ClickFORMS Online Integration Guide

The following guidelines elaborate on the methods and the steps a prospective integrator should follow in order to integrate **Bradford ClickFORMS Online** into their services. A cloud or desktop application of the company will be communicating with the **Bradford Public Label API** to create sessions within which the ClickFORMS Online will be operated and sent data to.

* Version 2 -

**Private Label API Subscription**

The first step of the integration is to subscribe into the Bradford White Label API and receive the **private API key** and the **integrator-id** assigned to your company or to the application. This API key will be representing your company or the application you have requested the subscription for. It will be used by you to sign JWTs which are required when making Bradford Private Label API calls. The signature of this key will be used to monitor the subscription. The subscriber is expected to store and utilize the key within a secure perimeter when signing security tokens for the API’s use.

Please contact us for more information on subscribing into Bradford Private Label API.

**Creating a Session**

A POST request to the following end-point creates a session of ClickFORMS Online that would last a duration you may specify. (Please jump to the “Bradford WSMS Client Library” section for easy integration steps in Node JS)

**URL**: https://clickforms.appraisalworld.com/wsms/sessions

The request should contain following headers.

**HTTP HEADERS**

“Content-Type” : “application-json”

“Authorization” : “Bearer”<SPACE>JWT

**The JWT**

The Public Label API expects a JWT security token in the “Authorization” header of every call. The JWT should contain the required fields in its payload and should contain “kid” field in its header. The asymmetric RS256 algorithm should be used to sign the JWT with the API key introduced in the first step.

PAYLOAD:

{

iat : <standard iat>

iss : <custom string>

userId : <user whom the session will be created for>

groupId : <OPTIONAL group of the user>

email : <OPTIONAL user info (email)>

productId : 101

jti : <standard jti>

}

HEADER:

“kid” : <integrator-id>

EXPIRATION:

A time which does not exceed 24hours starting from the time of issuance.

**End-point return value**

The end-point will return the HTTP code 201 if the session is created. Further, it will return the following JSON structure as the response data.

{

sessionId : <id of the session>

url : <url of the ClickFORMS Online for the session>

}

The URL is not sharable and is restricted to be used only one time. The session is considered to have been consumed by the user when it was opened for the first time. The original user (who opened the url first) can continue to use the url in the same browser any number of times until the session meets the time of its designated expiry.

**The Integration Sandbox**

The Integration Sandbox is a sample integrator account which can be used for testing. You can contact us and obtain a copy of the Sandbox private key for early testing.

**Bradford WSMS Client Libraries**

The Bradford WSMS Client Libraries enables the integrator a convenient and faster way of integration.

**For Node JS:**

Package name - bradford-wsms-client-library

Url - https://www.npmjs.com/package/bradford-wsms-client-library/

**Steps**

* Install the package

Execute the following NPM command –

npm install bradford-wsms-client-library

Or, add the following line to the package.json under “dependancies”

“bradford-wsms-client-library”: “1.0.0”

* Import the bradford-wsms-client-library module in your project

const wsms = require(‘bradford-wsms-client-library’)

* Use the createSession imported function to create the session

**Arguments**:

Integrator-id : String

Claims : JS Object

Exp. Hours : Number

Private Key : String (Base 64)

Options : JS Object (optional)

Data : JS Object (optional)

**Options:**

randomJti : A random jti will automatically be created if

true. The claims should include jti if false

The method returns a promise.

* Sample Code

var privateKey = fs.readFileSync('./private\_key.pem');

var claims = {

iss: 'sample-integrator',

userId: 'user-1',

teamId: 0,

orderId: 'order-1',

email: 'user-1@example.com',

productId: 101,

}

var pKey = Buffer.from(privateKey).toString('base64');

var options = {randomJti: true};

var data = {hello: ‘world’}

wsms.createSession('sample-id', claims, 3, pKey, options, data)

.then(function(){

console.log(‘Session Created’);

},

function(e) {

console.log(e);

})

* Security

The loading of the private key from a disk file in the above example is for demonstration purposes only. It is advisable to use environment variables to supply the key to the container running your app.

**For C# .NET**

Package name - bradford-wsms-client-library

Url -

https://github.com/Bradtech302/wsms-client-libraries/tree/master/c-sharp.net

**Steps**

* Download the package
* Install Dependencies : Install the following NuGet Packages

1. Install-Package jose-jwt –Version 2.4.0
2. Install-Package BouncyCastle –Version 1.8.4

* Instantiate the WSMSClient class and use the createSession method.

**Arguments**:

Integrator-id : Number

Claims : List<Claim>

Key Path : String

Exp. hours : Number

* Sample Code

This is a sample ASP.NET code which executes the WSMS Client Library

int iid = 998;

var claims = new List<Claim>();

claims.Add(new Claim("iat", DateTime.Now.ToString("M/d/yyyy")));

claims.Add(new Claim("iss", "develop"));

claims.Add(new Claim("userId", "111"));

claims.Add(new Claim("userType", "true"));

claims.Add(new Claim("productId", "101"));

claims.Add(new Claim("orgName", "testOrg"));

claims.Add(new Claim("jti", Guid.NewGuid().ToString()));

int expHours = 24;

WSMSClient libObj = new WSMSClient();

txtResponse.Text = libObj.CreateSession(iid, @"x:\sandbox\_key.pem",

claims, expHours).ToString();

**For PHP**

Package name - wsmsclient.php

Url -

<https://github.com/Bradtech302/wsms-client-libraries/tree/master/php>

**Steps**

* Download the package
* Install dependencies with “Composer” in the application path.

composer require firebase/php-jwt

* Use the createSession static method of the WSMSClient class

**Arguments**:

Integrator-id : Number

productId : Number

userId : String

Exp. hours : Number

groupId : String

keyPath : String

* Sample Code

require 'wsmsclient.php';

$kid = 1001;

$productId = 102;

$userId = 'developer';

$groupId = 'developers'; //Optional - can be an empty string

$expHours = 1;

$keyPath = 'private\_key.pem';

vardump( WSMSClient::createSession($iid, $productId, $userId, $groupId,

$expHours, $keyPath));