# AuthToken V2
HELP DOCUMENT

Home - OpenEdge
Global Payments Inc. [US] https://www.openedgepayment.com
openedge
DEVELOPER SOLUTIONS
EMV RESOURCES
MERCH
We're OpenEdge
The innovative arm of Global Paym
and the result of the merger betwe
integrated payments pioneers Pay
and Accelerated Payment Technol
openedge
LEARN

# TABLE OF CONTENTS

## GENERATING AUTHTOKEN

The format is based on **JWT** (JSON Web Token). For more information on JWT please refer to https://jwt.io/introduction/

As mentioned above, AuthToken will be JWT based, so let's understand the different parts: header, payload, and signature.

➢ Header

The header consists of two parts, the type of the token and hashing algorithm. In our case, the type is JWT and hashing algorithm is HMAC-SHA256. So, the header looks like –

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

➢ Payload

The payload contains the claims, which essentially are entities of user with metadata. There are three types of claims: *reserved*, *public*, and *private* claims. For our case, we have private claims since it is customized for our needs. The private claim consists of four fields:

- *"type"* The name of the validation type. Enter "AuthTokenV2".
- *"region"* The region of the transaction. The currently available values are:
  - "US"
  - "CA"
  - "AU"
  - "NZ"
- *"account_credential"* A credential distributed to integrating partners by OpenEdge onboarding staff. Please contact OpenEdge Developer Services to obtain an 'account_credential'.
- *"ts"* A millisecond-scale timestamp

The private claim looks like this:

```
{
  "type" = "AuthTokenV2",
  "region" = "US",
  "account_credential" =  "encryptedAccountCredential",
  "ts": 1521839766066
}
```

➢ Signature

It is the generated value by Hashing the encoded header, encoded payload with secret using algorithm specified in header. In our case, the secret will be **ApiSecret** —distributed to integrating partners by OpenEdge onboarding staff. Please contact OpenEdge Developer Services to obtain an 'ApiSecret'— and the hashing algorithm **HMAC-SHA256**. The signature will be created as follows:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  ApiSecret)
```

Create the AuthToken by concatenating the encoded header, payload, and signature. The AuthToken would look like xxx.yyy.zzz
Where –

xxx is the encoded header using UTF-8 charset
yyy is the encoded payload using UTF-8 charset
zzz is the signature generated by hashing encoded header and payload with secret

Now, let's generate the AuthToken with below sample values

type = AuthTokenV2  (This must be  constant)
account_credential = Qsfe9sGnqj6vuDhsoDezPuMRg7awDYmfSRs5UCM
region = US
ApiSecret = W9dlVYgcfRkdIckJkNdFydiJdf3Kdge2

## JAVA Sample Code

```
/**
 ** AuthToken GENERATION SAMPLE CODE FOR REFERENCE - ONLY.
 ** Exception Handling, Data Validations are NOT considered in this sample
 ** json-simple JAR is required for successful BUILD.
 ** <h2>GenerateToken.java</h2>
 **/


package authtoken;

import java.io.UnsupportedEncodingException;
import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import org.json.simple.JSONObject;

public class GenerateToken {
        private final static Charset UTF8_CHARSET = Charset.forName("UTF-8");

        public static void main(String args[])
                        throws NoSuchAlgorithmException, InvalidKeyException,
IllegalStateException, UnsupportedEncodingException {
                // Merchant account details
                String accountCredential= "Qsfe9sGnqj6vuDhsoDezPuMRg7awDYmfSRs5UCM";
                String region= "US";
                String apiSecret = " W9dlVYgcfRkdIckJkNdFydiJdf3Kdge2";

                // Creating JSON object for Header , json-simple JAR is required.
                JSONObject jwtHeaderObj = new JSONObject();
                jwtHeaderObj.put("alg", "HS256");
                jwtHeaderObj.put("typ", "JWT");
                // Base64 encoding of Header
                String jwtHeaderATBase64 = Base64.getUrlEncoder()

        .encodeToString(jwtHeaderObj.toString().getBytes(UTF8_CHARSET));

                // Creating JSON object for Payload
                JSONObject jwtPayloadObj = new JSONObject();
                jwtPayloadObj.put("type", "AuthTokenV2");
                jwtPayloadObj.put("account_credential", accountCredential);
                jwtPayloadObj.put("region", region);
                jwtPayloadObj.put("ts", System.currentTimeMillis());
                // Base64 encoding of Payload
                String jwtPayloadATBase64 = Base64.getUrlEncoder()

        .encodeToString(jwtPayloadObj.toString().getBytes(UTF8_CHARSET));

