use this command in console mode:

```
forever stop app.js
```

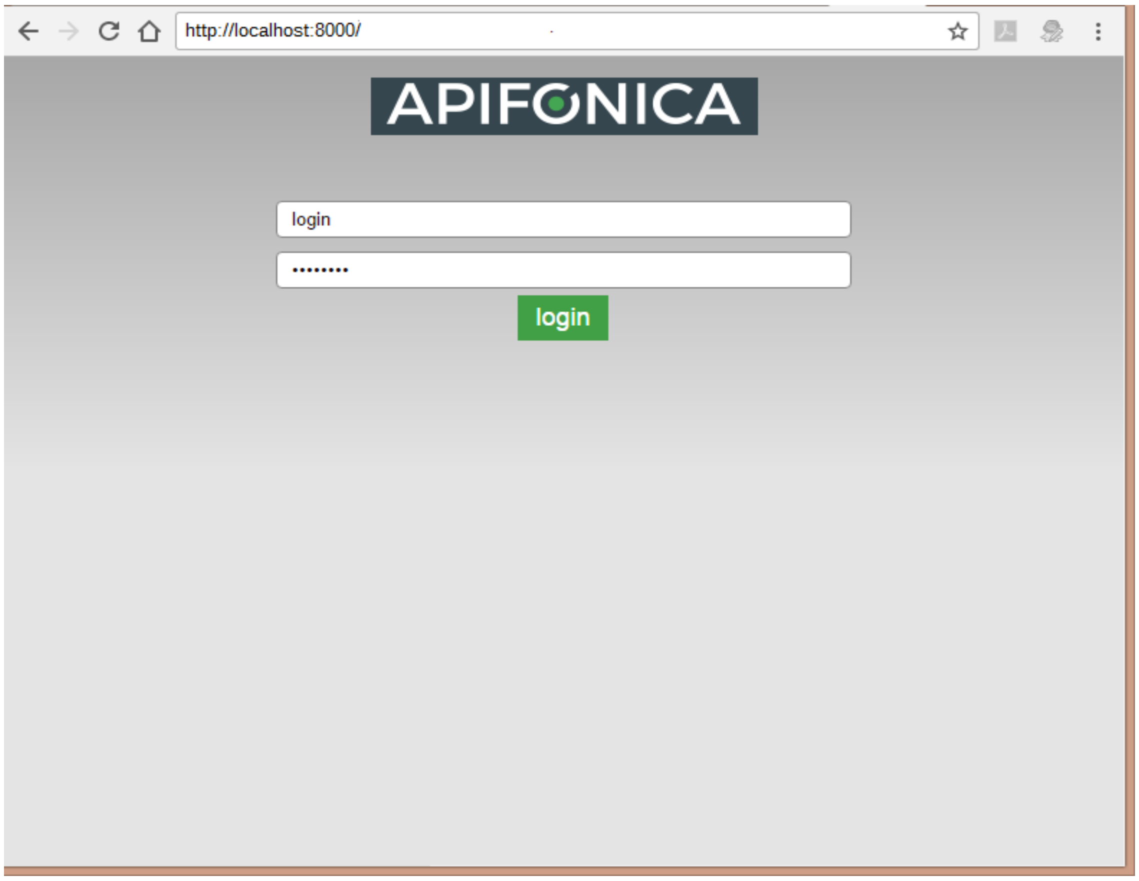
6. Open this URL in your browser to view a server-side management example:

https://websdk.yourserver.com:8080/register

7. Open this URL in your browser to see a WebPhone (WebRTC) demonstration:

https://websdk.yourserver.com:8080/

If you have gone through all the topics, you can look at the WebSDK login form for a WebPhone demonstration.

