JavaScript SDK for AT&T Enhanced WebRTC

Web App Migration Guide from old versions to v1.0.0-rc15.0

Summary

This document describes how to migrate your existing web app using our **Node DHS** for AT&T OAuth services and SDK versions older than v1.0.0-rc15.0.

Audience

Developers who are currently using our **Node DHS** for AT&T OAuth Services or **ATT.rtc.configure** method to configure their own OAuth Services.

Why

Core JavaScript WebRTC functionality is de-coupled from DHS functionality. From this release forward, SDK does not need **configure** method to initialize itself.

Migration Cases

Typically, client-side JavaScript code needs to do 3 things when interacting with DHS: Obtain Configuration, Obtain Access Token, and Create E911 Id.

If you are using this	Then you will need to do this	Reasons	Migration Case
Hosting your web application and AT&T Node DHS in separate servers	Replace AT&T Node DHS with the AT&T Node Sample Server	Deprecated DHS Services	A
Hosting your web application and AT&T Node DHS in a single server	Replace AT&T Node DHS with the AT&T Node Sample Server and move your web application to Node Sample Server	Deprecated DHS Services	A & B
ATT.rtc.dhs.createAccessToken	Replace it with XMLHttpRequest POST Method	Removed from library	C
ATT.rtc.dhs.createE911Id	Replace it with XMLHttpRequest POST Method	Removed from library	D
ATT.rtc.asscociateAccessToken	Update it to ATT.rtc.Phone.associateAccessToken	Removed from library	E
ATT.rtc.configure	Remove it	Removed from library	F
ATT.rtc.getEWebRTCDomain	Replace it with a) your EWebRTC domain name or b) or see Migration Case G.	Removed from library	G
If you want to use this	**Then you will need to do this**	**Reasons**	**Migration Case**
New AT&T DHS Configuration Information (Optional)	Add XMLHttpRequest GET Method	New DHS Service	H

Migration Case A

Replace your current AT&T Node DHS server with the new Node Sample server.

1. Download the new Node Sample package from https://github.com/attdevsupport/ewebrtc-sdk.
2. Update the new /node-sample/package.json with your own DHS ports, CORS Domain, and Sandbox data set. Please refer to your existing /node-dhs/package.json for the referenced sections. Ensure that these entries are top-level children in the new package.json.

```
"http_port": 9000, //Default or replace with your DHS port
"https_port": 9001, //Default or replace with your DHS port
"cert_file": "sample.cert", //Default or replace with your SSL filename
"key_file": "sample.key", //Default or replace with your SSL key filename
"logs_dir": "logs", //Default or replace with your log directory name
"cors_domains": ["*"] ////Default or replace with your CORS Domain list

"sandbox": {
    "api_endpoint": "https://api.att.com", // Default
    "ewebrtc_uri": "/RTC/v1", //Default
    "app_key": "your_app_key",
    "app_secret": "your_app_secret",
    "oauth_callback": "your_oauth_callback_url",
    "app_token_url": "your_dhs_url/tokens",
    "app_e911id_url": "your_dhs_url/e911ids",
    "virtual_numbers_pool": ["Your VTN Pool"]
    "ewebrtc_domain": "your_ewebrtc_domain"
},
```

3. Install the Node dependencies for the new Node Sample server.

```
$ cd /your_path/node-sample
$ npm install
```

4. Start the Node Sample server

```
$ npm start
```

The new AT&T Node Sample server will now be used for AT&T OAuth Services.

Migration Case B

Migrate your existing web app to the new Node Sample server.

1. Copy your web app files into the corresponding /node-sample/public root and sub directories.

Migration Case C

Replace ATT.rtc.DHS.createAccessToken with XMLHttpRequest POST Method.

1. Locate ATT.rtc.DHS.createAccessToken reference in your web application.

```
ATT.rtc.dhs.createAccessToken ({
    app_scope: appScope,
    auth_code: authCode,
    success: success,
    error: error
});
```

2. Replace it with

```
var xhrToken = new XMLHttpRequest();
xhrToken.open("POST", "your_dhs_url /tokens");
xhrToken.setRequestHeader("Content-Type", "application/json");
xhrToken.onreadystatechange = function() {
    if (xhrToken.readyState == 4) {
        if (xhrToken.status == 200) {
            sucess(JSON.parse(xhrToken.responseText));
        } else {
            error(xhrToken.responseText);
        }
    }
}
xhrToken.send(JSON.stringify({app_scope: appScope, auth_code: authCode}));
```

Migration Case D

Replace ATT.rtc.DHS.createE911Id with a XMLHttpRequest POST Method.

