**Merchant documentation guide for SFCC Ent P12 authentication**

Contents

[1. Merchants using ENT cartridge v21.1.0 and above 1](#_Toc185009300)

[Step 1: Create p12 file 1](#_Toc185009301)

[Step 2: Upload the p12 file in our cartridge 1](#_Toc185009302)

[Step 3: Extract friendly name from the keystore 1](#_Toc185009303)

[Step 4: Create custom preference to add p12 username and password 1](#_Toc185009304)

[Step 5: Code Changes 1](#_Toc185009305)

[Step 6: Remove unused code 3](#_Toc185009306)

[Create Configurations 3](#_Toc185009307)

[2. Setup for Klarna payment method to use p12 authentication - ENT cartridge v21.1.0 and above 5](#_Toc185009308)

[Step 1: Create p12 file 5](#_Toc185009309)

[Step 2: Upload the p12 file to Business Manager 5](#_Toc185009310)

[Step 3: Create custom preference to add key alias 5](#_Toc185009311)

[Step 4: Code Changes 5](#_Toc185009312)

[Step 5: Remove custom from payment-methods.xml references 6](#_Toc185009313)

[Create Configurations 6](#_Toc185009314)

[3. Merchants using Site Genesis cartridge v21.1.0 and above 7](#_Toc185009315)

[Step 1: Create p12 file 7](#_Toc185009316)

[Step 2: Upload the p12 file in our cartridge 7](#_Toc185009317)

[Step 3: Extract friendly name from the keystore 7](#_Toc185009318)

[Step 4: Create custom preference to add p12 username and password. 7](#_Toc185009319)

[Step 5: Code Changes 7](#_Toc185009320)

[Step 6: Remove unused code 9](#_Toc185009321)

[Create Configurations 9](#_Toc185009322)

[4. Setup for Klarna payment method to use p12 authentication - Site Genesis cartridge v21.1.0 and above 10](#_Toc185009323)

[Step 1: Create p12 file 10](#_Toc185009324)

[Step 2: Upload the p12 file to Business Manager 10](#_Toc185009325)

[Step 3: Create custom preference to add key alias. 10](#_Toc185009326)

[Step 4: Code Changes 11](#_Toc185009327)

[Step 5: Remove custom from payment-methods.xml references 11](#_Toc185009328)

[Create Configurations 12](#_Toc185009329)

[5. Merchants using cartridge version older than v21.1.0 12](#_Toc185009330)

# Merchants using ENT cartridge v21.1.0 and above

## Step 1: Create p12 file

1. Follow steps mentioned in the [link](https://developer.cybersource.com/docs/cybs/en-us/platform/developer/all/rest/rest-getting-started/restgs-jwt-message-intro/restgs-security-p12-intro/restgs-security-P12.html) to generate a p12 certificate from EBC.
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 in the webreferences2 folder of the same cartridge as the WSDL file.

**Path:** cartridges\int\_cybersource\_sfra\cartridge\webreferences2

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

## 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

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

To use the file for authentication, we need to pass the p12 username and password.

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

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

1. Set **CsP12\_UserName** with friendly name extracted in [Step 3](#_Step_3:_Extract).
2. Set **CsP12\_Password** with Password created in [Step 1](#_Step_1:_).

## Step 5: Code Changes

* + - 1. Replace the execute function (present in below path) with below code snippet.

**Path:** \int\_cybersource\_sfra\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);

},

* + - 1. Add below code snippet to libCybersource.js

**Path:** int\_cybersource\_sfra\cartridge\scripts\cybersource\libCybersource.js

var CybersourceHelper = {

getP12Password: function() {

return Site.getCurrent().getCustomPreferenceValue('CsP12\_Password');

},

getP12UserName: function() {

return Site.getCurrent().getCustomPreferenceValue('CsP12\_UserName');

},

1. 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;

}

## Step 6: Remove unused code

1. Remove the definition of **getMerhcantCredentials()** and **getSoapSecurityKey()** functions in libCybersource.js.

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

1. Remove all the references **getMerhcantCredentials()** from our cartridge.
2. Refer to **getMerchantID()’s merchantId** instead of **getMerhcantCredentials()’s** method.
3. Remove CsSecurityKey attribute definition and from attribute group in Cybersource.xml file.

**Path:** metadata\sfra\_meta\meta\Cybersource.xml

1. 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\_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>

<attribute-definition attribute-id="CsP12\_UserName">

<display-name xml:lang="x-default">Cybersorce P12 User Name</display-name>

<description xml:lang="x-default">Friendly name extracted from p12</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">

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

<attribute attribute-id="IsCartridgeEnabled"/>

<attribute attribute-id="CsMerchantId"/>

<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>

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

## Step 1: Create p12 file

Refer [Step 1](#_Step_1:__1) of [Section 1](#_Merchants_using_ENT) 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, we need to pass the p12 alias.

