LocalSense Wireless Positioning System

WebSocket Interface Development Manual (js) v2.2

April 2021

Revision Description

Version	Description	Date
V1.9	Added MD5+salt for password	201912
V2.0	Modified the connection port number of WebSocket to 48300	20200229
V2.1	Supported 64-bit tag analysis (compatible with the original 32-bit)	20200926
V2.2	Added longitude and latitude coordinate analysis for 0xb4 tags, and global coordinate analysis of 0xb4 tags	20210329
	Supplemented five output modes of position data	20210426

1. Interface Description

LocalSense WebSocket API (JavaScript version) implements WebSocket client with JavaScript in accordance with the "Localsense Client Communication Protocol", obtains positioning data, tag power, alarm information and other data, and provides alarm switch, tag vibration and other control functions.

Users can easily obtain data and send commands through the API interface.

2. File Description

The detailed folder description in LocalSense WebSocket Api is as follows:

1. demo folder: Call routine program

1) css folder: Save the style related files.
2) js folder: js library that the demo depends on.
3) index.js: demo's main js, providing api call instances.
4) tool folder: Save the salting related files.

2. WebSocket api folder: api library

1) localsense_websocket_api.js: WebSocket api main program
2) reconnecting-websocket.js: Reconnectable basic WebSocket implementation
3) md5.min.js: md checking

3. Call Process

1. Copy the WebSocket api folder to the appropriate directory of the project
2. Introduce three js files, localsense_websocket_api.js, reconnecting-websocket.js, md5.min.js under the WebSocket api folder.
3. Call the relevant API functions

1) Define an api variable
   var ws_api = window.LOCALSENSE.WEBSOCKET_API;
2) First, call the function "ws_api.SetAccount" to set the user name, cleartext passwords and salt value to log in (the account name is admin and the password is #LocalSense by default)

2.1. The password will be MD5 salted in version 2.6.3 and later. The internal

process is as follows:

a) Assuming that the login password is a string: pwd = "#LocalSense";

b) Assuming that the salt value is a string: salt "abcdefghijklmnopqrstuvwxyz20191107salt:

c) The final encrypted string is: salt_en_val=MD5(MD5('#LocalSense')+abcdefghijklmnopqrstuvwxyz20191107salt)

d) Please encrypt and save the input salt value. demo/js/tool provides a way to save the salt value encrypted with aes for reference

2.2. The salt value of the software versions before 2.6.3 is empty, and only use the MD5 method to log in

Note: MD5() represents the MD5 algorithm function
Call the required interface function according to the user's needs

3.1 Basic information, including position and power, etc.

Call interface: ws_api.RequireBasicInfo(url)

3.2 Other information, including distance, anchor status information, etc.

Call interface: ws_api.RequireExtraInfo(url)

3.3 Control information, including alarm switch status, video linkage return parameter information, set tag temporary withdraw time, tag vibration buzzer, etc.

1) Call interface: ws_api.RequireControlInfo(url)

2) Call the relevant control interface
Call the relevant interface in api to complete the corresponding functions。according to the user's needs. For the interface list, see the next section.

3) Define the interface to get the information
Implement the interfaces such as onRecvTagPos, onRecvGaojing, onRecvDmData, onRecvModfiyData, onRecvAppendInfo, onRecvPersonInfo, onRecvErrorInfo, and onRecvBaseStData in api to obtain the relevant data. For the interface description, see the next section. For specific use, refer to the example in index.js in the demo.

4) Close connection

Close the methods WebSocket connection through the RejectBasicInfo, RejectExtraInfo, and RejectControlInfo interfaces. For specific use, refer to the example in index.js in the demo.

4. Interface List

The specific interfaces in localsense_websocket_api.js are as follows:

1) Call the method RequireBasicInfo to open the WebSocket for obtaining the basic information, including tag location, alarm information, roll call data, modification data, personnel information, etc., as follows:

Method Name	Parameter Type	Description
ws_api.onRecvTagPos (value)	object	Process tag position data. bindmaster: Tag ID capacity: Tag capacity information id: Tag ID regid: Map ID where the tag is located sleep: Whether the tag sleeps (the value is true/false) timestamp: Timestamp for WebSocket tag x/y/z: x/y/z coordinates of the tag timestamp_web: The time of the tag information received on the Web end Reserverd: Reserved field
ws_api.onRecvTagPos Bin(value)	Object	Obtain the hexadecimal source code of the tag position
ws_api.onRecvGaojing(value)	object	Process alarm data. id: Alarm id related_tagid: Alarm related id timestamp: Current alarm timestamp type: Alarm type
ws_api.onRecvGaojing Bin(value)	object	Obtain the hexadecimal source code of the alarm data
ws_api.onRecvDmData (value)	object	Process roll call data areaTagNumber: Detailed information of the tag in the area area_name_length: Area name length

		area_tag_number: Area tag number areaname: Area name object (stored in ASCII value) id: Area id obligatetag: Reserved field
ws_api.onRecvDmDataBin(value)	object	Obtain the hexadecimal source code of the roll call
ws_api.onRecvModfiyData(value)	object	Process all data modified by Tag, Effence, and Group type: Notification type param: Notification parameter related_id: Related tag id
ws_api.onRecvModfiyDataBin(value)	object	Obtain the hexadecimal source code of all data modified by Tag, Effence, and Group
ws_api.onRecvAppendInfo(result)	object	Additional data, please refer to related documents for details
ws_api.onRecvAppendInfoBin(result)	object	Obtain the hexadecimal source code of the additional data
ws_api.onRecvPersonInfo(result)	object	Process the personnel related information map_infos: Map information, including map id, map name of regname, and the tag information in the map map_num: Map number tag_online_total: Total number of online people for tags in the system tag_total: Total number of people for the tag
ws_api.onRecvPersonInfoBin(result)	object	Obtain the hexadecimal source code of the personnel related information
ws_api.onRecvErrorInfo(result)	string	Send error information

It is required to call ws_api.RequireBasicInfo(url) first to apply for the basic information, so that the above interface can work properly.

2) Call the method RequireExtraInfo to open the WebSocket for obtaining other information, including Anchor data, as follows:

Method Name	Parameter Type	Description
ws_api.onRecvBaseStData (result)	object	Anchor data id: Anchor ID regid: Map ID state: Anchor state (1: online 0: offline)

		x/y/z: x/y/z value of the Anchor
ws_api.onRecvBaseStDataBin (result)	Object	Obtain the hexadecimal source code of the Anchor data

It is required to call ws_api.RequireExtraInfo(url) first to apply for other information, so that the above interface can work properly.

3) Call method RequireControlInfo to open the WebSocket for obtaining control information, including the electronic fence alarm switch, no accompany alarm switch, arraign alarm switch, electronic roll call switch, video linkage on/off, and sending vibration buzzer commands.

Function Name	Method Name	Parameter Type	Description
Receive the status of WebSocket switch	ws_api.onRecvWebScoketSwitchBack (result)	object	type_conf: Function configuration items, including effence (efence switch), noaccompany (no accompany switch), rollcall (electronic roll call switch), arraign (arraign switch) vibrate (vibration buzzer switch) type_value: Function configuration item value, including enable or disable
Receive video linkage data	ws_api.onRecvVideoChange(result)	object	tagid: Tag id ip: Camera IP port: Camera port number user: Login user name pwd: Login password success: true or false type: 1(Hikvision), 2 (Tiandy), 3 (Dahua), or 4 (Uniview) model: 1 (main code stream) or 2 (sub code stream)
Request efence alarm switch	ws_api.Send2WS.RequsetSwitchAlarm(state)	state	The state value has two states (true, false) true: Enable the efence switch false: Disable the efence switch
Receive the status of efence switch	ws_api.onRecvWebScoketSwitc	result	result: Include enable or disable enable: Enable disable: Disable

	hBack (result)		
Request no accompany alarm switch	ws_api.Send2WS.RequsetSwitchNoAccompany(state)	state	The value of state has two states (true, false) true: Enable the no accompany alarm switch false: Disable the no accompany alarm switch
Receive the status of no accompany switch	ws_api.onRecvWebScoketSwitchBack (result)	result	result: Include enable or disable enable: Enable disable: Disable
Request arraign monitoring alarm switch	ws_api.Send2WS.RequsetSwitchArraign(state)	state	The value of state has two states (true, false) true: Enable the arraign monitoring alarm switch false: Disable the arraign monitoring alarm switch
Receive the status of arraign monitoring switch	ws_api.onRecvWebScoketSwitchBack (result)	result	result: Include enable or disable enable: Enable disable: Disable
Request electronic roll call switch	ws_api.Send2WS.RequsetSwitchRollCall(state)	state	The value of state has two states (true, false) true: Enable the electronic roll call switch false: Disable the electronic roll call switch
Receive the status of electronic roll call switch	ws_api.onRecvWebScoketSwitchBack (result)	result	result: Include enable or disable enable: Enable disable: Disable
Request to open a tab's video linkage	ws_api.Send2WS.RequsetVedioOpen(tagid)	tagid	tagid: Tag id
Request to close a tab's video linkage	ws_api.Send2WS.RequsetVideoClose(tagid)	tagid	tagid: Tag id

