Merchant documentation guide for SFCC Ent P12 authentication

Contents

1. Merchants using ENT cartridge v21.1.0 and above

Step 1: Create p12 file

1. Follow steps mentioned in the link to generate a p12 certificate in Business Center.
2. Make a note of password set to the p12 key.
3. Download the generated p12 file.

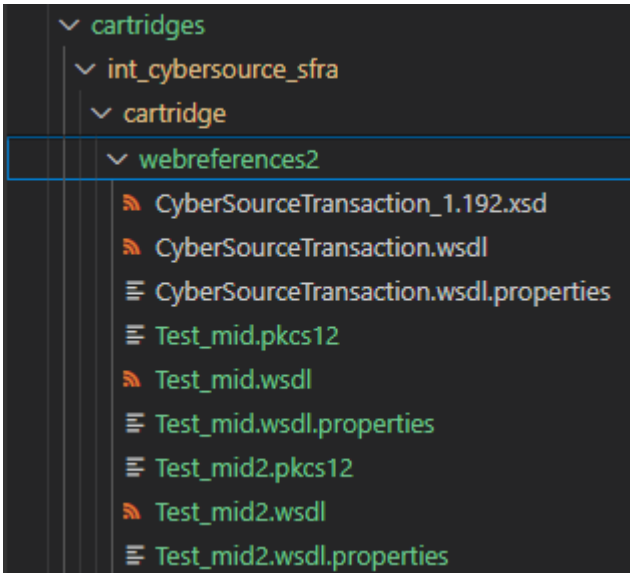
Step 2: Upload the p12 file in our cartridge

Place the file/files in the webreferences2 folder of the same cartridge as the WSDL file.
Path: cartridges\int_cybersource_sfra\cartridge\webreferences2

In case of multiple merchant Ids, duplicate the **CyberSourceTransaction.wsdl** file, **CyberSourceTransaction.wsdl.properties** file and rename them with the same name as your respective p12 files.

Note: The name of the WSDL file and properties file must be same as the p12 file. Change the extension of p12 file to jks or pkcs12, if it has a different extension.

Example: Test_mid and Test_mid2 are our generated p12 files added to webreferences2 folder



Step 3: Extract friendly name from the keystore

Run the below command in the terminal to extract the content of p12 file and make a note of the friendly name of the first certificate.
Command: openssl pkcs12 -in CyberSourceTransactsion.pkcs12 -info
Friendly name example: serialNumber=169039929641101872410 2,CN=Test_mid

Step 4: Create custom preference to add p12 file name, p12 username and password

To use the file for authentication, the p12 file Name, username and password need to be passed.
Refer section **Create Configurations** to create required configurations.
Go to **Merchant Tools** > **Site Preferences** > **Custom Preferences** > **Cybersource** and set values for the following parameters

Field	Description
CsP12_Name	Name of the p12 file added in webreferences2 folder in Step2.
CsP12_UserName	Friendly name extracted in Step 3. Use the serialNumber and CN of the friendly name returned. Ex: serialNumber=169039929641101872410 2,CN=sfcc_cybs
CsP12_Password	Password of the p12 file which was generated in Step 1.

