

Workshop guide

Creado por Pablo Moreno, modificado por última vez Hace 31 minutos

This guide contains a few blocks that the attendees can follow to learn how to build an App Builder extension.

Access & Credentials

- **Adobe Experience Cloud IMS Organization:** AEP Partner Shared Training Sandbox
- Access to **Adobe Developer Console & App Builder:** <https://developer.adobe.com>
- **Adobe Commerce admin:** <https://staging-5em2ouy-ef02rc3kezxe.eu-4.magentositel.cloud/admin>
- **Base source code:** <https://github.com/adobe/app-builder-workshop/tree/skeleton>

Technical prerequisites

- node 14...18 LTS (Recommended v18)
- npm 9 or higher
 - @adobe/aio-cli >= 9.3.0

If you plan to develop App Builder locally using `aio app run --local`, then the following additional requirements are needed:

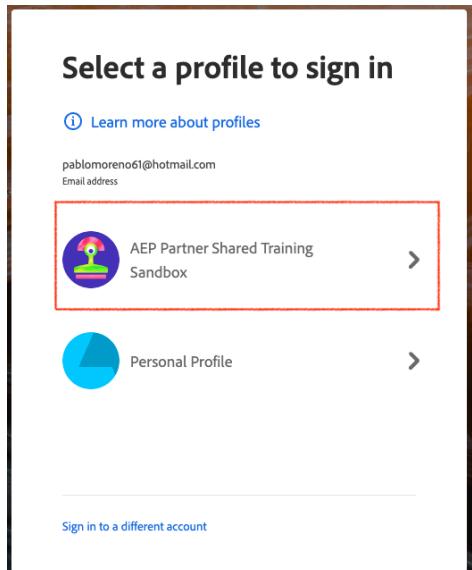
- Java 11 or Higher
- Docker 24 or higher

Official reference: https://developer.adobe.com/app-builder/docs/getting_started/

Add yourself to the Developer Console

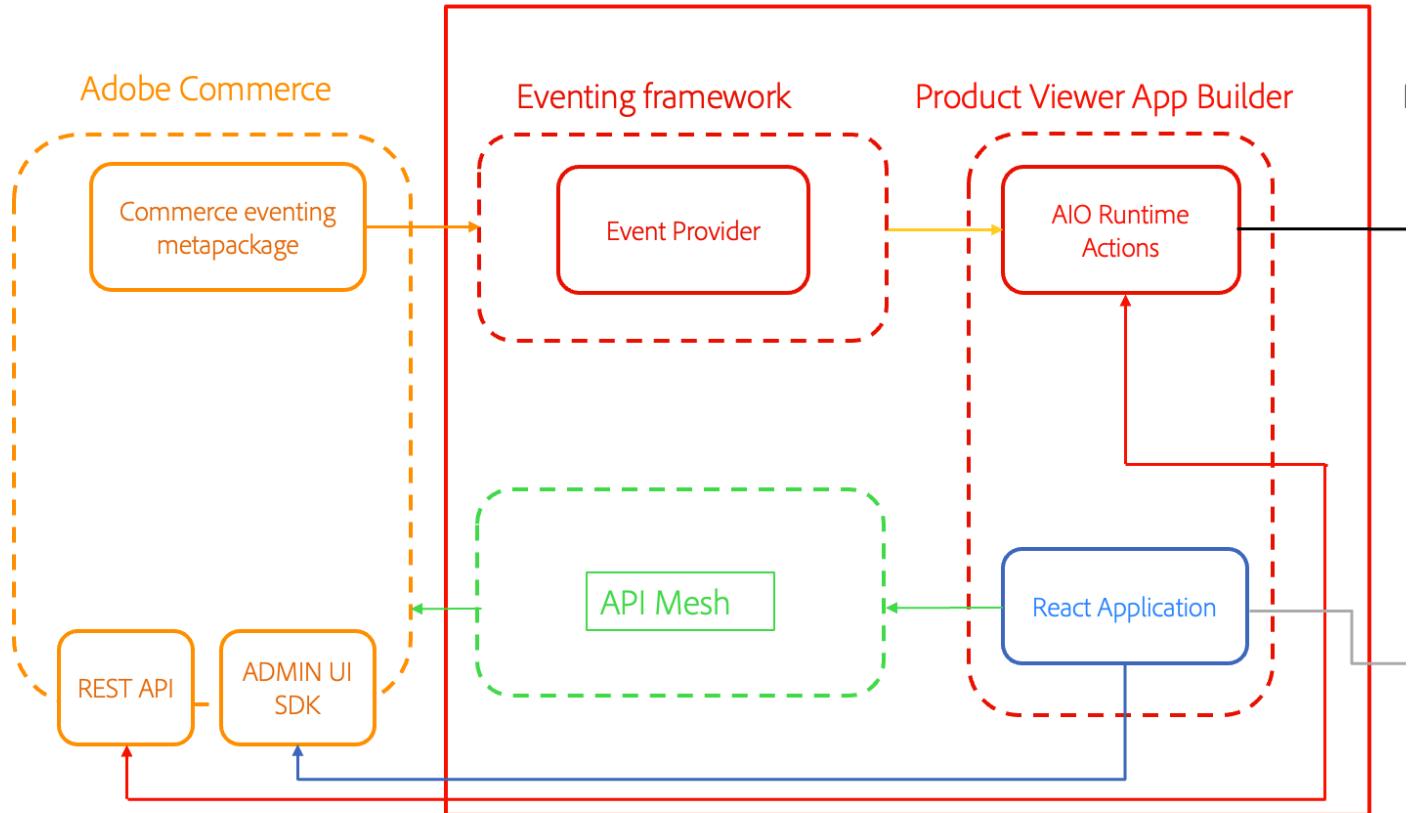
Accessing the developer console for the very first time will require to set up your account:

1. Using the guest e-mail you provided to the organizers, navigate to developer.adobe.com and click on the *Sign-in* link.
2. Fill out the form with your e-mail and click on Continue.
3. Click on the *Reset your password* link.
4. You will receive a verification code in your inbox, copy it onto the Verification code page and introduce your new password.
5. After setting your new password, you will receive a confirmation e-mail. you will be able to sign up using your guest e-mail.
 - a. If you have multiple profiles and the sign-up was properly done, you should see a profile selection window when accessing the Sign-in link from the developer.adobe.com page.



Project diagram

Technical Diagram



Setup your App Builder project

1. Log in to the Adobe Developer Console and select the "**AEP Partner Shared Training Sandbox**" organization from the dropdown menu in the top-right corner.

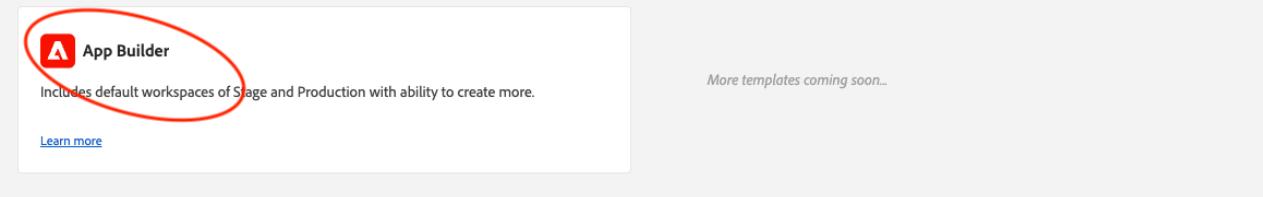
Screenshot of the Adobe Developer Console homepage:

- The top navigation bar shows "Adobe Developer Console" and the user role "Role: System Administrator | AEP Partner Shared Training Sandbox".
- The main message says "Welcome back Pablo. Let's start building."
- The "Welcome to the Adobe Developer Console" section provides an overview of the tools available.
- The "Quick start" section includes three buttons: "Create new project" (highlighted with a red oval), "Create project from template" (highlighted with a red oval), and "Download resources".
- At the bottom, it shows "AEP Partner Shared Training Sandbox's recent projects" and a link to "All projects".

