Cybersource®

Oracle Commerce Cloud Installation Guide

August 2023

cybersource
A Visa Solution

Copyright:

Contents

1. Prerequisites for Oracle Commerce Cloud

The following is required before going through installation steps:

- Yarn version: 1.22.4
- NodeJS version: 16.15.0, You could use NVM to manage multiple versions locally
- OCC environment
  - **OCC Admin interface:** https://asbx80c1dev-admin-{env}.oraclecloud.com/occs-admin/
  - **OCC Storefront:** https://asbx80c1dev-store-{env}.oraclecloud.com
- Application Key
- User credentials for OCC Admin

Install all the dependencies by running '**yarn install'** from the project's root.
You need to build all packages at once using '**yarn build:prod'** command from project's root.

2. Install Payment Gateway

2.1. Configure gateway

Before installing payment gateway you should decide whether you will support all payment methods or only some of it. There are two ways you can manage it:

1. Disabling payment type from OCC Admin interface which will result in payment option not being rendered in UI (Payment Widget).
2. Removing unrelated configuration properties from `packages/payment-gateway/gateway/isv-occ-gateway/config/config.json`

Note: In most cases just disabling unsupported payment type from OCC Admin is preferable. Removing unsupported configuration properties can be done in case particular payment types (e.g. GooglePay) should be initially excluded from OCC Admin interface.

To remove unrelated properties please consider the following:

1. To exclude GooglePay payment settings remove the following from "packages/payment-gateway/gateway/isv-occ-gateway/config/config.json"
   a. **configuration properties:** "googlePaySdkUrl","googlePayEnvironment", "googlePayGateway","googlePayGatewayMerchantId","googlePayMerchant Id"," googlePayMerchantName"," googlePaySupportedNetworks"
   b. **Remove "googlepay" option from following properties**: "paymentOptions", "dmDecisionSkip"
2. To exclude ApplePay payment settings remove the following from "packages/payment-gateway/gateway/isv-occ-gateway/config/config.json"
   a. **configuration properties:** "applePayMerchantId","applePayInitiative", "applePayInitiativeContext","applePaySupportedNetworks"
   b. **Remove "applepay" option from following properties:** "paymentOptions", "dmDecisionSkip"

After adjusting payment gateway settings you should also remove respective properties from "packages/payment-gateway/settings.json".

2.2. Deploy

2.2.1. Create an extension ID

In order to upload an extension into Commerce Cloud, you must generate an ID for the extension and update the same in packages/payment-gateway/ext.json file

To create an extension ID:

- Log into Commerce Cloud.
- Navigate to Settings -> Extensions -> Developer tab.
- Click on Generate ID button.
- Enter a name for the extension and click Save.
- Copy and update the generated extension ID

2.2.2. Upload Extension

Before uploading the extension, zip up all the files within your packages/payment-gateway directory except settings.json. This is the file you upload to Commerce Cloud to make the extension available for use.

- In the "Settings" tab on the left panel.
- In settings, click on "Extension" button.
- Click the Upload Extension button and select the extension zip file from your local file system.

Once the module is installed, head back to the Oracle Commerce Cloud Admin to configure it.

2.3. Enable gateway in OCC gateway

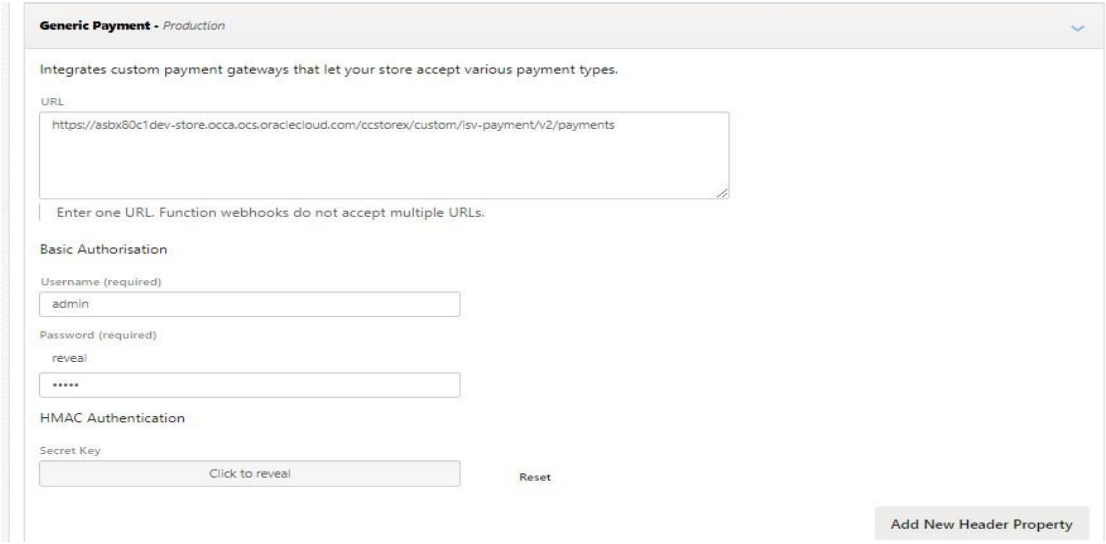
After successful deployment you will need to enable payment gateway:

- Go to OCC Admin -> Settings -> Payment Processing
- Open 'Payment gateways' tab and choose 'ISV OCC Gateway' from the list
- Select 'Payment Gateway Enabled' option
- Configure gateway settings by providing values (e.g. merchant credentials) for particular channel (Preview, Storefront, Agent)
- Save Changes
- Go back to the 'Payment Types' type
- Select supported credit card types from the list
- You might want to also provide list of supported billing countries as well as default one
- Save Changes

3. Configure Generic Webhooks

You should setup Generic Payment so that OCC knows where to send payment requests. See the "Understand function webhooks" section from the Understand webhooks document

- Open OCC Admin and got to Settings -> Web APIs -> Function APIs
- Configure both Generic Payment Production and Preview webhooks as shown below



- Please replace URL (the `{env}` part) with your specific value
- Make sure you copy secret key as it will be required for SSE deployment

4. Install SSE (server-extension)

4.1. Configure

Configure production settings in the following file `packages/server-extension/config/app.prod.json`:

- "cache.service.ttl.secs" - Default caching TTL, can be zero value
- "cache.gatewaysettings.ttl.secs" - Caching TTL for gateway settings call (see "packages/server-extension/src/middlewares/gatewaySettings.ts"). You might want to use TTL value '1' while testing SSE so that changes in gateway settings performed in OCC Admin become immediately available to SSE and Payment Widget respectively
- "crypto.service.key" - Random key which is used to encrypt data so that it is not tampered in UI
- "partner.developerId" - Leave the value as is
- "partner.solutionId" - Leave the value as is
- "logging.webhook.http" - Enable webhook request/response logging
- "logging.api.error" - Enable logging for errors
- "logging.api.access" - Enable logging for incoming requests
- "payments.secret.key" - Webhook secret key

Note: "packages/server-extension/config/app.local.json" is applied only in local development environment.

4.2. Deploy

Build SSE extension with the following command

cd packages/occ-sse-gateway

yarn build:prod

Deploy SSE by running commands (from project's root):

yarn occ package-app osf-payment-sse

yarn occ upload-app -u ${OCC_ADMIN_HOST} -k ${APPLICATION_KEY} osf-payment-sse

where

- "APPLICATION_KEY" - Application Key created in Settings -> Web APIs -> Registered Applications
- "OCC_ADMIN_HOST" –your OCC specific environment, e.g. "asbx80c1dev-admin-{env}.oraclecloud.com"

5. Install Payment Widget (payment-widget)

5.1. Deploy

- Copy the contents from cybersource-plugins-oraclecxcommerce/plugins into the plugins directory of your storefront (OSF workspace) code.
  Copy plugins/actions into your storefront code and export the actions in the index and meta files:

```
export * from '@oracle-cx-commerce/actions';

export const flexMicroformAction = () => import('./flex-microform-action');
export const applePayValidationAction = () => import('./apple-pay-validation-action');
export const getPayerAuthSetupAction = () => import('./get-payer-auth-setup-action');
```

plugins/actions/index.js

```
export * from '@oracle-cx-commerce/actions/meta';

export {flexMicroformAction} from './flex-microform-action/meta';
export {applePayValidationAction} from './apple-pay-validation-action/meta';
export {getPayerAuthSetupAction} from './get-payer-auth-setup-action/meta';
```