Step 5: Code Changes

1. Replace the **createRequest** & **execute** method (present in below path) with below code snippet.
   Path: \int_cybersource_sfra\cartridge\scripts\init\SoapServiceInit.js

```
createRequest: function (svc, requestObj) {
        // eslint-disable-next-line

        var libCybersource =
require('*/cartridge/scripts/cybersource/libCybersource');
        var CybersourceHelper = libCybersource.getCybersourceHelper();
        var csReference = new CybersourceHelper.getcsReference();
        var service = csReference.getDefaultService();
        CybersourceHelper.setEndpoint(service);

        // eslint-disable-next-line
        svc.webReference = csReference;
        // eslint-disable-next-line
        svc.serviceClient = service;

        if (requestObj) {
            return requestObj;
        }
        return null;
    },

execute: function (svc, parameter) {
        var libCybersource =
require('~/cartridge/scripts/cybersource/libCybersource');
```

```
        var CybersourceHelper = libCybersource.getCybersourceHelper();
        var password = CybersourceHelper.getP12Password();
        var userName = CybersourceHelper.getP12UserName();

        var secretsMap = new HashMap();
        secretsMap.put(userName, password);

        var requestCfg = new HashMap();
        requestCfg.put(WSUtil.WS_ACTION, WSUtil.WS_TIMESTAMP + " " +
WSUtil.WS_SIGNATURE);

        requestCfg.put(WSUtil.WS_SIGNATURE_USER, userName);
        requestCfg.put(WSUtil.WS_PASSWORD_TYPE, WSUtil.WS_PW_TEXT);
        requestCfg.put(WSUtil.WS_SIG_DIGEST_ALGO,
"http://www.w3.org/2001/04/xmlenc#sha256");

        // define signature properties
        // the keystore file has the basename of the WSDL file and the
        // file extension based on the keystore type (for example,
HelloWorld.pkcs12).
        // The keystore file has to be placed beside the WSDL file.
        requestCfg.put(WSUtil.WS_SIG_PROP_KEYSTORE_TYPE, "pkcs12");
        requestCfg.put(WSUtil.WS_SIG_PROP_KEYSTORE_PW, password);
        requestCfg.put(WSUtil.WS_SIG_PROP_KEYSTORE_ALIAS, userName);
        requestCfg.put(WSUtil.WS_SIGNATURE_PARTS,
"{Element}{http://schemas.xmlsoap.org/soap/envelope/}Body");
        requestCfg.put(WSUtil.WS_SIG_KEY_ID,
WSUtil.KEY_ID_TYPE_DIRECT_REFERENCE);

        requestCfg.put(WSUtil.WS_SECRETS_MAP, secretsMap);

        //response-config--------------------------
        var responseCfg = new HashMap();
        responseCfg.put(WSUtil.WS_ACTION, WSUtil.WS_TIMESTAMP);

        WSUtil.setWSSecurityConfig(svc.serviceClient, requestCfg, responseCfg);
// Setting WS security
        return svc.serviceClient.runTransaction(parameter.request);
    },
```

2. Add below code snippet to **libCybersource.js**

   Path: int_cybersource_sfra\cartridge\scripts\cybersource\libCybersource.js

```
var CybersourceHelper = {
```

```
    getcsReference: function() {
        var wsdlName = Site.getCurrent().getCustomPreferenceValue('CsP12_Name');
        var webref = webreferences2[wsdlName];
        return webref;
    },

    getP12Password: function() {
        return Site.getCurrent().getCustomPreferenceValue('CsP12_Password');
    },

    getP12UserName: function() {
        return Site.getCurrent().getCustomPreferenceValue('CsP12_UserName');
    },
```

3. Update **signedDataUsingHMAC256()** in below file

   Path: \int_cybersource_sfra\cartridge\scripts\helper\CommonHelper.js

```
function signedDataUsingHMAC256(dataToSign, secretKey, paymentType) {
    var KeyRef = require('dw/crypto/KeyRef');
    var libCybersource =
require('~/cartridge/scripts/cybersource/libCybersource');
    var signature;
    var mac = new dw.crypto.Mac(dw.crypto.Mac.HMAC_SHA_256);
    var CybersourceHelper = libCybersource.getCybersourceHelper();
    if(paymentType === 'KLI'){
        var privateKey = new
KeyRef(CybersourceHelper.getklarnaPrivateKeyAlias());
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign,
privateKey));
    } else{
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign, new
dw.util.Bytes(secretKey, 'UTF-8')));
    }
    return signature;
}
```

4. Replace **var csReference = webreferences2.CyberSourceTransaction;** with below lines of code

```
var libCybersource = require('*/cartridge/scripts/cybersource/libCybersource');
var CybersourceHelper = libCybersource.getCybersourceHelper();
var csReference = new CybersourceHelper.getcsReference();
```

5. Replace **new CybersourceHelper.csReference.methodName();** with below lines of code

```
new CybersourceHelper.getcsReference().methodName();
```

Note: Here the **methodName**() can be any value so please search for **new CybersourceHelper.csReference.** to find the references in our cartridge. A find and replace option can be used.