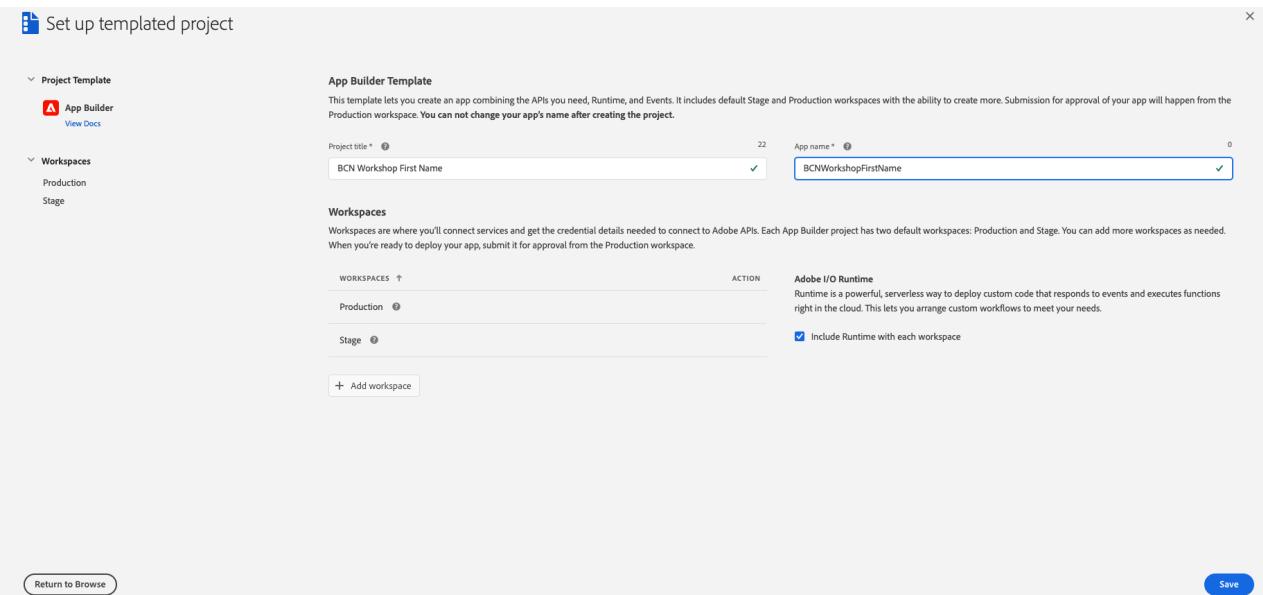
2. Click **Create project from template** and select **App Builder**.