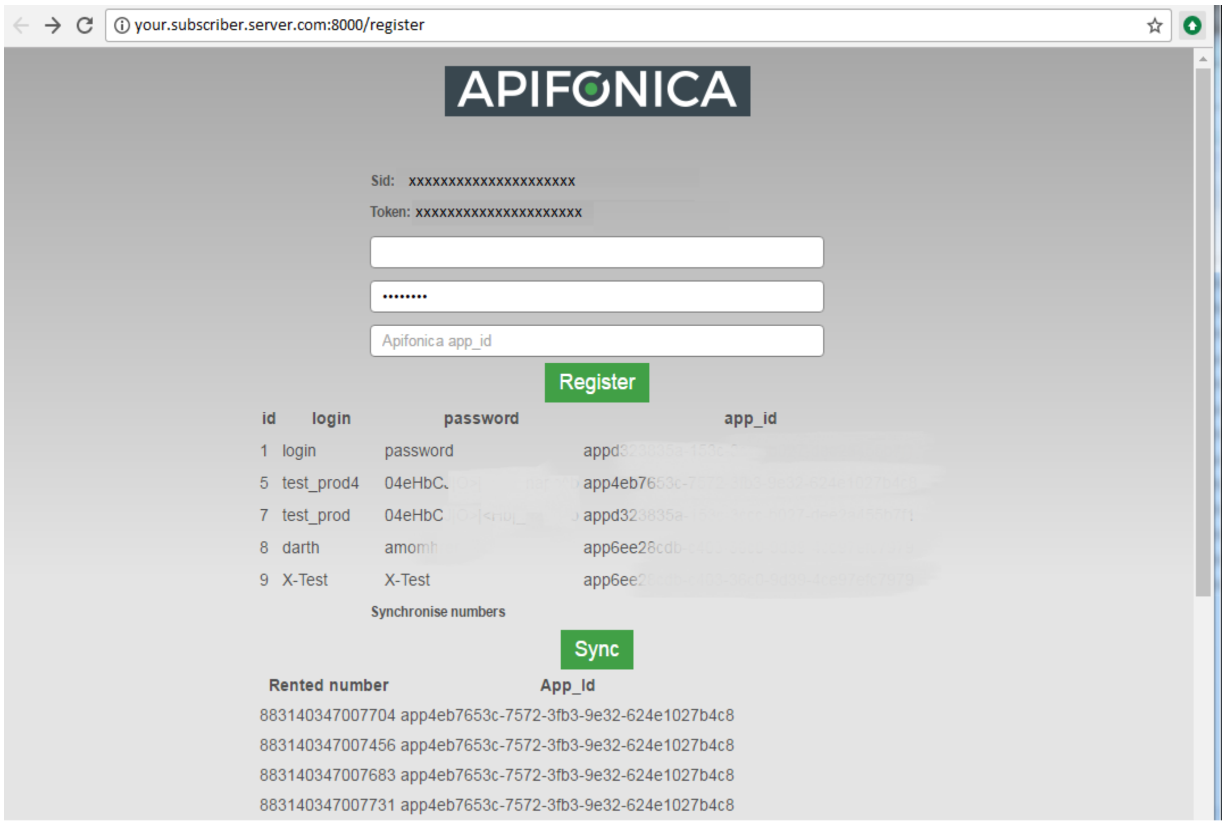


Server-side Management Example

1. Here is the URL which demonstrates server management and its relation to MobileSDK applications and the **apifonica.com** server:

https://websdk.yourserver.com:8080/register

2. Depending your server's security settings, a window may pop up prompting you to enter a login and password in order to access the page that allows you to enter the authorization details issued to you by your server administrator.



your.subscriber.server.com:8000/register

APIFONICA

Sid: xxxxxxxxxxxxxxxxxxxxx

Token: xxxxxxxxxxxxxxxxxxxxx

••••••••

Apifonica app_id

Register

id	login	password	app_id
1	login	password	appd
5	test_prod4	04eHbC	app4eb76
7	test_prod	04eHbC	appd32
8	darth	amom	app6ee28
9	X-Test	X-Test	app6ee

