Cybersource for Salesforce B2C Commerce SOAP - Microform v2 Upgrade Steps

Contents

Introduction

This guide is for merchants who are using older versions of our Salesforce B2C Commerce SOAP API cartridges and want to update the version of Microform without updating to our cartridge version 24.1.3 which adds Microform v2.

Step 1. Generate the server-side capture context

1. Navigate to **Administration > Operations > Services > CybersourceFlexToken -> Details** and replace the existing URL with the below one
   TEST: https://apitest.cybersource.com/microform/v2/sessions?format=JWT
   PRODUCTION: https://api.cybersource.com/microform/v2/sessions?format=JWT
2. Create a custom preference to add allowed networks for Microform.
   Refer to section **Create Configurations** to create required configurations.

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

Field	Description	Value to Set
SA_Flex_AllowedCardNetworks	Configure card types for Cybersource Microform	VISA MASTER DISCOVER DINERSCLUB JCB MAESTRO AMEX ELO CUP JCREW CARTESBANCAIRES

3. Navigate to below path and replace **CreateFlexKey()** method with below code
   Path: \cartridges\int_cybersource_sfra\cartridge\scripts\secureacceptance\adapter\Flex.js

```
function CreateFlexKey() {
    var HashMap = require('dw/util/HashMap');
    var collections = require('*/cartridge/scripts/util/collections');
    var CRServices = require('*/cartridge/scripts/init/RestServiceInit');
    var signedHeaders = new HashMap();
    var Site = require('dw/system/Site');
    var sharedSecret =
Site.getCurrent().getCustomPreferenceValue('SA_Flex_SharedSecret');
    var keyID = Site.getCurrent().getCustomPreferenceValue('SA_Flex_KeyID');
    // eslint-disable-next-line
    var host =
dw.system.Site.getCurrent().getCustomPreferenceValue('SA_Flex_HostName');
    var signature;
```

```
    var targetOrigin;
    // eslint-disable-next-line    var merchantId =
dw.system.Site.getCurrent().getCustomPreferenceValue('CsMerchantId');

    // eslint-disable-next-line
    if (request.isHttpSecure()) {
        // eslint-disable-next-line
        targetOrigin = 'https://' + request.httpHost;
    } else {
        // eslint-disable-next-line
        targetOrigin = 'http://' + request.httpHost;
    }

    var allowedCNetworks =
dw.system.Site.getCurrent().getCustomPreferenceValue('SA_Flex_AllowedCardNetworks
');
    var list = [];
    if (empty(allowedCNetworks)) {
        list.push('VISA');
    } else {
        for (let i = 0; allowedCNetworks[i] != null; i++) {
            list.push(allowedCNetworks[i].value);
        }
    }

    var digest = {
        'targetOrigins': [
            targetOrigin
        ],
        'allowedCardNetworks': list,
        'clientVersion': "v2"
    };
    var CybersourceConstants =
require('*/cartridge/scripts/utils/CybersourceConstants');

    digest.clientReferenceInformation = {};
    var digestString = JSON.stringify(digest);

    signedHeaders.put('host', host);
    signedHeaders.put('date', getTime());
    signedHeaders.put('request-target', 'post
/microform/v2/sessions?format=JWT');
    signedHeaders.put('digest', getDigest(digestString));
```

```
    signedHeaders.put('v-c-merchant-id', merchantId);
    signature = generateSignature(signedHeaders, keyID, sharedSecret);
    var headerString = '';
    collections.forEach(signedHeaders.keySet(), function (key) {
        // for each(var key in signedHeaders.keySet()){
        headerString = headerString + ' ' + key;
    });
    var signatureMap = new HashMap();
    signatureMap.put('keyid', keyID);
    signatureMap.put('algorithm', 'HmacSHA256');
    signatureMap.put('headers', headerString);
    signatureMap.put('signature', signature);
    var signaturefields = '';
    collections.forEach(signatureMap.keySet(), function (key) {
        signaturefields = signaturefields + key + '="' + signatureMap.get(key) +
'", ';
    });
    signaturefields = signaturefields.slice(0, signaturefields.length - 2);
    signedHeaders.put('signature', signaturefields);
    signedHeaders.remove('request-target');
    var service = CRServices.CyberSourceFlexTokenService;
    var serviceResponse = service.call(signedHeaders, digestString);
    return serviceResponse.object;
}
```