                // Concatenating encoded Header and Payload with "."
                String jwtMessage = jwtHeaderATBase64 + "." + jwtPayloadATBase64;

                // Create Signature using HMAC-SH256 algorithm and ApiSecret as the
secret
                Mac sha256_HMAC = Mac.getInstance("HmacSHA256");
                SecretKeySpec secret_key = new
SecretKeySpec(String.valueOf(apiSecret).getBytes(), "HmacSHA256");
                sha256_HMAC.init(secret_key);
                String hashSignature = Base64.getUrlEncoder()

        .encodeToString(sha256_HMAC.doFinal(jwtMessage.getBytes(UTF8_CHARSET)));

                /**
                 * Create the JWT AuthToken by concatenating Base64 URL encoded
header,
                 * payload and signature
                 */

                String AuthTokenV2 = jwtHeaderATBase64 + "." + jwtPayloadATBase64 +
"." + hashSignature;
                System.out.println("Generated AuthtokenV2: " + AuthTokenV2);

        }
}
```

**Output**

```
Generated AuthtokenV2:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtaWQiOjkwNjAwMDA1NDEyMywidHlwZSI6IkFVVEhUT0
tFTiIsInRpZCI6MTIzNDU2NzgsInRzIjoxNTM2MTYzMjAzOTUxfQ==.Iuc6gVH_yTrZxwXYcV1AVp7j9xFA
q5hoJhxoGRW_iXw=
```

## JavaScript Sample Code

```
<!--
  - AuthToken GENERATION SAMPLE CODE .
  - AuthToken GENERATION SAMPLE CODE FOR REFERENCE
  - Exception Handling, Input Data Validations are NOT considered in this sample.
  - Please NOTE the libraries were loaded in-line just for reference.
 -->

<!-- Include of hmac-sha256 library, App team has to ensure compliance and any other aspect while using
external library-->
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.2/rollups/hmac-sha256.js"></script>
```

```
<!--  Include of enc-base64 library, App team has to ensure compliance and any other aspect while using
external library -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.2/components/enc-base64-min.js"></script>

<script>

        /**
         * This JS method takes accountCredential,region and apiSecret to generate sample AuthToken
         */

function generateAuthToken(accountCredential, region, apiSecret) {

  if (accountCredential == null || accountCredential == "" || region == null ||
     region == "" || apiSecret== null || apiSecret== "") {
     throw "MANDATORY INPUT DATA IS EMPTY";
  }

  try {
          var encodedHeaderJSON = encodeURL(generateEncodedJSONHeader());
        var encodedPayLoadJSON = encodeURL(generateEncodedPayloadJSON(accountCredential,
       region));
     var encodedSignature = encodeURL(generateHashSignature(encodedHeaderJSON,
       encodedPayLoadJSON, apiSecret));

     return encodedHeaderJSON + "." + encodedPayLoadJSON + "." +
       encodedSignature;

  } catch (err) {
     var errorMsg = "Error While Generating Auth-Token : " + err;
     throw errorMsg;
  }
}

// Generates JWT Signature
function generateHashSignature(encodedHeaderJSON, encodedPayLoadJSON, apiSecret) {
  var data = encodedHeaderJSON + "." + encodedPayLoadJSON;
  var hash = CryptoJS.HmacSHA256(data, apiSecret);
  return CryptoJS.enc.Base64.stringify(hash);

}
// Generates JWT Payload
function generateEncodedPayloadJSON(accountCredential, region) {
  var payLoadObj = new Object();
  payLoadObj.accountCredential = accountCredential;
  payLoadObj.region = region;
  payLoadObj.type = "AuthTokenV2";
  payLoadObj.ts = new Date().getTime();

  return CryptoJS.enc.Base64.stringify(CryptoJS.enc.Utf8.parse(JSON
     .stringify(payLoadObj)));
}

// Generates JWT Header
function generateEncodedJSONHeader() {
   var headerObj = new Object();
   headerObj.alg = "HS256";
           headerObj.typ = "JWT";
   return CryptoJS.enc.Base64.stringify(CryptoJS.enc.Utf8.parse(JSON
      .stringify(headerObj)));
}

 function encodeURL(encodedSource){
    //Making Base64 URL Safe
           encodedSource = encodedSource.replace(/\+/g, '-');
           encodedSource = encodedSource.replace(/\//g, '_');
           return encodedSource;
}