Browse templates

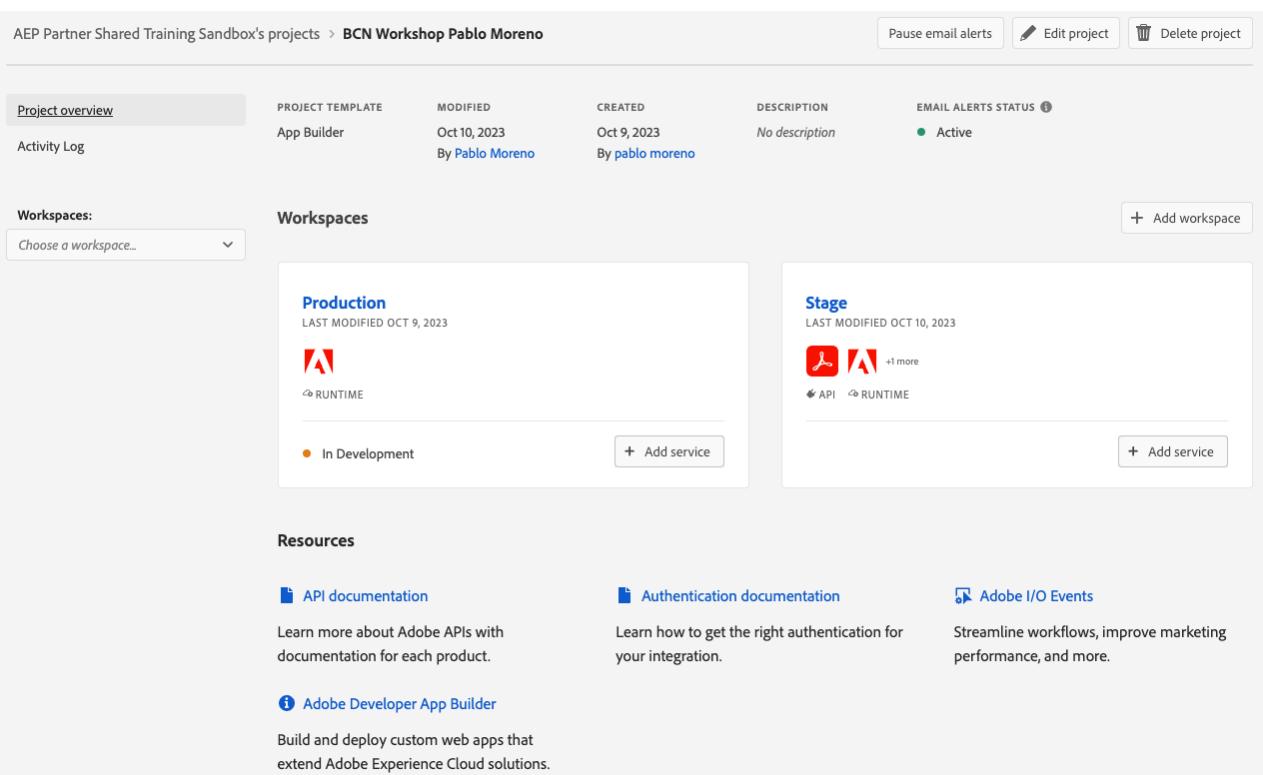
Find a project template that includes the services you need to get going faster. After selecting a template, you'll still need to customize some setup before your project is ready to go.



3. Specify a project title **BCN Workshop <Your name>** and an app name. Make sure to check the option **Include Runtime with each workspace**. Click **Save**.



4. The Console will create 2 workspaces for you, so you should see Production and Stage. **This session is going to be focused on Production workspace only.**



Prepare your code

1. Go to the path where you want to store the source code and **clone the base repo**.

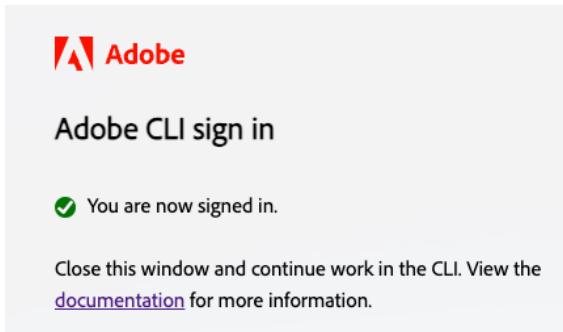
```
git clone -b skeleton https://github.com/adobe/app-builder-workshop.git
```

2. Log in to *aio cli* tool using your guest account.

- Run the following command

```
→ <you-project-path> ✘ aio login
```

- A new page should be opened in your browser, asking for sign-in. Use the credentials you provided during the "Add yourself to the Developer Console". Once logged in, you should see a screen as follows:



- Back to the CLI, you should verify that "Login successful" appears in the *aio login* response.