Synchronise numbers

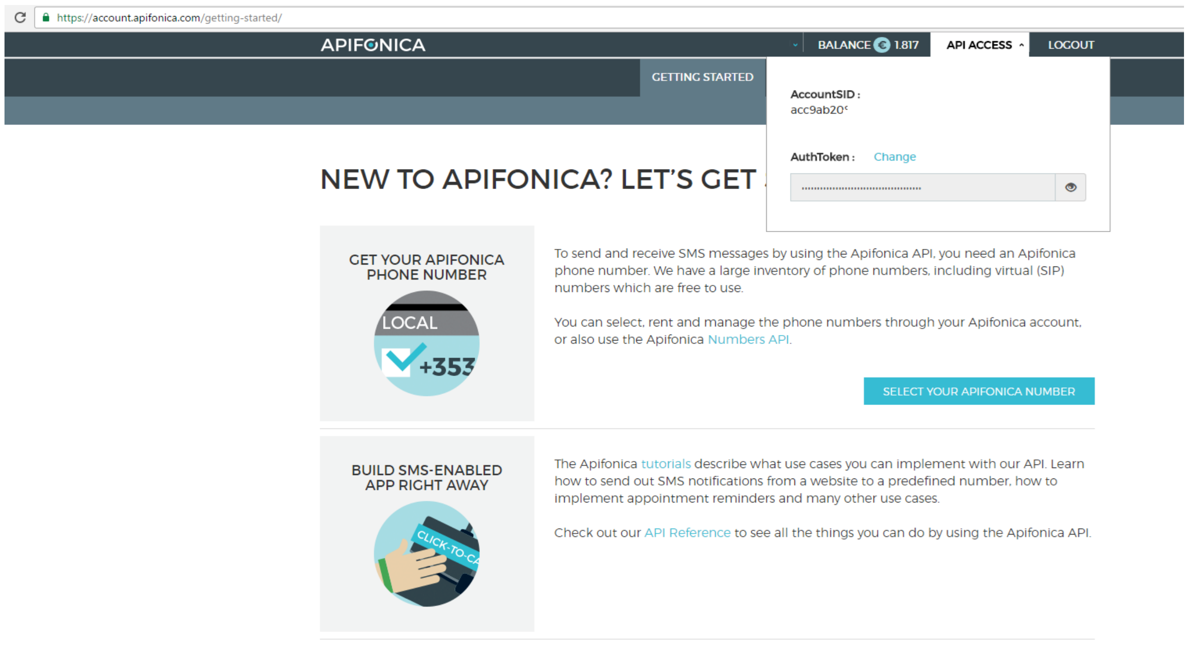
Sync

Rented number	App_Id
883140347007704	app4eb7653c-7572-3fb3-9e32-624e1027b4c8
883140347007456	app4eb7653c-7572-3fb3-9e32-624e1027b4c8
883140347007683	app4eb7653c-7572-3fb3-9e32-624e1027b4c8
883140347007731	app4eb7653c-7572-3fb3-9e32-624e1027b4c8

3. The page displays the current values of your user identifiers (AccountSID / AuthToken), depending on the details shown in your Apifonica account on https://account.apifonica.com web site:

> Sid: 'AccountSID'
> Token: 'AuthToken'

These values can then be compared with the user data stored in your Apifonica account.


https://account.apifonica.com/getting-started/

APIFONICA BALANCE € 1.817 API ACCESS LOGOUT

GETTING STARTED

AccountSID :
acc9ab20‹

AuthToken : Change

NEW TO APIFONICA? LET'S GET

GET YOUR APIFONICA PHONE NUMBER

To send and receive SMS messages by using the Apifonica API, you need an Apifonica phone number. We have a large inventory of phone numbers, including virtual (SIP) numbers which are free to use.

You can select, rent and manage the phone numbers through your Apifonica account, or also use the Apifonica Numbers API.

SELECT YOUR APIFONICA NUMBER

BUILD SMS-ENABLED APP RIGHT AWAY

The Apifonica tutorials describe what use cases you can implement with our API. Learn how to send out SMS notifications from a website to a predefined number, how to implement appointment reminders and many other use cases.

Check out our API Reference to see all the things you can do by using the Apifonica API.

The page (https://websdk.yourserver.com:8080/register) also displays the current list of available accounts for iOS/Android applications with pre-installed MobileSDK packages to connect to.

4. You can now enter the available commands for the WebSDK server side.

Note: The current demo case at the URL above is provided for demonstration purposes only. Depending on your server security policies, it may be strongly unrecommended to use this example in the live mode!