1. Locate ATT.rtc.DHS.createE911Id reference in your web application.

```
ATT.rtc.dhs.createE911Id ({
    token:  accessToken,
    address:  e911Address,
    is_confirmed:  isConfirmed,
    success:  success,
    error: error
});
```

2. Replace it with

```
var xhrE911 = new XMLHttpRequest();
xhrE911.open("POST", "your_dhs_url/e911ids");
xhrE911.setRequestHeader("Content-Type", "application/json");
xhrE911.onreadystatechange = function() {
    if (xhrE911.readyState == 4) {
        if (xhrE911.status == 200) {
            success(JSON.parse(xhrE911.responseText));
        } else {
            error(xhrE911.responseText);
        }
    }
}
xhrE911.send(JSON.stringify({
    token: accessToken,
    address: e911Address,
    is_confirmed: isConfirmed
}));
```

Migration Case E

Replace ATT.rtc.associateAccessToken with phone.associateAccessToken.

1. Locate ATT.rtc.associateAccessToken reference in your web application.

```
ATT.rtc.associateAccessToken ({
    userID:  userID,
    token:  accessToken,
    success:  success,
    error: error
});
```

2. Replace it with

```
// var phone has been declared previously

phone.associateAccessToken ({
    userID:  userID,
    token:  accessToken,
    success:  success,
    error: error
});
```

Migration Case F

Remove ATT.rtc.configure method.

1. Locate ATT.rtc.configure reference in your web application and remove it.

```
ATT.rtc.configure ({
    ewebrtc_domain: "your_domain",
    api_endpoint: "https://api.att.com"
});
```

Migration Case G

Replace ATT.rtc.getEWebRTCDomain method.

1. Locate ATT.rtc.getEWebRTCDomain reference in your web application.

```
var eWebRTCDomain = ATT.rtc.getEWebRTCDoamin();
```

2. ATT.rtc.getEWebRTCDomain reference in your web application and remove it.

```
//Option 1: Use actual value
var eWebRTCDomain = "your_ewebrtc_domain";

//Option 2: Use DHS Configuration information. See Migration Case H
var eWebRTCDomain = config.your_ewebrtc_domain;
```

Migration Case H (Optional)

Add XMLHttpRequest GET Method to get DHS Configuration information

1. Add XMLHttpRequest GET

```
var config

var xhrConfig = new XMLHttpRequest();
xhrConfig.open("GET", "your_dhs_url/config");
xhrConfig.onreadystatechange = function() {
    if (xhrE911.readyState == 4) {
        if (xhrConfig.status == 200) {
            config = (JSON.parse(xhrConfig.responseText));
        } else {
            console.log(xhrE9Config.responseText)
        }
    }
}
xhrConfig.send();
```

On success, the DHS will response back with the following DHS information in a JSON object. A sample object is show below:

```
{
api_endpoint: "https://api.att.com",
authorize_uri: "/oauth/v4/authorize",
oauth_callback: "your_oauth_callback_url",
app_key: "your_app_key",
ewebrtc_domain: "your_ewebrtc_domain'",
virtual_numbers_pool: array(your_vtn_list),
app_e911id_url: "your_dhs_url/e911ids",
app_token_url: "your_dhs_url/tokens",
info: {
    api_env: "1.0.1",
    ewebrtc_uri: "/RTC/v1",
    dhs_name: "att.dhs",
    dhs_platform: "Node",
    dhs_version: "1.0.0",
    e911id_uri: "/emergencyServices/v1/e911Locations",
    token_uri: "/oauth/v4/token",
    scope_map: {
        ACCOUNT_ID: "WEBRTC",
        E911: "EMERGENCYSERVICES",
        MOBILE_NUMBER: "WEBRTCMOBILE",
        VIRTUAL_NUMBER: "WEBRTC"
        }
    }
}
```

Sample Code and Tutorial

Code walk-through is provided in the tutorial http://attdevsupport.github.io/ewebrtc-sdk/tutorial/ under section **Client Code Snippets**.