Step 2: Decode and Validate Capture Context

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

   Path: cartridges\int_cybersource_sfra\cartridge\controllers\CYBSecureAcceptance.js

```
server.get('CreateFlexToken', server.middleware.https, function (req, res, next)
{
    var Flex = require(CybersourceConstants.CS_CORE_SCRIPT +
'secureacceptance/adapter/Flex');
    var flexResult = Flex.CreateFlexKey();
    var parsedPayload = Flex.jwtDecode(flexResult);
    if (parsedPayload != null) {
        var clientLibrary = parsedPayload.ctx[0].data.clientLibrary;
        var clientLibraryIntegrity =
parsedPayload.ctx[0].data.clientLibraryIntegrity;
        if(clientLibraryIntegrity && clientLibrary){
res.render('checkout/billing/paymentOptions/secureAcceptanceFlexMicroformContent'
, {
```

```
                flexTokenResult: flexResult,
                clientLibrary: clientLibrary,
                clientLibraryIntegrity: clientLibraryIntegrity
            });
        }
        next();
    }
});
```

2. Decode capture context

Path: cartridges\int_cybersource_sfra\cartridge\scripts\secureacceptance\adapter\Flex.js

```
// function to decode capture context and validate capture context using the
public key
function jwtDecode(jwt) {
    var response = jwt;
    var Logger = require('dw/system/Logger');
    var Encoding = require('dw/crypto/Encoding');

    var encodedHeader = response.split('.')[0];
    var kid = JSON.parse(Encoding.fromBase64(encodedHeader)).kid;
    var alg = JSON.parse(Encoding.fromBase64(encodedHeader)).alg;

    var encodedPayload = response.split('.')[1];
    var decodedPayload = Encoding.fromBase64(encodedPayload).toString();
    var parsedPayload = JSON.parse(decodedPayload);

    // return parsedPayload;
    var decodedJwt = null;
    var jwtSignature = response.split('.')[2];
    var pKid = getPublicKey(kid);
    var pkey = require('../../helper/publicKey');
    if (!empty(pKid.n) && !empty(pKid.e)) {
        var RSApublickey = pkey.getRSAPublicKey(pKid.n, pKid.e);
        var JWTAlgoToSFCCMapping = {
            RS256: "SHA256withRSA",
            RS512: "SHA512withRSA",
            RS384: "SHA384withRSA",
        };
        var Signature = require('dw/crypto/Signature');
        var apiSig = new Signature();
        var Bytes = require('dw/util/Bytes');
        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 HashMap = require('dw/util/HashMap');
    var collections = require('*/cartridge/scripts/util/collections');
    var CRServices = require('*/cartridge/scripts/init/RestServiceInit');
    var signedHeaders = new HashMap();
    var Site = require('dw/system/Site');

    var sharedSecret =
Site.getCurrent().getCustomPreferenceValue('SA_Flex_SharedSecret');
    var keyID = Site.getCurrent().getCustomPreferenceValue('SA_Flex_KeyID');
    // eslint-disable-next-line
    var host =
dw.system.Site.getCurrent().getCustomPreferenceValue('SA_Flex_HostName');
    var signature;
    var targetOrigin;
    // eslint-disable-next-line
    var merchantId =
dw.system.Site.getCurrent().getCustomPreferenceValue('CsMerchantId');

    signedHeaders.put('host', host);
    signedHeaders.put('User-Agent', 'Mozilla/5.0');
    signedHeaders.put('date', getTime());
    signedHeaders.put('request-target', 'get /flex/v2/public-keys/' + kid);
    signedHeaders.put('v-c-merchant-id', merchantId);
    signature = generateSignature(signedHeaders, keyID, sharedSecret);
    var headerString = '';
    collections.forEach(signedHeaders.keySet(), function (key) {
```

```
        headerString = headerString + ' ' + key;
    });
    var signatureMap = new HashMap();
    signatureMap.put('keyid', keyID);
    signatureMap.put('algorithm', 'HmacSHA256');
    signatureMap.put('headers', headerString);
    signatureMap.put('signature', signature);
    var signaturefields = '';
    collections.forEach(signatureMap.keySet(), function (key) {
        signaturefields = signaturefields + key + '="' + signatureMap.get(key) +
'", ';
    });
    signaturefields = signaturefields.slice(0, signaturefields.length - 2);
    signedHeaders.put('signature', signaturefields);
    signedHeaders.remove('request-target');
    var service = CRServices.CyberSourceAssymentricKeyManagement;
    var serviceResponse = service.call(signedHeaders, kid);
    return JSON.parse(serviceResponse.object);
}
```

