**Merchant documentation guide for SFCC REST Microform v2 upgrade**

Contents

[**Step 1. Generate the server-side capture context** 1](#_Toc188979918)

[**Step 2: Decode and Validate Capture Context** 2](#_Toc188979919)

[**Step 3: Add clientLibrary and clientLibraryIntegrity values** 4](#_Toc188979920)

[**Step 4: Load flex IFrame** 5](#_Toc188979921)

[**Step 5: Replace diners-club with dinersclub** 6](#_Toc188979922)

[**Step 6: Create Configurations** 6](#_Toc188979923)

# **Step 1. Generate the server-side capture context**

1. Create a custom preference to add allowed networks for flex microform.

Refer section [**Create Configurations**](#_Create_Configurations)to create required configurations.

Go to **Merchant Tools** > **Site Preferences** > **Custom Preferences** > **Cybersource\_FlexMicroform** and set values for the following parameters:

Set **Cybersource\_AllowedCardNetworks** with the values that are allowed in flex microform service. If no card type is selected it would take **Visa** as default card type.

1. Navigate to below path and add code changes to **createFlexKey()** method

**Path: cartridges/int\_cybs\_sfra\_base/cartridge/scripts/http/payments.js**

var allowedCNetworks = dw.system.Site.getCurrent().getCustomPreferenceValue('Cybersource\_AllowedCardNetworks');

var list =[];

if(empty(allowedCNetworks)){

list.push('VISA'); // if no card networks are added send Visa as default

}else{

for (let i = 0; allowedCNetworks[i]!= null ; i++) {

list.push(allowedCNetworks[i].value);

}

}

var publicKeyRequest = {

'encryptionType': Constants.ENCRYPTION\_TYPE,

'targetOrigins':[

Constants.PROXY\_PREFIX + '://' + request.httpHost

],

'allowedCardNetworks': list,

'clientVersion': Constants.CLIENT\_VERSION

};

1. Create constants for encryption type and client version

**Path: cartridges/int\_cybs\_sfra\_base/cartridge/apiClient/constants.js**

/\* Flex microform constants \*/

ENCRYPTION\_TYPE: "RsaOaep256",

CLIENT\_VERSION : "v2",

1. Update below code to generate capture context for microform v2

**Path: cartridges/int\_cybs\_sfra\_base/cartridge/apiClient/api/KeyGenerationApi.js**

**Method: this.generatePublicKey**

return this.apiClient.callApi(

'/microform/v2/sessions', 'POST',

pathParams, queryParams, headerParams, formParams, postBody,

authNames, contentTypes, accepts, returnType, callback

);

1. Update ApiClient.js with below code to send response back

**Path: cartridges/int\_cybs\_sfra\_base/cartridge/apiClient/ApiClient.js**

**Method: CallApi method**

if (response.ok) {

var responseObj = response.object;

if(path === '/microform/v2/sessions'){

callback(responseObj, false, response);

}else{

callback(JSON.parse(responseObj), false, response);

}

} else {

callback(response.errorMessage, response.error, response);

}

# **Step 2: Decode and Validate Capture Context**

1. Update controller with below code to call script and render template with required information.

**Path: cartridges/int\_cybs\_sfra\_base/cartridge/controllers/SecureAcceptance.js**

server.get('CreateFlexToken', server.middleware.https, function (req, res, next) {

var Flex = require('~/cartridge/scripts/http/payments');

var flexResult = Flex.createFlexKey(); // call to create capture context

var parsedPayload = Flex.jwtDecode(flexResult); // parse capture context and validate public key

if(parsedPayload != null){ // extract client library as clientLibrary integrity values from capture context

var clientLibrary = parsedPayload.ctx[0].data.clientLibrary;

var clientLibraryIntegrity = parsedPayload.ctx[0].data.clientLibraryIntegrity;

res.render('secureAcceptanceFlexMicroformContent', { // add client library and client library integrity values dynamically

flexTokenResult: flexResult,

clientLibrary: clientLibrary,

clientLibraryIntegrity: clientLibraryIntegrity

});

next();

}

});

1. Decode capture context:

**Path: cartridges\int\_cybs\_sfra\_base\cartridge\scripts\http\payments.js**

// function to decode capture context and validate capture context using the public key

function jwtDecode(jwt){

var captureContext = jwt;

var Encoding = require('dw/crypto/Encoding');

var Signature = require('dw/crypto/Signature');

var Bytes = require('dw/util/Bytes');

var apiSig = new Signature();

var encodedHeader = captureContext.split('.')[0];

var encodedPayload = captureContext.split('.')[1];

var jwtSignature = captureContext.split('.')[2];

var kid = JSON.parse(Encoding.fromBase64(encodedHeader)).kid ;

var alg = JSON.parse(Encoding.fromBase64(encodedHeader)).alg;

var decodedPayload = Encoding.fromBase64(encodedPayload).toString();

var parsedPayload = JSON.parse(decodedPayload);

var decodedJwt = null ;

// generate public key using the kid from capture context

var pKid = getPublicKey(kid);

// Create public key using modulus and exponent value to validate capture context

var pkey = require('../http/publicKey');

if(!empty(pKid.n) && !empty(pKid.e)){

var RSApublickey = pkey.getRSAPublicKey(pKid.n, pKid.e);

var JWTAlgoToSFCCMapping = {

RS256 : "SHA256withRSA",

RS512 : "SHA512withRSA",

RS384 : "SHA384withRSA",

};