</script>
```

## .Net Sample Code

```
/** ** AuthToken GENERATION SAMPLE CODE FOR REFERENCE.
 ** Exception Handling, Data Validations are NOT considered in this sample
 ** Newtonsoft.Json is required for successful BUILD.
 **/


using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using Newtonsoft.Json;

namespace JWTSharpConsoleApplication
{
    public class Program
    {
        static readonly char[] padding = { '=' };

        public static void Main(string[] args)
        {
            string accountCredential= "Qsfe9sGnqj6vuDhsoDezPuMRg7awDYmfSRs5UCM";
            string region= "US";
            string apiSecret= "secret";
            try
            {
                generateEncodedAuthToken(accountCredential, region, apiSecret);
            }
            catch (Exception ex)
            {
                Console.Out.WriteLine("ex = {0}", ex);
            }
        }

        public static void generateEncodedAuthToken(string accountCredential,
string region, string apiSecret)
        {

            if ( accountCredential== null || region== null
||string.IsNullOrEmpty(apiSecret))
            {
                throw new Exception("MANDATORY INPUT DATA IS EMPTY");
            }

            try
            {
                string encodedHeaderJSON = generateEncodedJSONHeader();
                string encodedPayLoadJSON =
generateEncodedPayloadJSON(accountCredential, region);
                string encodedSignature = generateHashSignature(encodedHeaderJSON,
encodedPayLoadJSON, apiSecret);

                string GeneratedAuthTokenV2 = encodedHeaderJSON + "." +
encodedPayLoadJSON + "." + encodedSignature;

                System.Console.WriteLine(string.Format("Response: {0}",
GeneratedAuthTokenV2));;
                return ;
            }
            catch (Exception err)
            {
                Console.Out.WriteLine("Error While Generating AuthTokenV2");
            }
        }

        public static string generateHashSignature(string encodedHeaderJSON, string
encodedPayLoadJSON, string apiSecret)
        {
            string data = encodedHeaderJSON + "." + encodedPayLoadJSON;
            byte[] hash = new
HMACSHA256(System.Text.Encoding.UTF8.GetBytes(apiSecret)).ComputeHash(System.Text.E
ncoding.UTF8.GetBytes(data));
            return UrlSafe(Convert.ToBase64String(hash));
        }

        public static string generateEncodedPayloadJSON(string accountCredential,
string region)
        {
            object payLoadObj = new
            {
                account_credential= accountCredential,
                region = region,
                type = "AuthTokenV2",
                ts = new DateTimeOffset(DateTime.UtcNow).ToUnixTimeMilliseconds()
            };

            return objectToString(payLoadObj);
        }

        public static string UrlSafe(string input)
        {
            return input.Replace('+', '-').Replace('/', '_');
        }

        public static string objectToString(object obj)
        {
            return
UrlSafe(Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(JsonConvert.Seria
lizeObject(obj))));
        }


        public static string generateEncodedJSONHeader()
        {
            object headerObj = new
            {
                alg = "HS256",
                typ = "JWT"
            };
            return objectToString(headerObj);
        }
    }
}
```

## PHP v7 Sample Code

```
<!--
AuthToken GENERATION SAMPLE CODE FOR REFERENCE.
Exception Handling, Data Validations are NOT considered in this sample.
-->
<?php
function generateAuthToken($accountCredential, $region, $apiSecret) {
    $headerObj = new stdClass();
    $headerObj->alg = "HS256";
    $headerObj->typ = "JWT";
    $headerJSON = base64url_encode(json_encode($headerObj));

    $microseconds = microtime(true);
    $jwtPayloadObj = new stdClass();
    $jwtPayloadObj->account_credential = $accountCredential;
    $jwtPayloadObj->region = $region;
    $jwtPayloadObj->type = "AuthTokenV2";
    $jwtPayloadObj->ts = $microseconds;
    $payloadJSON = base64url_encode(json_encode($jwtPayloadObj));

    $signature = base64url_encode(hash_hmac('sha256',
"{$headerJSON}.{$payloadJSON}", $apiSecret, true));
    echo "Printing Generated AuthToken : \r\n";
    $authToken = "{$headerJSON}.{$payloadJSON}.{$signature}";

    echo $authToken;
}


function base64url_encode($data) {
  return strtr(base64_encode($data), '+/', '-_');
}

$accountCredential = "Qsfe9sGnqj6vuDhsoDezPuMRg7awDYmfSRs5UCM"; //accountCredential
to be input, sample used here
$region = "12345678"; // region to be input, sample used here.
$apiSecret = "secret"; // Sample apiSecret used
generateAuthToken($accountCredential, $region, $apiSecret); // call the function
with required inputs to generate AuthToken
?>
```

# CHANGE HISTORY

| Change History | | | |
|---|---|---|---|
| **Version** | **Date** | **Author** | **Reason for Update** |
| 2.0 | 06/27/2019 | NSC | Initial document distribution. |