List of Available Commands for Server-side Management

1. Create a new account to connect WebPhone/iOS/Android applications with MobileSDK packages installed, to rent a phone number or to use already rented phone numbers to make voice calls. The user enters his/her desired username and password, sets the "Application SID" code for which a new account will be created to connect with iOS/Android applications and clicks the "Register" button.

Note: The "Application SID" code identifier can be created in the Applications section of the Apifonica Account web site (see Step 9.2 of the WebSDK installation above). If you are planning to have multiple iOS/Android applications, it is possible to create an "Application SID" identifier for each of them.

2. Synchronize the rented numbers stored on the WebSDK server with the rented numbers stored in your account on **apifonica.com**. This command can be executed by pressing the "Sync" button. It is recommended to perform this command immediately after the first launch of the WebSDK server. As

a result, the list of rented numbers will be displayed under the "Sync" button. Note that there is no indication when the synchronization is finished.

Note: This command synchronizes the data in the MySQL database with the rented numbers only for the current user ID specified in the **config.js** file:

```
default_user:develop?1:2
```

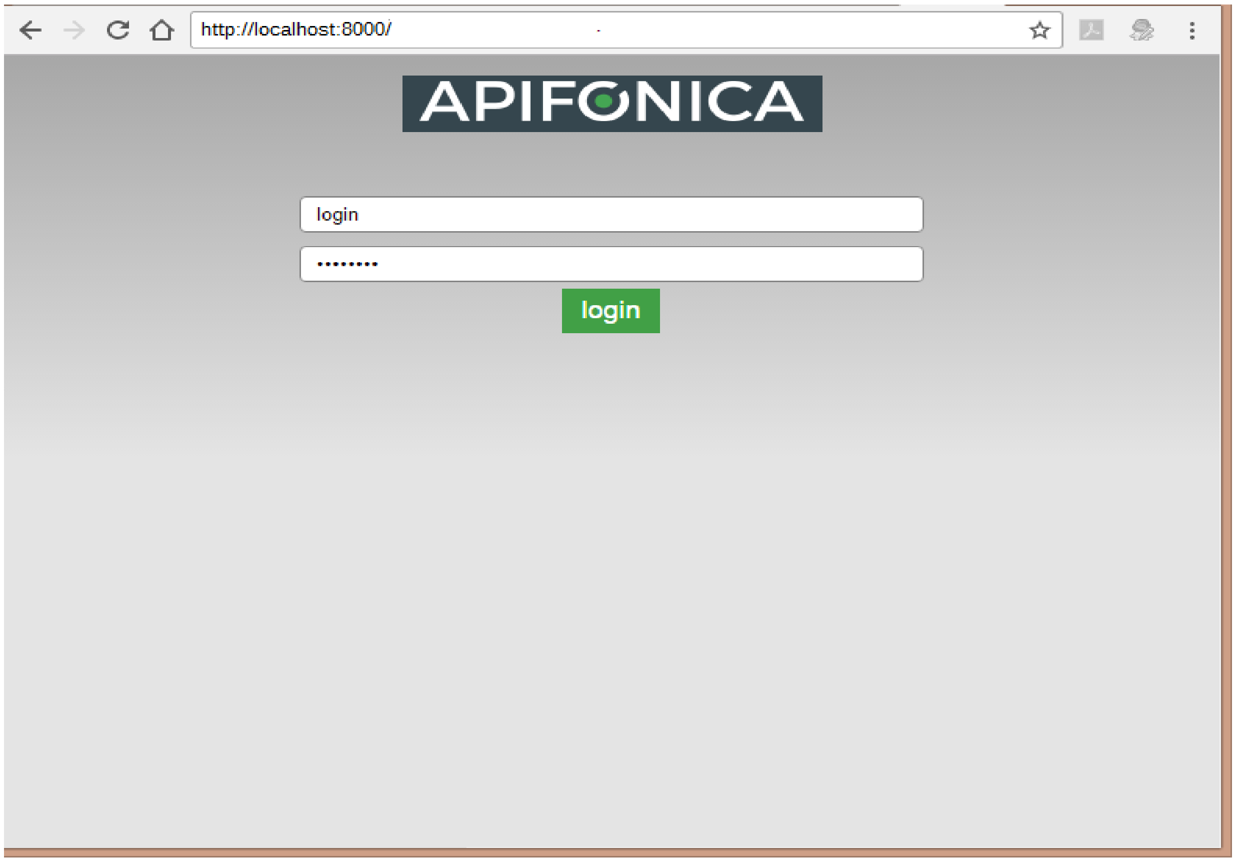
WebPhone (WebRTC) Demonstration

1. To view the WebPhone (WebRTC) demonstration, open this URL in your web browser:

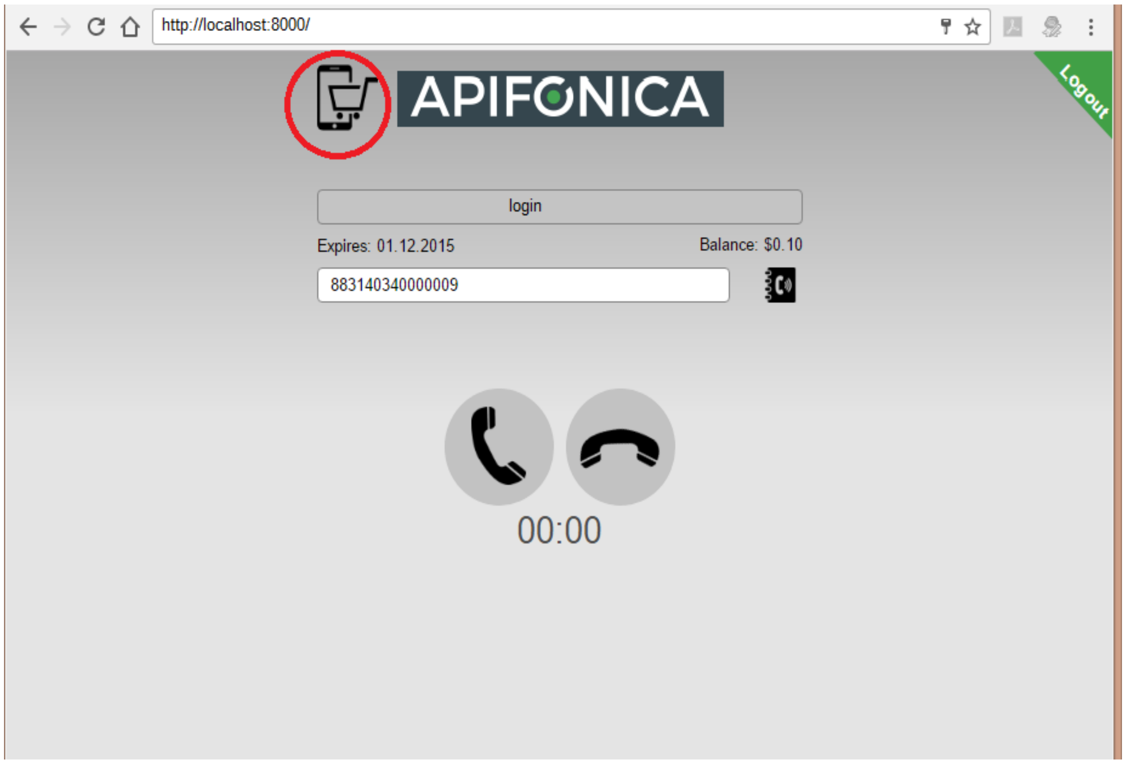
https://websdk.yourserver.com:8080/

You should see the login form for the WebPhone demonstration.