Step 6: Remove unused code

1. Remove the definition of **getMerhcantCredentials()** and **getSoapSecurityKey()** functions in **libCybersource.js**.
   Path: cartridge\scripts\cybersource\libCybersource.js
2. Remove all the references **getMerhcantCredentials()** from our cartridge.
3. Refer to **getMerchantID()'s merchantId** instead of **getMerhcantCredentials()'s** method.
4. Remove CsSecurityKey attribute definition and from attribute group in **Cybersource.xml file**.
   Path: metadata\sfra_meta\meta\Cybersource.xml
5. Remove merchantId and merchantKey references from **payment-methods.xml**
   Path: metadata\sfra_meta\sites\yourSiteId\payment-methods.xml

Create Configurations

Add below lines of code in **Cybersource.xml** file to add configurations

```
<attribute-definition attribute-id="CsP12_Name">
        <display-name xml:lang="x-default">Cybersorce P12 Name</display-name>
        <description xml:lang="x-default">Name of the p12 file added in webrefernces2
folder</description>
        <type>string</type>
        <mandatory-flag>false</mandatory-flag>
        <externally-managed-flag>false</externally-managed-flag>
        <min-length>0</min-length>
 </attribute-definition>
  <attribute-definition attribute-id="CsP12_UserName">
        <display-name xml:lang="x-default">Cybersorce P12 User Name</display-name>
        <description xml:lang="x-default">Use the serialNumber and CN of the friendly name part of
the p12 file</description>
        <type>string</type>
        <mandatory-flag>false</mandatory-flag>
        <externally-managed-flag>false</externally-managed-flag>
        <min-length>0</min-length>
 </attribute-definition>
<attribute-definition attribute-id="CsP12_Password">
        <display-name xml:lang="x-default">Cybersource P12 Password</display-name>
        <description xml:lang="x-default">Enter the password added while generating p12
certificate</description>
        <type>password</type>
        <mandatory-flag>false</mandatory-flag>
```

```
        <externally-managed-flag>false</externally-managed-flag>
 </attribute-definition>
<group-definitions>
       <attribute-group group-id="CyberSource">
         <display-name xml:lang="x-default">CyberSource: Core</display-name>
         <attribute attribute-id="IsCartridgeEnabled"/>
         <attribute attribute-id="CsMerchantId"/>
        <attribute attribute-id="CsP12_Name"/>
         <attribute attribute-id="CsP12_UserName"/>
         <attribute attribute-id="CsP12_Password"/>
         <attribute attribute-id="CsEndpoint"/>
         <attribute attribute-id="CsDeveloperID"/>
         <attribute attribute-id="CsDebugCybersource"/>
         <attribute attribute-id="csMasterCardAuthIndicator"/>
         <attribute attribute-id="csCardDecisionManagerEnable"/>
         <attribute attribute-id="CsOrderImportLookBack"/>
       </attribute-group>
</group-definitions>
```

2. Setup for Klarna payment method to use p12 authentication - ENT cartridge v21.1.0 and above

Step 1: Create p12 file

Refer Step 1 of Section 1 to generate P12 file.

Step 2: Upload the p12 file to Business Manager

1. Login to **Business manager**
2. Navigate to **Administration** > **Operations** > **Private Keys and Certificates**
3. Click on **Import** button, a popup appears
4. Click on **select and browse the p12 file** from your local.
5. Enter **Alias and Source Password** (generated in step 1) and click on Save.

Step 3: Create custom preference to add key alias

To use the file for authentication, the p12 alias need to be passed.

Refer **Create Configurations** to create required configurations.

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

Configure **klarnaPrivateKeyAlias** with the Alias provided in Step 2.

Step 4: Code Changes

1. Update **signedDataUsingHMAC256()** in below file

   Path: int_cybersource_sfra\cartridge\scripts\helper\CommonHelper.js

```
function signedDataUsingHMAC256(dataToSign, secretKey, paymentType) {
    var signature;
    var mac = new dw.crypto.Mac(dw.crypto.Mac.HMAC_SHA_256);
    var KeyRef = require('dw/crypto/KeyRef');
    var libCybersource =
require('~/cartridge/scripts/cybersource/libCybersource');
    var CybersourceHelper = libCybersource.getCybersourceHelper();
    var privateKey = new KeyRef(CybersourceHelper.getklarnaPrivateKeyAlias());
    if(paymentType === 'KLI'){
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign,
privateKey));
    }else{
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign, new
dw.util.Bytes(secretKey, 'UTF-8')));
    }
    return signature;
}
```