Refer [**Create Configurations**](#_Create_Configurations:__1)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_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;

}

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

**Path:** int\_cybersource\_sfra\cartridge\scripts\cybersource\libCybersource.js

getklarnaPrivateKeyAlias: function() {

return Site.getCurrent().getCustomPreferenceValue('klarnaPrivateKeyAlias');

},

1. 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);

1. 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>

# Merchants using Site Genesis cartridge v21.1.0 and above

## Step 1: Create p12 file

Refer [Step 1](#_Step_1:__1) of [Section 1](#_Merchants_using_ENT) to generate P12 file.

## Step 2: Upload the p12 file in our cartridge

Place the file in the webreferences2 folder of the same cartridge as the WSDL file.

**Path:** cartridges\int\_cybersource\cartridge\webreferences2

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

## 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

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

To use the key for authentication we need to pass the p12 username and password.

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

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

1. Set **CsP12\_UserName** with friendly name extracted in [Step 3](#_Step_3:_Extract_2).
2. Set **CsP12\_Password** with Password created in [Step 1](#_Step_1:_Create_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);

},

1. Add below code snippet to libCybersource.js

**Path:** int\_cybersource\cartridge\scripts\cybersource\libCybersource.js

var CybersourceHelper = {

getP12Password: function() {

return Site.getCurrent().getCustomPreferenceValue('CsP12\_Password');

},

getP12UserName: function() {

return Site.getCurrent().getCustomPreferenceValue('CsP12\_UserName');

},

1. 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;

}

## Step 6: Remove unused code

1. Remove definition of **getMerhcantCredentials()** and **getSoapSecurityKey()** functions in libCybersource.js.

**Path:** int\_cybersource\cartridge\scripts\cybersource\libCybersource.js

1. Remove all the references **getMerhcantCredentials()** from our cartridge.
2. Refer to **getMerchantID()’s merchantId** instead of **getMerhcantCredentials()’s** method.
3. Remove CsSecurityKey attribute definition and from attribute group in Cybersource.xml file.

**Path:** metadata\site\_genesis\_meta\meta\Cybersource.xml

1. 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\_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>

<attribute-definition attribute-id="CsP12\_UserName">

<display-name xml:lang="x-default">Cybersorce P12 User Name</display-name>

<description xml:lang="x-default">Friendly name extracted from p12</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">

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

<attribute attribute-id="IsCartridgeEnabled"/>

<attribute attribute-id="CsMerchantId"/>

<attribute attribute-id="CsP12\_UserName"/>

<attribute attribute-id="CsP12\_Password"/>

--------------

</attribute-group>

</group-definitions>

# 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](#_Step_1:__1) of [Section 1](#_Merchants_using_ENT) 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, we need to pass the p12 alias.

Refer to [**Create Configurations**](#_Step_6:_Create)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_2:_Upload)

## 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;

}

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

**Path:** int\_cybersource\cartridge\scripts\cybersource\libCybersource.js

getklarnaPrivateKeyAlias: function() {

return Site.getCurrent().getCustomPreferenceValue('klarnaPrivateKeyAlias');

},

1. 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>

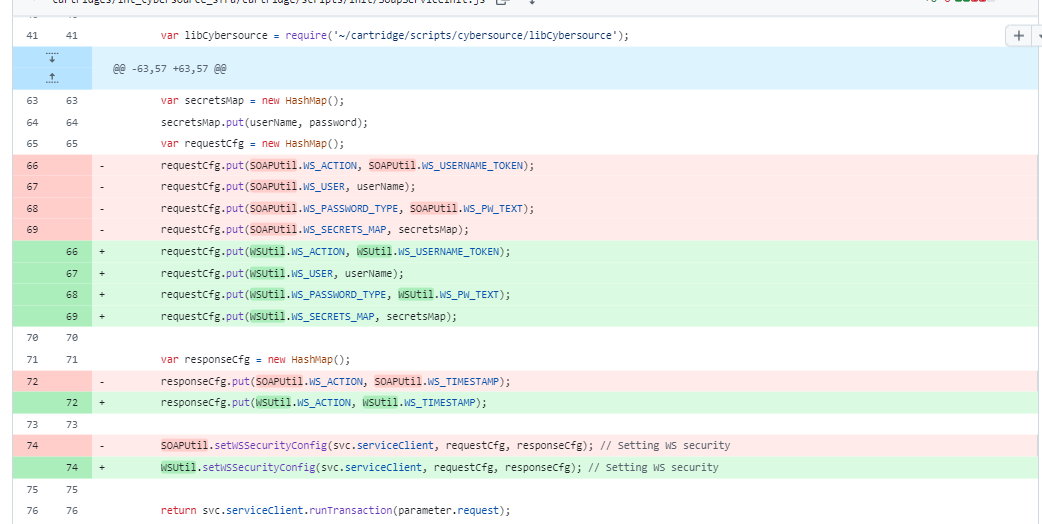
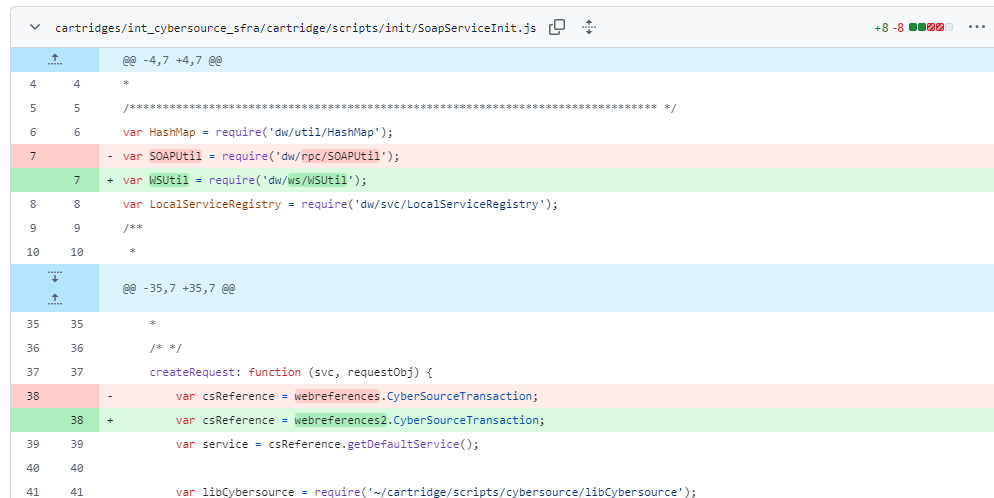
# Merchants using cartridge version older than v21.1.0