3. Add below lines of code in **services.xml** file

   Path: \metadata\sfra_meta\sfra_meta\services.xml

```
<service-credential service-credential-id="AssymentricKeyManagement">
    <url>https://apitest.cybersource.com/flex/v2/public-keys</url>
    <user-id></user-id>
    <password encrypted="true" encryption-type="common.export"></password>
</service-credential>

<service service-id="cybersource.assymentrickeymanagement">
    <service-type>HTTP</service-type>
    <enabled>true</enabled>
    <log-prefix>Cybersource</log-prefix>
    <comm-log-enabled>true</comm-log-enabled>
    <force-prd-enabled>false</force-prd-enabled>
    <mock-mode-enabled>false</mock-mode-enabled>
    <profile-id>CybersourceProfile</profile-id>
    <credential-id>AssymentricKeyManagement</credential-id>
</service>
```

4. Add below code to generate public key in **RestServiceInit.js** file

   Path: cartridges\int_cybersource_sfra\cartridge\scripts\init\RestServiceInit.js

```
var CyberSourceAssymentricKeyManagement =
LocalServiceRegistry.createService('cybersource.assymentrickeymanagement', {
    createRequest: function (svc, requestObj, keyId) {
```

```
        var collections = require('*/cartridge/scripts/util/collections');
        svc.setRequestMethod('GET');
        svc.addHeader('Accept', 'application/json');
        svc.addHeader('Content-Type', 'application/json;charset=utf-8');
        collections.forEach(requestObj.keySet(), function (key) {
            //  for each (var key in requestObj.keySet()) {
            svc.addHeader(key, requestObj.get(key));
        });
        // eslint-disable-next-line
        svc.URL += "/" + keyId;
    },
    parseResponse: function (svc, client) {
        return client.text;
    },
    filterLogMessage: function (msg) {
        //  No need to filter logs.  No sensitive information.
        return msg;
    }
});

module.exports = {
    CyberSourceFlexTokenService: CyberSourceFlexTokenService,
    CyberSourceDMService: CyberSourceDMService,
    CyberSourceAssymentricKeyManagement: CyberSourceAssymentricKeyManagement
};
```

5. Navigate to the path **cartridges\int_cybersource_sfra\cartridge\scripts\helper\publicKey.js** in our cartridge version 24.1.3 available in GitHub and add this file to your custom cartridge. This is used to create a public key required to validate the capture context.

Step 3: Add clientLibrary and clientLibraryIntegrity values

Update the **secureAcceptanceFlexMicroformContent.isml file** with the below code to add **clientLibrary** and **clientLibraryIntegrity** values.