2. Add **getklarnaPrivateKeyAlias()** definition in **libCybersource.js**

   Path: int_cybersource_sfra\cartridge\scripts\cybersource\libCybersource.js

```
getklarnaPrivateKeyAlias: function() {
        return
Site.getCurrent().getCustomPreferenceValue('klarnaPrivateKeyAlias');
    },
```

3. Update below line of code in **CreateKlarnaSecureKey()** of **CybKlarna.js**

   Path: int_cybersource_sfra\cartridge\controllers\CYBKlarna.js

```
var signature = CommonHelper.signedDataUsingHMAC256(token, null, paymentType);
```

4. Update below line of code in **CreateKlarnaSecureKey()** of **KlarnaAdaptor.js**

   Path: int_cybersource_sfra\cartridge\controllers\ KlarnaAdaptor.js

```
var signature = CommonHelper.signedDataUsingHMAC256(token, null, paymentType);
```

Step 5: Remove custom from payment-methods.xml references

```
<custom-attribute attribute-id="merchantID" xml:lang="x-default"></custom-
attribute>
<custom-attribute attribute-id="merchantKey" xml:lang="x-default"></custom-
attribute>
```

Create Configurations

Add below lines of code in **Cybersource_Klarna.xml** file

```
<attribute-definition attribute-id="klarnaPrivateKeyAlias">
      <display-name xml:lang="x-default">Klarna Private Key Alias</display-name>
      <description xml:lang="x-default">Private Key Alias of imported Key in
Private Keys and Certificates.</description>
      <type>string</type>
      <mandatory-flag>false</mandatory-flag>
      <externally-managed-flag>false</externally-managed-flag>
      <min-length>0</min-length>
</attribute-definition>
<group-definitions>
      <attribute-group group-id="CyberSource_Klarna">
            <display-name xml:lang="x-default">CyberSource: Klarna</display-
name>
             <attribute attribute-id="isKlarnaRedirectionRequired"/>
             <attribute attribute-id="klarnaJSAPIPath"/>
             <attribute attribute-id="klarnaPrivateKeyAlias"/>
             <attribute attribute-id="IsKlarnaPaymentFlowModeEnabled"/>
             <attribute attribute-id="isKlarnaDecisionManagerRequired"/>
        </attribute-group>
</group-definitions>
```

3. Merchants using Site Genesis cartridge v21.1.0 and above

Step 1: Create p12 file

Refer Step 1 of Section 1 to generate P12 file.

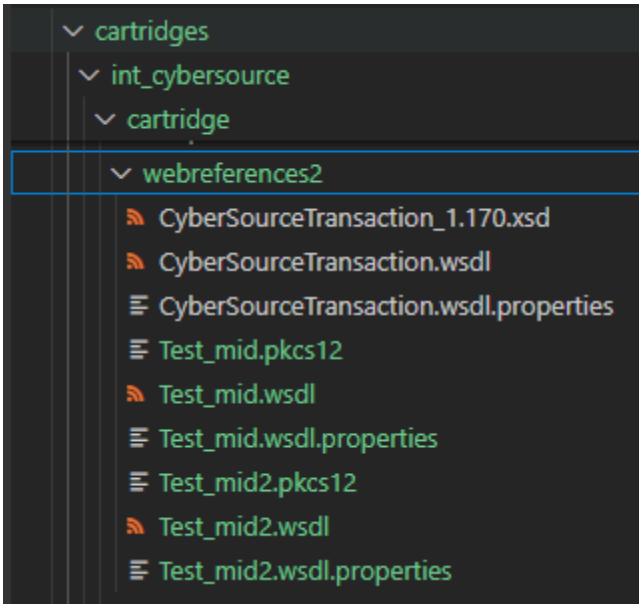
Step 2: Upload the p12 file in our cartridge

Place the file/files in the webreferences2 folder of the same cartridge as the WSDL file.
Path: cartridges\int_cybersource_sfra\cartridge\webreferences2

In case of multiple merchant Ids, duplicate the **CyberSourceTransaction.wsdl** file, **CyberSourceTransaction.wsdl.properties** file and rename them with the same name as your respective p12 files.