// validate capture context using the generated public key

var jwtSignatureInBytes = new Encoding.fromBase64(jwtSignature);

var contentToVerify = encodedHeader + '.' + encodedPayload;

contentToVerify = new Bytes(contentToVerify);

var isValid = apiSig.verifyBytesSignature(jwtSignatureInBytes, contentToVerify , new Bytes(RSApublickey) ,JWTAlgoToSFCCMapping[alg]) ;

if(isValid){

decodedJwt = parsedPayload;

}

}

return decodedJwt;

}

// Add below method to get public key by passing kid (extracted from capture context)

function getPublicKey(kid){

var cybersourceRestApi = require('../../apiClient/index');

var instance = new cybersourceRestApi.AsymmetricKeyManagementApi(configObject);

var jwk = '';

instance.getP12KeyDetails(kid, function (data, error, response) {

jwk = data;

})

return jwk;

}

1. Update below code to generate public key

**Path: cartridges/int\_cybs\_sfra\_base/cartridge/apiClient/api/AsymmetricKeyManagementApi.js**

**Method: this.getP12KeyDetails**

var accepts = ['application/json'];

return this.apiClient.callApi(

'/flex/v2/public-keys/{keyId}', 'GET',

pathParams, queryParams, headerParams, formParams, postBody,

authNames, contentTypes, accepts, returnType, callback

);

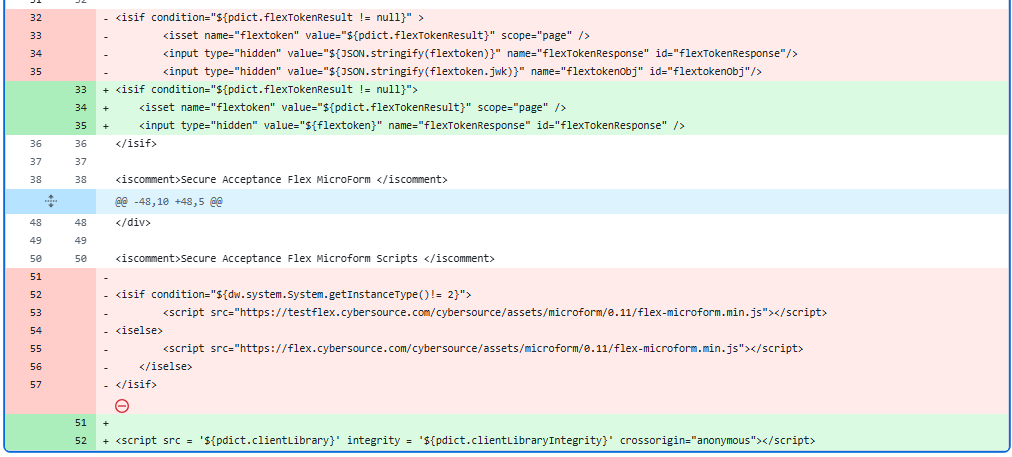
1. Navigate to the path below in our cartridge and add this file to your custom cartridge. This will create a public key which is used to validate our capture context

**Path: cartridges/int\_cybs\_sfra\_base/cartridge/scripts/http/publicKey.js**

# **Step 3: Add clientLibrary and clientLibraryIntegrity values**

Update **secureAcceptanceFlexMicroformContent.isml file** as per the screenshot attached below.

**Path: cartridges\int\_cybs\_sfra\cartridge\templates\default\secureAcceptanceFlexMicroformContent.isml**



# **Step 4: Load flex IFrame**

**Path: cartridges\int\_cybs\_sfra\cartridge\client\default\custom\flexMicroform.js**



# **Step 5: Replace diners-club with dinersclub**

Replace all the occurrences of **diners-club** with **dinersclub** in below files.

**Path:**

* **cartridges/int\_cybs\_sfra/cartridge/client/default/custom/flexMicroform.js**
* **cartridges/int\_cybs\_sfra/cartridge/client/default/scss/components/\_creditCardField.scss**
* **cartridges/int\_cybs\_sfra\_base/cartridge/scripts/hooks/payment/processor/payments\_credit\_form\_processor.js**

# **Step 6: Create Configurations**

Add below lines of code in FlexMicroform.xml and merged.xml files

**Path:**

* **metadata\payments\_metadata\meta\merged.xml**
* **metadata\payments\_metadata\meta\FlexMicroform.xml**

<attribute-definition attribute-id="Cybersource\_AllowedCardNetworks">

<display-name xml:lang="x-default">allowedCardNetworks</display-name>

<description xml:lang="x-default">Configure card types for Cybersource Flex Microform</description>

<type>enum-of-string</type>

<mandatory-flag>false</mandatory-flag>

<externally-managed-flag>false</externally-managed-flag>

<select-multiple-flag>true</select-multiple-flag>

<value-definitions>

<value-definition default="true">

<display xml:lang="x-default">VISA</display>

<value>VISA</value>

</value-definition>

<value-definition>

<display xml:lang="x-default">MAESTRO</display>

<value>MAESTRO</value>

</value-definition>

<value-definition>

<display xml:lang="x-default">MASTERCARD</display>

<value>MASTERCARD</value>

</value-definition>

<value-definition>

<display xml:lang="x-default">AMEX</display>

<value>AMEX</value>

</value-definition>

<value-definition>

<display xml:lang="x-default">DISCOVER</display>

<value>DISCOVER</value>

</value-definition>

<value-definition>

<display xml:lang="x-default">DINERSCLUB</display>

<value>DINERSCLUB</value>

</value-definition>

<value-definition>

<display xml:lang="x-default">JCB</display>

<value>JCB</value>

</value-definition>

</value-definitions>

</attribute-definition>

<group-definitions>

<attribute-group group-id="Cybersource\_FlexMicroform">

<attribute attribute-id="Cybersource\_AllowedCardNetworks"/>

</attribute-group>

</group-definitions>