Request a tag's temporary disarming function	ws_api.Send2WS.RequsetWithdrawUpdateReq(tagid, time)	Tagid time	tagid: Tag id time: Set the time of a tag's temporary disarming function in timestamp format, e.g. 1519982010
Request tag vibration buzzer	ws_api.Send2WS.RequsetTagShakeBuzzReq(conf_type, conf_value, tagid)	conf_type conf_value tagid	conf_type: tagvibrateandshake A fixed value. You can set a variable to save the value before sending the request. For details, please see the example in the fifth part of the demo. conf_value: enable A fixed value. You can set a variable to save the value before sending the request. For details, please see the example in the fifth part of the demo tagid: Tag id that needs to set the temporary disarming

4) Obtain version SDK

Method Name	Description	Response Mode
ws_api.getVersionMajor()	Obtain the major SDK version number of JS	Click "Connect" to view in the browser's console
ws_api.getVersionMinor()	Obtain the minor SDK version number of JS	Click "Connect" to view in the browser's console

It is required to call ws_api.RequireControlInfo(url) first to apply for other information, so that the above interface can work properly.

5. Demo Description

Demo provides an example of calling an api function.

1. Obtain the relevant data:

1） Input the address and port number of WebSocket server in the WebSocket server address input field (as shown in Figure 1).

(Figure 1)

2） Click the "Connect" button

3） Check box of whether the tag is 64-bit (default to 32-bit) (Figure 2)

(Figure 2)

4） View the log of all the information obtained from the console. (As shown in Figure 3)

(Figure 3)

Note: The above operations are all in the /demo/index.html page.

2. Send control commands:

1） A button to enable and disable the video linkage is added to the index.html page under the demo folder, and the switch can be started and stopped by calling the clicking event.

```
<button class="btn btn-primary" type="button" onclick="VedioOpenBtnClick()">请求视频联动</button>
<button class="btn btn-primary" type="button" onclick="VedioCloseBtnClick()">关闭视频请求</button>
<button class="btn btn-primary" type="button" onclick="TagShakeBuzzBtnClick()">标签振动蜂鸣</button>
```

For example: If you need to enable the video linkage through the button, call the clicking event VedioOpenBtnClick(tagid) and introduce the tag id.

If you need to disable the video linkage through the button, call the clicking event VedioCloseBtnClick(tagid) and introduce the tag id.

Index.js file:

```
window.VedioOpenBtnClick = function () { //打开视频联动
    var ws_api = window.LOCALSENSE.WEBSOCKET_API;
    ws_api.Send2WS_RequsetVedioOpen(g_filter_tag);
};
window.VedioCloseBtnClick = function () { //关闭视频联动
    var ws_api = window.LOCALSENSE.WEBSOCKET_API;
    ws_api.Send2WS_RequsetVideoClose(g_filter_tag);
};
window.TagShakeBuzzBtnClick = function () { //标签振动蜂鸣
    var ws_api = window.LOCALSENSE.WEBSOCKET_API;
    var conf_type = "tagvibrateandshake"; //标签振动蜂鸣类型
    var conf_value = "enable"; //启用
    ws_api.Send2WS_RequsetTagShakeBuzzReq(conf_type, conf_value, g_filter_tag);
};
```

3. Close control command

1） Click the "Disconnect" button. (as shown in Figure 1)

2） Call the CloseWebsocket function in index.js in the demo folder to close the WebSocket connection.

```
//客户端断开的请求
function CloseWebsocket(){
  var ws_api = window.LOCALSENSE.WEBSOCKET_API;
  ws_api.RejectBasicInfo();
  ws_api.RejectControlInfo();
  ws_api.RejectExtraInfo();

};
```