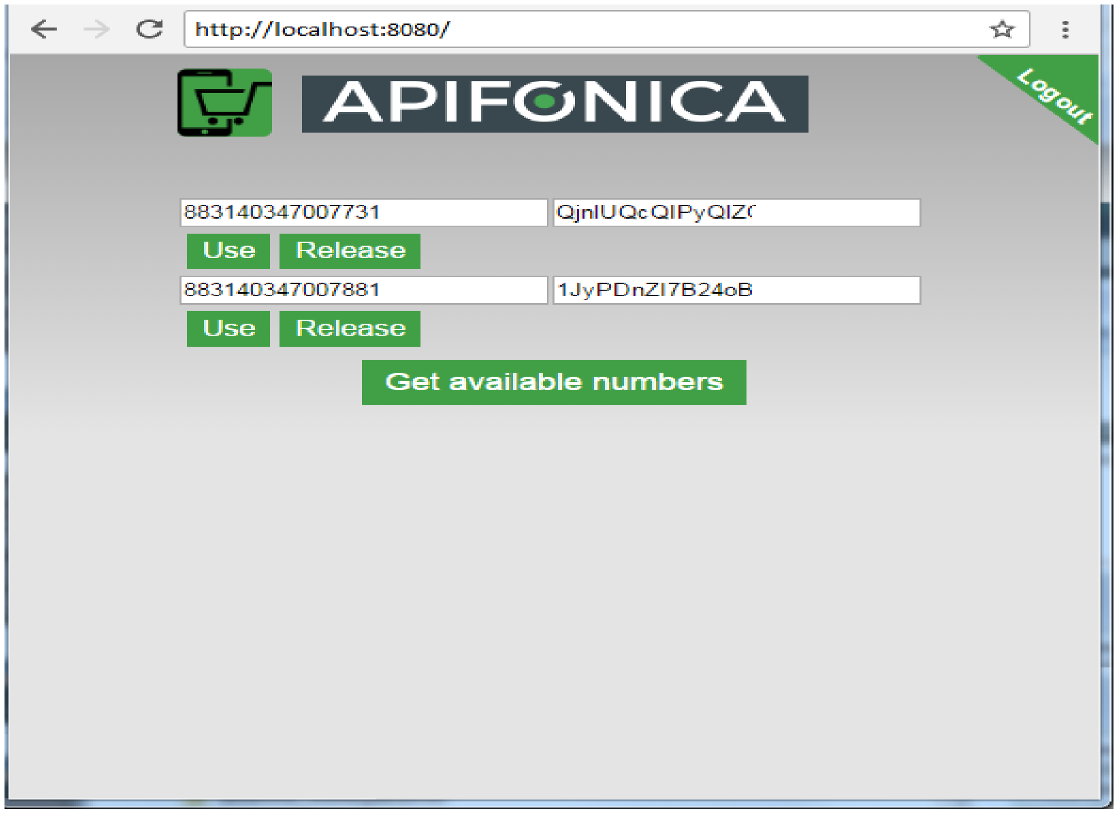
2. Enter the login and password associated with the user ID used by default in Server-side Management and click the "login" button.