Path:
cartridges\int_cybersource_sfra\cartridge\templates\default\checkout\billing\paymentOptions\secureAcceptanceFlexMicroformContent.isml

```
<isset name="flexError"
    value="${pdict.flexTokenResult == null ? Resource.msg('flex.tokenerror',
'cybersource', null) : ''}" scope="page" />
<div class="row">
    <div class="col-12">
        <div class="form-group cardNumber"
```

```
            data-cardNumber="${Resource.msg('cardnumber.placeholder',
'cybersource', null)}">
            <label class="form-control-label"
for="cardNumber">${Resource.msg('field.credit.card.number', 'creditCard',
null)}</label>
            <div class="card-number-wrapper">
                <div id="cardNumber-container" class="form-control"></div>
                <div class="invalid-feedback"></div>
            </div>
            <isif condition="${pdict.flexTokenResult == null}">
                <div class="alert alert-danger">${flexError}</div>
            </isif>
        </div>
    </div>
</div>

<div class="row">
    <div class="col-12">
        <div class="form-group securityCode"
            data-cardNumber="${Resource.msg('securityCode.placeholder',
'cybersource', null)}">
            <label class="form-control-label"
for="securityCode">${Resource.msg('field.credit.card.security.code',
'creditCard', null)}</label>
            <div class="security-code-wrapper">
                <div id="securityCode-container" class="form-control"></div>
                <div class="invalid-feedback"></div>
            </div>
            <isif condition="${pdict.flexTokenResult == null}">
                <div class="alert alert-danger">${flexError}</div>
            </isif>
        </div>
    </div>
</div>

<isif condition="${pdict.flexTokenResult != null}">
    <isset name="flextoken" value="${pdict.flexTokenResult}" scope="page" />
    <input type="hidden" value="${flextoken}" name="flextokenRespose"
id="flextokenRespose" />
</isif>

<iscomment>Secure Acceptance Flex MicroForm </iscomment>
<div class="row">
    <div class="col-12">
        <div class="form-group">
```

```
            <div class="flexresponse-wrapper">
                <input type="hidden" class="form-control flex-response" id="flex-
response" value=""
                    name="dwfrm_billing_creditCardFields_flexresponse">
            </div>
        </div>
    </div>
</div>

<iscomment>Secure Acceptance Flex Microform Scripts </iscomment>

<script src='${pdict.clientLibrary}' integrity='${pdict.clientLibraryIntegrity}'
crossorigin="anonymous"></script>
```

Step 4: Load Microform iframe

Update the **flexMicrofom.js** file with the below code to load Microform iframe.