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.

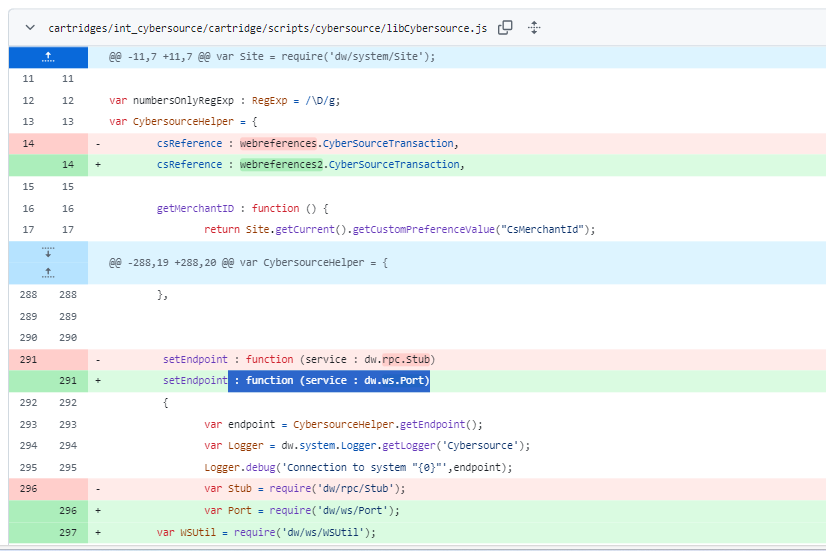
However, please follow below steps to update required files to be compliant with the p12 authentication change.

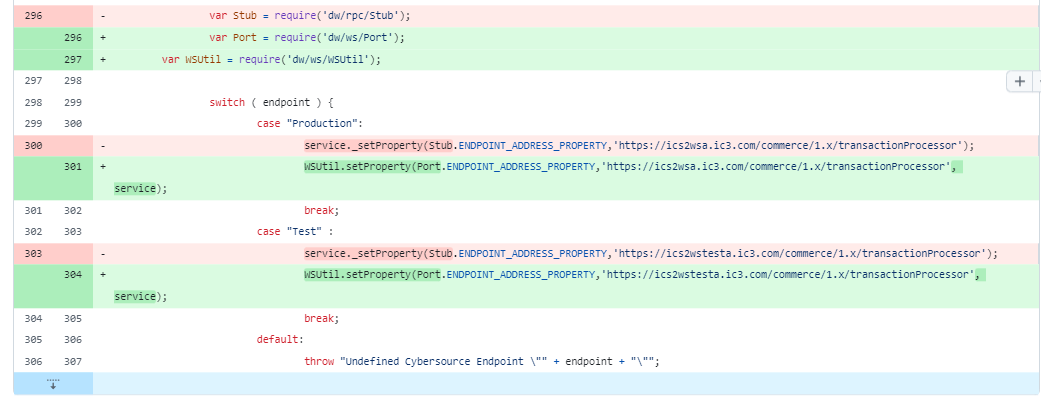
**Step1: Update folder name from webreferences to webreferences2.**

Change all the references of webreference to webreferences2 in our cartridge.

**Step 2: Add below changes to SoapServiceInit.js** 

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





**Step 4: Post completing the above changes please make the changes by referring** **to** [Section 1](#_Merchants_using_ENT_1).