Note: The name of the WSDL file and properties file must be same as the p12 file. Change the extension of p12 file to jks or pkcs12, if it has a different extension.

Example: Test_mid and Test_mid2 are our generated p12 files added to webreferences2 folder



Step 3: Extract friendly name from the keystore

Run the below command in the terminal to extract the content of p12 file and make a note of the friendly name of the first certificate.

Command: openssl pkcs12 -in CyberSourceTransaction.pkcs12 -info

Friendly name example: serialNumber=169039929641101872410 2,CN=Test_mid

Step 4: Create custom preference to add p12 username and password.

To use the key for authentication, the p12 file name, username and password need to be passed.

Refer to **Create Configurations** to create required configurations.

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

Field	Description
CsP12_Name	Name of the p12 file added in webreferences2 folder in Step2.
CsP12_UserName	Friendly name extracted in Step 3. Use the serialNumber and CN of the friendly name returned. Ex: serialNumber=169039929641101872410 2,CN=sfcc_cybs
CsP12_Password	Password of the p12 file which was generated in Step 1.

Step 5: Code Changes

1. Replace the **execute** function (present in below path) with below code snippet.
   Path: \int_cybersource\cartridge\scripts\init\SoapServiceInit.js

```
execute: function (svc, parameter) {
        var libCybersource =
require('~/cartridge/scripts/cybersource/libCybersource');
```

```
        var CybersourceHelper = libCybersource.getCybersourceHelper();
        var password = CybersourceHelper.getP12Password();
        var userName = CybersourceHelper.getP12UserName();

        var secretsMap = new HashMap();
        secretsMap.put(userName, password);

        var requestCfg = new HashMap();
        requestCfg.put(WSUtil.WS_ACTION, WSUtil.WS_TIMESTAMP + " " +
WSUtil.WS_SIGNATURE);

        requestCfg.put(WSUtil.WS_SIGNATURE_USER, userName);
        requestCfg.put(WSUtil.WS_PASSWORD_TYPE, WSUtil.WS_PW_TEXT);
        requestCfg.put(WSUtil.WS_SIG_DIGEST_ALGO,
"http://www.w3.org/2001/04/xmlenc#sha256");

        // define signature properties
        // the keystore file has the basename of the WSDL file and the
        // file extension based on the keystore type (for example,
HelloWorld.pkcs12).
        // The keystore file has to be placed beside the WSDL file.
        requestCfg.put(WSUtil.WS_SIG_PROP_KEYSTORE_TYPE, "pkcs12");
        requestCfg.put(WSUtil.WS_SIG_PROP_KEYSTORE_PW, password);
        requestCfg.put(WSUtil.WS_SIG_PROP_KEYSTORE_ALIAS, userName);
        requestCfg.put(WSUtil.WS_SIGNATURE_PARTS,
"{Element}{http://schemas.xmlsoap.org/soap/envelope/}Body");
        requestCfg.put(WSUtil.WS_SIG_KEY_ID,
WSUtil.KEY_ID_TYPE_DIRECT_REFERENCE);

        requestCfg.put(WSUtil.WS_SECRETS_MAP, secretsMap);

        //response-config--------------------------
        var responseCfg = new HashMap();
        responseCfg.put(WSUtil.WS_ACTION, WSUtil.WS_TIMESTAMP);

        WSUtil.setWSSecurityConfig(svc.serviceClient, requestCfg, responseCfg);
// Setting WS security
        return svc.serviceClient.runTransaction(parameter.request);
    },
```

2. Add below code snippet to **libCybersource.js**
   Path: int_cybersource\cartridge\scripts\cybersource\libCybersource.js

```
var CybersourceHelper = {
```

```
    getcsReference: function() {
        var wsdlName = Site.getCurrent().getCustomPreferenceValue('CsP12_Name');
        var webref = webreferences2[wsdlName];
        return webref;
    },

    getP12Password: function() {
        return Site.getCurrent().getCustomPreferenceValue('CsP12_Password');
    },

    getP12UserName: function() {
        return Site.getCurrent().getCustomPreferenceValue('CsP12_UserName');
    },
```