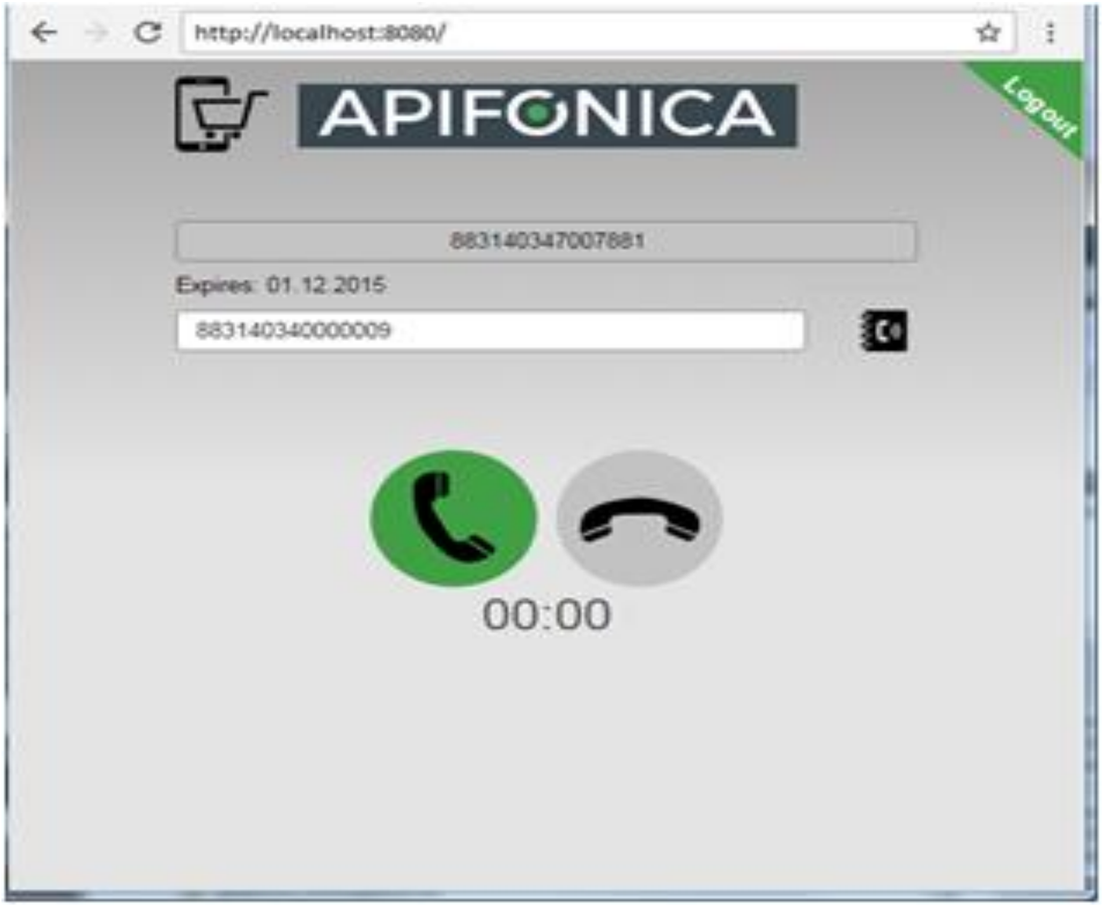
3. Click the "Food Basket" button to rent a new phone number.



4. Select any rented phone number and click the "Use" button or click the "Get available numbers" button to rent a new number.



5. You can now make any phone call.



Setting Up the WebPhone (WebRTC) Demonstration

1. Readme file: **public/apifojs/readme.md**
2. HTML file: **templates/callout.jade**
3. CSS file: **templates/style/callout.less**
4. JS file: **public/js/callout.js**
5. Interface elements in folder: **public/img**
6. Main files: **app.js, engine.js, config.js**
7. Router with API example: **routes/index.js**

Main Functions Used in WebPhone (WebRTC) Demonstration

Where It All Begins

The document's *ready* function is where the Apifonica JS Object is initialized:

```
$(document).ready(function() {
        Apifo.onerror=function(error){
            ui.error(error);
            ui.ready_to_call();
        }
        Apifo.onready = onReady;
        Apifo.onfinish = ui.ready_to_call;
        Apifo.incoming = onIncoming;
        Apifo.init(number.number, number.password);
});
```

Registering

The UI starts off with a basic login screen that accepts your credentials. To enter the username and password values for login, use the following code:

```
function login() {
        Apifo.init(number.number, number.password);
}
```

The *onReady* listener function, which was registered at the start, is invoked when Apifonica sends back the *onReady* event:

```
function onReady() {
        ui.ready_to_call();
        ui.bind_call_events();
}
```

Making a Call

Enter the number in the post-login UI and click the "Call" button. This action causes the following code to run:

```
call_button.onclick=function(){
                var number=document.getElementById('call_to').value
                if(!/\+?[\d]{10,15}/.test(number)){
                          ui.error('Not correct number');
                }else{
                          ui.start_counter();
                          Apifo.call(number);
                }
      }
```

After discarding validations and UI state changes, the code boils down to one line:

```
Apifo.call(number);
```

Handling Incoming Calls

By creating the onIncoming call listener, the Apifonica JS object can handle calls coming into the Apifonica number.

```
function onIncoming(call) {
                console.log('Incoming call from', call.caller);
                answer_button.onclick = function(){
                          Apifo.answer();
                          ui.start_counter();
                };
                hangup_button.onclick = function(){
                          Apifo.cancel();
                          ui.stop_counter();
                }
      }
```

The two actions that can be performed then are *answer* and *reject*.

Terminating a Call

This code may be used to terminate a call.

```
Apifo.cancel();
```