Path: cartridges/int_cybersource_sfra/cartridge/client/default/custom/flexMicroform.js

```
/* eslint-disable no-undef */

'use strict';

$(document).ready(function () {
  var captureContext = $('#flextokenRespose').val();
  var flex = new Flex(captureContext); // eslint-disable-line no-undef
  var cardNumberplaceholder = $("#credit-card-content.cardNumber").attr(
    "data-cardNumber"
  );
  var customStyles = {
    input: {
      "font-family":
        '-apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,"Helvetica
Neue",Arial,sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol"',
      "font-size": "1rem",
      "line-height": "1.5",
      color: "#495057",
    },
    ":focus": {
      color: "blue",
    },
    ":disabled": {
      cursor: "not-allowed",
    },
    valid: {
      color: "#3c763d",
```

```
  },
  invalid: {
    color: "#a94442",
  },
};
var microform = flex.microform("card", {
  styles: customStyles,
});
var number = microform.createField("number");
var securityCode = microform.createField("securityCode");
securityCode.load("#securityCode-container");
number.load("#cardNumber-container");
number.on("change", function (data) {
  var cardType = data.card[0].name;
  $(".card-number-wrapper").attr("data-type", cardType);
  $("#cardType").val(cardType);
});

function parseJwt(token) {
  // eslint-disable-line no-inner-declarations
  var base64Url = token.split(".")[1];
  var base64 = base64Url.replace(/-/g, "+").replace(/_/g, "/");
  var jsonPayload = decodeURIComponent(
    atob(base64)
      .split("")
      .map(function (c) {
        // eslint-disable-line no-undef
        return "%" + ("00" + c.charCodeAt(0).toString(16)).slice(-2);
      })
      .join("")
  );

  return JSON.parse(jsonPayload);
}

function flexTokenCreation() {
  // eslint-disable-line no-inner-declarations
  var expMonth = $("#expirationMonth").val() || $("#month").val();
  var expYear = $("#expirationYear").val() || $("#year").val();
  if (expMonth == "" || expYear == "") {
    return false;
  }
  // Send in optional parameters from other parts of your payment form
  var options = {
    expirationMonth: expMonth.length == 1 ? "0" + expMonth : expMonth,
```

```
      expirationYear: expYear,
      // cardType: /* ... */
    };
    // validation
    // look for field validation errors

    microform.createToken(options, function (err, response) {
      // At this point the token may be added to the form
      // as hidden fields and the submission continued
      if (err) {
        $(".card-number-wrapper .invalid-feedback")
          .text(err.message)
          .css("display", "block");
        return true;
      }

      var decodedJwt = parseJwt(response);
      document.getElementById("cardNumber").valid = true;

      $("#flex-response").val(decodedJwt.jti);

      $('#cardNumber').val(decodedJwt.content.paymentInformation.card.number.mask
edValue);

      if ($(".submit-payment").length === 1) {
        $(".submit-payment").trigger("click");
      } else {
        $(".save-payment").trigger("click");
      }
    });
    return true;
  }
  // check for card type function
  function assignCorrectCardType() {
    // eslint-disable-line no-inner-declarations
    var cardType = $("#cardType").val();
    if (cardType.charCodeAt(0) !== cardType.toUpperCase().charCodeAt(0)) {
      var correctCardType = "";
      switch (
      cardType // eslint-disable-line default-case
      ) {
        case "visa":
          correctCardType = "Visa";
          break;
        case "mastercard":
```

```
        correctCardType = "Master Card";
        break;
      case "amex":
        correctCardType = "Amex";
        break;
      case "discover":
        correctCardType = "Discover";
        break;
      case "dinersclub":
        correctCardType = "DinersClub";
        break;
      case "maestro":
        correctCardType = "Maestro";
        break;
      case "jcb":
        correctCardType = "JCB";
        break;
      case "cartesbancaires":
        correctCardType = "CartesBancaires";
        break;
      case "elo":
        correctCardType = "Elo";
        break;
      case "cup":
        correctCardType = "China UnionPay";
        break;
      case "jcrew":
        correctCardType = "JCrew";
        break;
    }
    $("#cardType").val(correctCardType);
  }
}

$(".payment-summary .edit-button").on("click", function () {
  $("#flex-response").val("");
});

// intercept the form submission and make a tokenize request instead
$(".submit-payment").on("click", function (event) {
  if (
    $("#expirationMonth").val() !== "" &&
    $("#expirationYear").val() !== "" &&
    ($("#flex-response").val() === "" ||
      $("#flex-response").val() === undefined) &&
```

```
      ($(".data-checkout-stage").data("customer-type") === "guest" ||
        ($(".data-checkout-stage").data("customer-type") === "registered" &&
          $(".payment-information").data("is-new-payment")))
    ) {
      if (
        $("#flex-response").val() === "" ||
        $("#flex-response").val() === undefined
      ) {
        flexTokenCreation();
        assignCorrectCardType();
        event.stopImmediatePropagation();
      }
    }
  });
  $(".save-payment").on("click", function (event) {
    if (
      $("#flex-response").val() === "" ||
      $("#flex-response").val() === undefined
    ) {
      flexTokenCreation();
      assignCorrectCardType();
      event.preventDefault();
    }
  });
});
```

Step 5: Replace diners-club with dinersclub

In the file **cartridges/int_cybersource_sfra/cartridge/client/default/custom/flexMicroform.js,** replace **diners-club** with **dinersclub.**

Step 6: Create Configurations

Add below lines of code in Cybersource_SecureAcceptance.xml and test the changes.

Path: metadata\sfra_meta\sfra_meta\meta\Cybersource_SecureAcceptance.xml

```
<attribute-definition attribute-id="SA_Flex_AllowedCardNetworks">
    <display-name xml:lang="x-default">Allowed Card Networks for Flex</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-definition>
            <display xml:lang="x-default">ELO</display>
            <value>ELO</value>
        </value-definition>
        <value-definition>
            <display xml:lang="x-default">CUP</display>
            <value>CUP</value>
        </value-definition>
        <value-definition>
            <display xml:lang="x-default">JCREW</display>
            <value>JCREW</value>
        </value-definition>
        <value-definition>
            <display xml:lang="x-default">CARTESBANCAIRES</display>
            <value>CARTESBANCAIRES</value>
        </value-definition>
```

```
    </value-definitions>
</attribute-definition>
<group-definitions>
    <attribute-group group-id="CyberSource_SecureAcceptance">
        <attribute attribute-id="Cybersource_AllowedCardNetworks" />
    </attribute-group>
</group-definitions>
```