3. Update **signedDataUsingHMAC256()** in below file

   Path: \int_cybersource_sfra\cartridge\scripts\helper\CommonHelper.js

```
function signedDataUsingHMAC256(dataToSign, secretKey, paymentType) {
    var KeyRef = require('dw/crypto/KeyRef');
    var libCybersource =
require('~/cartridge/scripts/cybersource/libCybersource');
    var signature;
    var mac = new dw.crypto.Mac(dw.crypto.Mac.HMAC_SHA_256);
    var CybersourceHelper = libCybersource.getCybersourceHelper();
    if(paymentType === 'KLI'){
        var privateKey = new
KeyRef(CybersourceHelper.getklarnaPrivateKeyAlias());
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign,
privateKey));
    }else{
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign, new
dw.util.Bytes(secretKey, 'UTF-8')));
    }
    return signature;
}
```

4. Replace **var csReference = webreferences2.CyberSourceTransaction;** with below lines of code

```
var libCybersource = require('*/cartridge/scripts/cybersource/libCybersource');
var CybersourceHelper = libCybersource.getCybersourceHelper();
var csReference = new CybersourceHelper.getcsReference();
```

5. Replace **new CybersourceHelper.csReference.methodName();** with below lines of code

```
var csReference = new CybersourceHelper.methodName();
```

Note: Here the **methodName()** can be any value so please search for new **CybersourceHelper.csReference.** to find the references in our cartridge. A find and replace option can be used.

Step 6: Remove unused code

1. Remove definition of **getMerhcantCredentials()** and **getSoapSecurityKey()** functions in **libCybersource.js**.
   Path: int_cybersource\cartridge\scripts\cybersource\libCybersource.js
2. Remove all the references **getMerhcantCredentials()** from our cartridge.
3. Refer to **getMerchantID()'s merchantId** instead of **getMerhcantCredentials()'s** method.
4. Remove CsSecurityKey attribute definition and from attribute group in **Cybersource.xml** file.
   Path: metadata\site_genesis_meta\meta\Cybersource.xml
5. Remove merchantId and merchantKey references from **payment-methods.xml**
   Path: metadata\ site_genesis_meta \sites\yourSiteId\payment-methods.xml

Create Configurations

Add below lines of code in **Cybersource.xml** file

```
<attribute-definition attribute-id="CsP12_Name">
        <display-name xml:lang="x-default">Cybersorce P12 Name</display-name>
        <description xml:lang="x-default">Name of the p12 file added in webrefernces2
folder</description>
        <type>string</type>
        <mandatory-flag>false</mandatory-flag>
        <externally-managed-flag>false</externally-managed-flag>
        <min-length>0</min-length>
 </attribute-definition>
  <attribute-definition attribute-id="CsP12_UserName">
        <display-name xml:lang="x-default">Cybersorce P12 User Name</display-name>
        <description xml:lang="x-default">Use the serialNumber and CN of the friendly name part of
the p12 file</description>
        <type>string</type>
        <mandatory-flag>false</mandatory-flag>
        <externally-managed-flag>false</externally-managed-flag>
        <min-length>0</min-length>
 </attribute-definition>
<attribute-definition attribute-id="CsP12_Password">
        <display-name xml:lang="x-default">Cybersource P12 Password</display-name>
        <description xml:lang="x-default">Enter the password added while generating p12
certificate</description>
```

```
        <type>password</type>
        <mandatory-flag>false</mandatory-flag>
        <externally-managed-flag>false</externally-managed-flag>
 </attribute-definition>
<group-definitions>
      <attribute-group group-id="CyberSource">
        <display-name xml:lang="x-default">CyberSource: Core</display-name>
        <attribute attribute-id="IsCartridgeEnabled"/>
        <attribute attribute-id="CsMerchantId"/>
       <attribute attribute-id="CsP12_Name"/>
        <attribute attribute-id="CsP12_UserName"/>
        <attribute attribute-id="CsP12_Password"/>
        -------------
      </attribute-group>
    </group-definitions>
```

4. Setup for Klarna payment method to use p12 authentication - Site Genesis cartridge v21.1.0 and above

Step 1: Create p12 file

Refer Step 1 of Section 1 to generate P12 file.