plugins/actions/meta.js

Copy plugins/components into your storefront code and export the components in the index and meta files:

```
export * from '@oracle-cx-commerce/react-widgets';
export const IsvPaymentMethod = () => import('./isv-payment-method/index');
export const IsvCheckoutContinueToReviewOrderButton = () => import('./isv-checkout-continue-to-review-order-button');
export const IsvCheckoutPlaceOrderButton = () => import('./isv-checkout-place-order-button');
```

plugins/components/index.js

```
export * from '@oracle-cx-commerce/react-widgets/meta';
export {default as IsvPaymentMethod} from './isv-payment-method/meta';
export {default as IsvCheckoutContinueToReviewOrderButton} from './isv-checkout-continue-to-review-order-button/meta';
export {default as IsvCheckoutPlaceOrderButton} from './isv-checkout-place-order-button/meta';
```

plugins/components/meta.js

Copy plugins/endpoints into your storefront code and export the endpoints in the index and meta files:

```
export * from '@oracle-cx-commerce/endpoints';
export * from '@oracle-cx-commerce/oce-endpoints';
export const flexMicroformEndpoint = () => import('./flex-microform-endpoint');
export const paymentMethodConfigEndpoint = () => import('./payment-method-config-endpoint');
export const applePayValidationEndpoint = () => import('./apple-pay-validation-endpoint');
export const payerAuthSetupEndpoint = () => import('./payer-auth-setup-endpoint');
```

plugins/endpoints/index.js

```
export * from '@oracle-cx-commerce/endpoints/meta';
export * from '@oracle-cx-commerce/oce-endpoints';
export {default as flexMicroformEndpoint} from './flex-microform-endpoint/meta';
export {default as paymentMethodConfigEndpoint} from './payment-method-config-endpoint/meta';
export {default as applePayValidationEndpoint} from './apple-pay-validation-endpoint/meta';
export {default as payerAuthSetupEndpoint} from './payer-auth-setup-endpoint/meta';
```

plugins/endpoints/meta.js

Copy plugins/selectors into your storefront code and export the selector in the index file:

```
export * from './flex-microform-selector';
export * from './payment-method-config-selector';
```

plugins/selectors/index.js

Copy plugins/fetchers into your storefront code and export the fetchers in the hook, index and meta files:

```
export {default as useFlexMicroformFetcher} from './flex-microform-fetcher/hook';
export {default as usePaymentMethodConfigFetcher} from './payment-method-config-fetcher';
```

plugins/fetchers/hooks.js

```
export {default as flexMicroformFetcher} from './flex-microform-fetcher';
export {default as paymentMethodConfigFetcher} from './payment-method-config-fetcher';
```

plugins/fetchers/index.js

```
export {default as flexMicroformFetcher} from './flex-microform-fetcher/meta';
export {default as paymentMethodConfigFetcher} from './payment-method-config-fetcher/meta';
```

plugins/fetchers/meta.js

Note: Install jwt-decode package by running **'yarn add jwt-decode –W'**

- Deploy with the following command:

  yarn occ deploy

5.2. Add widget to the Checkout layout

Go to the OCC admin design tab and replace the default checkout-continue-to-review-order-button component with the IsvCheckoutContinueToReviewOrderButton on the checkout-payment page

Layouts > Checkout Payment - Default > checkout-payments-container

Drag  **IsvCheckoutContinueToReviewOrderButton** from the Components tray into the layout container.

- Go to the OCC admin design tab and replace the default checkout-place-order-button component with the IsvCheckoutPlaceOrderButton on the checkout-review-order page

  Layouts > Checkout Review Order - Default > checkout-review-order-container

  Drag  **IsvCheckoutPlaceOrderButton** from the Components tray into the layout container.

- Layout the IsvPaymentMethod component on the checkout payments container

  Layouts > Checkout Payment - Default > checkout-payments-container > checkout-payment-methods-container (shared)

  Remove the **CheckoutCreditCard** component from the layout and replace it with  **IsvPaymentMethod**

- Publish all changes

6. Test deployed widgets in OCC Storefront

In order to be able to see uploaded and configured widgets working in OCC Storefront you will need to publish changes:

- Go to OCC Admin -> Publishing
- Publish all recently introduced changes
- You should be able to see widgets in OCC Storefront afterwards (e.g.asbx80c1dev-store-{env}.oraclecloud.com)