# CarTalk V2X
# Web Application & Service Platform

## Developer’s Manual
## Version 1

Internet Education and Research Laboratory (intERLab)
Asian Institute of Technology (AIT)

## Table of Contents

## Introduction

The CarTalk V2X Web Application as a Service is a platform for developing V2X web applications running on the V2X communication system. This developer manual is intended for developers wishing to create and modify a web application on The CarTalk V2X Platform. Programmers wishing to create or modify V2X Web App are expected to know and understand JavaScript as well as the basics of network communications. This manual will provide instructions and examples for using APIs.

## API

This manual describes how to configure, connect to, and interact with the API. Examples are showed in JavaScript codes which can be written directly into an HTML page.

## Neighbor-related API

### Broadcast

API call: **/api/broadcast**

This API is used to broadcast data. The data will be broadcast to all the other node in the network. Input values can be separated by comma e.g. Value1,Value2,Value3, and so on. An output will be {success:true} if data is successfully sent or {success:false} if data is unsuccessfully sent.

- Requirements: JQuery

```
function broadcast(broadcastData) {

  $.ajax({

      method: 'POST',

      url: '/api/broadcast',

      data: {

        data: “broadcastData”

            },
        })
    .done(function(msg) {
            alert( msg);
        });
}
```

## List

API call: **/api/list**

This API is used to retrieve a list of neighbors in the network to see the number of available nodes in current network. It shows only the neighbors list (IP Address) with timestamp.

- Requirements: JQuery

```
$.ajax({
    method: 'GET',
    url: '/api/list',
    data: {}
    })
.done(function(result) {
    var json = jQuery.parseJSON(result)
        console.log(json)
    /*you can process the Json output e.g. Place it on the Map*/
    });
```

The above API will output:

{"olsrcast.recv.10.100.42.10.txt":[1479660316],"olsrcast.recv.10.100.43.10.txt":[1479660317],"olsrcast.recv.10.100.44.10.txt":[1473222667]}

## ListAllData

API call: **/api/listAllData**

This API is used to retrieve a list of neighbors in the network to see the number of available nodes and all data in the network. It shows the neighbors list (IP Address) with timestamp and all data that retrieved from broadcast.

- Requirements: JQuery

```
$.ajax({
  method: 'GET',
  url: '/api/listAllData',
  data: {}
  })
.done(function(result) {
  var json = jQuery.parseJSON(result)
      console.log(json)
  });
```

The above API will output:

{"olsrcast.recv.10.100.42.10.txt":["1479660584","14.0775715","100.6129275","","","","","10.100.42.10_","1"],"olsrcast.recv.10.100.43.10.txt":["1479660584","14.0775715","100.6129275","","","","","10.100.43.10_","1"],"olsrcast.recv.10.100.44.10.txt":["1473222702","14.077402100175885","100.61301074845987","","","","",null,"1"],"olsrcast.recv.192.168.1.196.txt":["1472026897","14.077650909352748","100.61290589291563","","","","","10.100.42.10_","1"]}

## GPS Positioning API

### getCurrentPosition

A location (latitude/longitude) of a user can be used in an application that require the current position of the user such as a navigation app. This example will retrieve the user's position(latitude/longitude) and broadcast using the above API.

- Requirements: JQuery

```
window.onload = function() {

  var startPos;

  var geoOptions = {

    enableHighAccuracy: true

    }

  var geoSuccess = function(position) {

         broadcast(position.coords.latitude+','+position.coords.longitude;)

    };

  var geoError = function(error) {

        console.log('Error occuerd.Errorcode:' + error.code);

    };

    navigator.geolocation.getCurrentPosition(geoSuccess, geoError, geoOptions);

};
```

## Vehicular Cloud and Web Socket API

### Store

API call: **/api/store**

This API is used to save data to a vehicular cloud. The data will be save in the vehicular cloud storage which is a local cloud established in a group of vehicles.

```
<form name="form1"  id="form1" action="/api/store" method="post" enctype="multipart/form-data">

        <!-- You can use one ore more file input. -->

        <input type="file" name="file[]">

        <input type="file" name="file[]">

        <input type="file" name="file[]">

</form>
```

### Retrieve

API call: **/api/retrieve**

To retrieve data from the vehicular cloud, you must use this API with Web Socket API. The web socket APIs are following.