Step 2: Upload the p12 file to Business Manager

1. Login to **Business manager**
2. Navigate to **Administration** > **Operations** > **Private Keys and Certificates**
3. Click on **Import** button, a popup appears
4. Click on **select and browse the p12 file** from your local.
5. Enter **Alias and Source Password** (generated in step 1) and click on Save.

Step 3: Create custom preference to add key alias.

To use the file for authentication, the p12 alias need to be passed.

Refer to **Create Configurations** to create required configurations.

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

Configure **klarnaPrivateKeyAlias** with the Alias provided in Step 2.

Step 4: Code Changes

1. Update **signedDataUsingHMAC256()** in below file

   Path: int_cybersource\cartridge\scripts\helper\CommonHelper.js

```
function signedDataUsingHMAC256(dataToSign, secretKey, paymentType) {
```

```
    var signature;
    var mac = new dw.crypto.Mac(dw.crypto.Mac.HMAC_SHA_256);
    var KeyRef = require('dw/crypto/KeyRef');
    var libCybersource =
require('~/cartridge/scripts/cybersource/libCybersource');
    var CybersourceHelper = libCybersource.getCybersourceHelper();
    var privateKey = new KeyRef(CybersourceHelper.getklarnaPrivateKeyAlias());
    if(paymentType === 'KLI'){
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign,
privateKey));
    }else{
        signature = dw.crypto.Encoding.toBase64(mac.digest(dataToSign, new
dw.util.Bytes(secretKey, 'UTF-8')));
    }
    return signature;
}
```

2. Add **getklarnaPrivateKeyAlias()** definition in **libCybersource.js**
   Path: int_cybersource\cartridge\scripts\cybersource\libCybersource.js

```
getklarnaPrivateKeyAlias: function() {
        return
Site.getCurrent().getCustomPreferenceValue('klarnaPrivateKeyAlias');
    },
```

3. Update below line of code in **CreateKlarnaSecureKey()** of **CybKlarna.js**
   Path: int_cybersource\cartridge\controllers\ KlarnaAdaptor.js

```
var signature = CommonHelper.signedDataUsingHMAC256(token, null, paymentType);
```

Step 5: Remove custom from payment-methods.xml references

```
<custom-attribute attribute-id="merchantID" xml:lang="x-default"></custom-
attribute>
<custom-attribute attribute-id="merchantKey" xml:lang="x-default"></custom-
attribute>
```

Create Configurations

Add below lines of code in **Cybersource_Klarna.xml** file

```
<attribute-definition attribute-id="klarnaPrivateKeyAlias">
      <display-name xml:lang="x-default">Klarna Private Key Alias</display-name>
      <description xml:lang="x-default">Private Key Alias of imported Key in
Private Keys and Certificates.</description>
       <type>string</type>
       <mandatory-flag>false</mandatory-flag>
       <externally-managed-flag>false</externally-managed-flag>
       <min-length>0</min-length>
```

```
</attribute-definition>
<group-definitions>
       <attribute-group group-id="CyberSource_Klarna">
             <display-name xml:lang="x-default">CyberSource Klarna</display-name>
             <attribute attribute-id="isKlarnaRedirectionRequired"/>
             <attribute attribute-id="klarnaPrivateKeyAlias"/>
             <attribute attribute-id="isKlarnaDecisionManagerRequired"/>
             <attribute attribute-id="klarnaJSAPIPath"/>
        </attribute-group>
</group-definitions>
```

5. Merchants using cartridge version older than v21.1.0

We strongly recommend merchants using older versions of our cartridge to upgrade to our latest cartridge version as the older version contains deprecated packages and methods which may not be compatible with our latest changes.

However, please follow below steps to update required files to be compliant with the p12 authentication change.

Step1: Update folder name from webreference to webreferences2.
Change all the references of webreference to webreferences2 in our cartridge.