### Web Socket API

| Web Socket API | Description |
|---|---|
| **Onopen** | Calls when a connection is opened |
| **Onmessage** | Message is received |
| **Onclose** | Server connection has been closed |
| **Onerror** | Error occurred |
| **Send** | Send message or data |

How to use these APIs is described in the next section.

## API with Example scenarios

This example is retrieving files from cloud after you save files to the vehicular cloud. The scenario is to retrieve image from the cloud and show in an HTML page.

```
var API_URL = 'ws://' + window.location.hostname + ':9001/api/retrieve';


try {

    socket = new WebSocket(API_URL);

    console.log('WebSocket - status ' + socket.readyState);

  socket.onopen = function(msg) {

    if (this.readyState == 1) {

            console.log("We are now connected to websocket server. readyState = " + this.readyStat
e);

      var myVar = setInterval(function() {

                updater()

            }, 1000); //1 second Loop

        } else {}

    };

  //Message received from websocket server

    socket.onmessage = function(msg) {

    try {

            json = JSON.parse(msg.data)

     /*json can be processed */
     /*e.g. listing all image in an HTML */

      var body = '';


      for (var key in json) {

                body = body + '<img src=/shared/' + key + ' width=100> ' + json[key] + '<br>';

            }

          console.log(body)

        } catch (err) {}

    };

  //Connection closed

    socket.onclose = function(msg) {

        console.log("Disconnected - status " + this.readyState);

        alert('Server has closed! Refresh again');

    };

    socket.onerror = function() {

        console.log("Some error");

        alert('Server error! Refresh again');

    }

} catch (ex) {

    console.log('Some exception : ' + ex);

    alert('Error connecting to server!');

}
```

## Checking for 3g Connection

### isConnected

API call: **/api/isConnected**

This API is to check a cellular network(3G/4G). If the cellular network is connected, it will return **1** otherwise return **0.**

```
$.ajax({
    method: 'GET',
    url: '/api/isConnected',
    data: {}
    })
  .done(function(result) {
        var json = jQuery.parseJSON(result)
        console.log(json)
    /*1 for connected */


    });
```

If the internet connection is established, we can use HTTP GET and POST method to request and submit data to a server.

## API Summary

| Neighbor-related API | Description | Target | Method | Returns | Requires |
|---|---|---|---|---|---|
| broadcast | Broadcast data, e.g user Lat/Lon | /api/broadcast | Post | JSON | JSON |
| list | List all neighbor (Only IP Address) | /api/list | Get | JSON | N/A |
| listAllData | List all neighbor (IP Address and Data, eg. Lat/Lon) | /api/listAllData | Get | JSON | N/A |

Table 1 API for Neighbor-related

| Vehicular Cloud API | Description | Target | Method | Returns |
|---|---|---|---|---|
| store | Store data | /api/store | Post | JSON |
| retrieve | Retrieve data | /api/retrieve | Get | JSON |

Table 2 API for Vehicular Cloud

| GPS Positioning API | Description | Returns |
|---|---|---|
| getCurrentPosition | Get Position | position.coords.latitude<br>position.coords.longitude |

Table 3 API for GPS Positioning

| Web Socket API | Description | Returns | Requires |
|---|---|---|---|
| onopen | Calls when a connection is opened | Boolean | |
| onmessage | Message is received | JSON | |
| onclose | Server connection has been closed | | |
| onerror | Error occured | | |
| send | Send message | | JSON |

Table 4 API for Web Socket

| 3G Cellular Status Check | Description | Returns | Requires |
|---|---|---|---|
| isConnected | Check if cellular data is connected | Boolean | |

Table 5 API for checking 3G/4G connection

| Web Service API (when 3G is available) | Description | Returns | Requires |
|---|---|---|---|
| get | HTTP GET | JSON (content) | JSON (url) |
| post | HTTP POST | JSON (content) | JSON (url, postdata) |

Table 6 API for Web Service when the 3G/4G connection is available

## Acknowledgement

This work is fully funded by National Science and Technology Development Agency(NSTDA). For more information about CatTalk V2X Web Application & Service Platform is available at http://interlab.ait.asia/CarTalkWaaS/