Step 2: Add below changes to SoapServiceInit.js

30 ...cartridge/scripts/init/SoapServiceInit.ds → ...cartridge/scripts/init/SoapServiceInit.js

```
@@ -6,7 +6,7 @@
6  6  /*******************************************************************************/
7  7  var dwsvc = require ("dw/svc");
8  8  var HashMap = require ("dw/util/HashMap");
9     - var SOAPUtil = require ("dw/rpc/SOAPUtil");
   9  + var WSUtil = require('dw/ws/WSUtil');
10 10  var LocalServiceRegistry = require('dw/svc/LocalServiceRegistry');
11 11  /**
```

```
41  41          var libCybersource = require('~/cartridge/scripts/cybersource/libCybersource');
    @@ -63,57 +63,57 @@
63  63          var secretsMap = new HashMap();
64  64          secretsMap.put(userName, password);
65  65          var requestCfg = new HashMap();
66      -       requestCfg.put(SOAPUtil.WS_ACTION, SOAPUtil.WS_USERNAME_TOKEN);
67      -       requestCfg.put(SOAPUtil.WS_USER, userName);
68      -       requestCfg.put(SOAPUtil.WS_PASSWORD_TYPE, SOAPUtil.WS_PW_TEXT);
69      -       requestCfg.put(SOAPUtil.WS_SECRETS_MAP, secretsMap);
    66  +       requestCfg.put(WSUtil.WS_ACTION, WSUtil.WS_USERNAME_TOKEN);
    67  +       requestCfg.put(WSUtil.WS_USER, userName);
    68  +       requestCfg.put(WSUtil.WS_PASSWORD_TYPE, WSUtil.WS_PW_TEXT);
    69  +       requestCfg.put(WSUtil.WS_SECRETS_MAP, secretsMap);
70  70
71  71          var responseCfg = new HashMap();
72      -       responseCfg.put(SOAPUtil.WS_ACTION, SOAPUtil.WS_TIMESTAMP);
    72  +       responseCfg.put(WSUtil.WS_ACTION, WSUtil.WS_TIMESTAMP);
73  73
74      -       SOAPUtil.setWSSecurityConfig(svc.serviceClient, requestCfg, responseCfg); // Setting WS security
    74  +       WSUtil.setWSSecurityConfig(svc.serviceClient, requestCfg, responseCfg); // Setting WS security
75  75
76  76          return svc.serviceClient.runTransaction(parameter.request);
```

Step 3: Please refer to below screenshots and make changes in libCybersource.js

```
278     -
279 483     setEndpoint: function (service) {
280 484         var endpoint = CybersourceHelper.getEndpoint();
281 485         var Logger = dw.system.Logger.getLogger('Cybersource');
282 486         Logger.debug('Connection to system "{0}"', endpoint);
283     -       var Stub = require('dw/rpc/Stub');
284     -
    487 +       var Port = require('dw/ws/Port');
    488 +       var WSUtil = require('dw/ws/WSUtil');
285 489         switch (endpoint) {
286 490             case 'Production':
287     -               service._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, 'https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor');
    491 +               WSUtil.setProperty(Port.ENDPOINT_ADDRESS_PROPERTY, 'https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor', service);
288 492                 break;
289 493             case 'Test':
290     -               service._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, 'https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor');
    494 +               WSUtil.setProperty(Port.ENDPOINT_ADDRESS_PROPERTY, 'https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor', service);
291 495                 break;
292 496             default:
    497 +               // eslint-disable-next-line
293 498                 throw 'Undefined Cybersource Endpoint "' + endpoint + '"';
294 499         }
295 500     },
296 501
```

```
296     -               var Stub = require('dw/rpc/Stub');
    296 +               var Port = require('dw/ws/Port');
    297 +       var WSUtil = require('dw/ws/WSUtil');
297 298
298 299                   switch ( endpoint ) {
299 300                           case "Production":
300     -                             service._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY,'https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor');
    301 +                             WSUtil.setProperty(Port.ENDPOINT_ADDRESS_PROPERTY,'https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor',
        service);
301 302                               break;
302 303                           case "Test" :
303     -                             service._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY,'https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor');
    304 +                             WSUtil.setProperty(Port.ENDPOINT_ADDRESS_PROPERTY,'https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor',
        service);
304 305                               break;
305 306                           default:
306 307                                   throw "Undefined Cybersource Endpoint \"" + endpoint + "\"";
```

Step 4: Post completing the above changes please make the changes by referring to Section 1.

Note: **webreference** has been updated to **webreferences2** in later versions of our cartridge. So, changes added to replace **webreferences2** in Section1 to be considered as **webreferences** in older versions.