3. Select the active organization:

```
→ <you-project-path> ✘ aio console org select
? Select Org: AEP Partner Shared Training Sandbox
Org selected AEP Partner Shared Training Sandbox
You are currently in:
1. Org: AEP Partner Shared Training Sandbox
2. Project: <no project selected>
3. Workspace: <no workspace selected>
```

4. Then, select the active project:

```
→ <you-project-path> ✘ aio console project select
? Select Project: BCN Workshop Pablo Moreno
Project selected BCNWorkshopPabloMor
You are currently in:
1. Org: AEP Partner Shared Training Sandbox
2. Project: BCNWorkshopPabloMor
3. Workspace: <no workspace selected>
```

5. Select the active workspace:

```
→ <you-project-path> ✘ aio console workspace select
? Select Workspace: Production
Workspace selected Production
You are currently in:
1. Org: AEP Partner Shared Training Sandbox
2. Project: BCNWorkshopPabloMor
3. Workspace: Production
```

6. Finally, **import the App Builder project configuration** for the active namespace selected using the *aio cli* tool:

- This command will automatically generate 2 new files: *.aio* and *.env*.

```
→ <you-project-path> ✘ aio app use
You are currently in:
1. Org: <no org selected>
2. Project: <no project selected>
3. Workspace: <no workspace selected>

? Switch to a new Adobe Developer Console configuration: A. Use the global Org / Project / Workspace configuration:
  1. Org: AEP Partner Shared Training Sandbox
  2. Project: BCNWorkshopPabloMor
  3. Workspace: Production

✓ Successfully imported configuration for:
  1. Org: AEP Partner Shared Training Sandbox
  2. Project: BCNWorkshopPabloMor
  3. Workspace: Production.
```

7. Within the root path of your project, find the `.env.dist` file, copy its content and append it to the `.env` file. Don't worry about the empty variables, you will be asked to fill them out later on.

Here you have an example of how the `.env.dist` looks like.

```
# To fill during API Mesh implementation
API_MESH_URL=

# To fill after configuring PDF Embed API
SERVICE_PDF_EMBED_API_CLIENT_ID=

# Commerce auth. To fill before integrating Commerce Events
COMMERCE_BASE_URL=
COMMERCE_CONSUMER_KEY=
COMMERCE_CONSUMER_SECRET=
COMMERCE_ACCESS_TOKEN=
COMMERCE_ACCESS_TOKEN_SECRET=

# Randomizer. To fill before integrating Commerce Events
ATTENDEE_FULLNAME=

RANDOMIZER_URL=
RANDOMIZER_FIRSTNAME=
RANDOMIZER_LASTNAME=
RANDOMIZER_COMPANY_NAME=
RANDOMIZER_API_KEY=
```

8. Install NPM dependencies found in your `package.json` file:

```
npm install
```

Populate React UI component

Goals

- Render a basic React UI using static content.

Add a Products grid React component

1. Create a new React component in `/web-src/src/components/Products.js`.

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
import {
  Content,
  Heading,
  IllustratedMessage,
  TableView,
  TableHeader,
  TableBody,
  Column,
  Row,
  Cell,
  View,
  Flex,
  ProgressCircle
} from '@adobe/react-spectrum'
import React from "react";

export const Products = props => {
  const isLoadingCommerceProducts = false
  const commerceProducts = [{name:'Sample Product', id:1}]
  const productsColumns = [
    {name: 'Name', uid: 'name'},
    {name: 'SKU', uid: 'sku'},
    {name: 'Id', uid: 'id'},
    {name: 'Price', uid: 'price'},
```

```

        {name: 'Type ID', uid: 'type_id'},
        {name: 'Created At', uid: 'created_at'},
        {name: 'Updated At', uid: 'updated_at'},
    ]

    function renderEmptyState() {
        return (
            <IllustratedMessage>
                <Content>No data available</Content>
            </IllustratedMessage>
        )
    }

    return (
        <View>
            {isLoadingCommerceProducts ? (
                <Flex alignItems="center" justifyContent="center" height="100vh">
                    <ProgressCircle size="L" aria-label="Loading..." isIndeterminate/>
                </Flex>
            ) : (
                <View margin={10}>
                    <Heading level={1}>Fetched products from Adobe Commerce</Heading>
                    <TableView
                        overflowMode="wrap"
                        aria-label="products table"
                        flex
                        renderEmptyState={renderEmptyState}
                        minHeight="static-size-1000"
                    >
                        <TableHeader columns={productsColumns}>
                            {column => <Column key={column.uid}>{column.name}</Column>}
                        </TableHeader>
                        <TableBody items={commerceProducts}>
                            {product => (
                                <Row key={product['sku']}>{columnKey => (
                                    let content = product[columnKey];
                                    return <Cell key={`${product['sku']}-${columnKey}`}>{content}</Cell>
                                )}
                                </Row>
                            )}
                        </TableBody>
                    </TableView>
                </View>
            ) }
        </View>
    )
}

```

2. Import and add the newly created component to the Browser route of your App React component:

```
<Route index element={<Products/>} />
```

Validation

Display React UI using static content

Fetched products from Adobe Commerce

Name	SKU	Id	Price	Type ID	Created At	Updated At
Sample Product		1				

Query products from Commerce API using API Mesh

Goals

- Use Commerce graphQL API call to query products from the Commerce catalog using API Mesh.
- Replace static content with API Mesh response.

Add API Mesh support

Configure API Mesh for Production environment

1. Go to your workspace and click on Add Service → API. On that page, select the **API Mesh for Adobe Developer App Builder**.

Add an API

Add a REST API to your project to access Adobe services and products. Browse this API list and customize. Note that many services are only available through paid license or subscription. If you believe you should have access to a disabled service, please contact your Adobe sales representative.

View by Available to you

Adobe Commerce with Adobe ID

Requires Adobe review

Adobe Commerce with Adobe ID allows you to configure Adobe Commerce to use Adobe ID for authentication and authorization.

[View docs](#)

Adobe Express Embed SDK

Requires Adobe review

Embed powerful one-click image and video editing tools, plus templates, stock images and assets so your users can design anything they need easily...

[View docs](#)

Adobe I/O Events for Adobe Commerce

Create event-driven extensions and custom integrations using transactional data from Adobe Commerce.

[View docs](#)

Adobe Photoshop API

Edit images with Photoshop, Lightroom and Sensei Powered Image Cutout APIs. This provides trial access to all three cloud services.

[Learn more](#) [View docs](#)

Adobe Stock

Give your users access to stunning Adobe Stock high-quality images, graphics, and videos to use in their ads, email templates, and websites.

[View docs](#)

API Mesh for Adobe Developer App Builder



The API Mesh enables developers to configure Adobe or 3rd party API sources into a single GraphQL mesh. Developers can then easily run queries...

[View docs](#)

Creative Cloud Libraries

Applications closed

Please note we are not currently accepting new integrations. Existing Creative Cloud Libraries API users may continue to edit and manage existing...

[View docs](#)

Globalization Content Service

Globalization Content Service APIs enable a translation partner to fetch content and submit its translation

[View docs](#)

I/O Events

[View docs](#)

Next

2. Enter the allowed domain base URL, it should be your App Builder URL, following this example:

To fill out the form, get the `AIO_runtime_namespace` environment value from your `.env` file and append "`.adobeio-static.net`" to it.

Configure API

API

API Key

API Mesh for Adobe Developer App Builder

[View Docs](#)

Credentials

API Key

Allowed domain (up to 5 domains) *

675172-bcnworkshopdryrun.adobeio-static.net

Use wildcards to enter multiple subdomains (*.my-domain.com) or commas to separate multiple domains (www.domain-1.com, www.domain-2.com). During local development, you can include ports greater than 1023 with localhost (e.g. localhost:3000). Standard ports (80, 443) will be used for non-localhost domains.

Back

Save configured API

3. Click on Save configured API.

4. Create `/mesh.config.json` with the following content:

```
{
  "meshConfig": {
    "sources": [
      {
        "name": "CommerceGRAPHQL",
        "handler": {
          "graphql": {
            "endpoint": "https://staging-5em2ouy-efo2rc3kezxte.eu-4.magentosite.cloud/graphql/"
          }
        }
      }
    ],
    "responseConfig": {
      "CORS": {
        "methods": [
          "POST",
          "GET",
          "PUT",
          "HEAD",
          "OPTIONS"
        ],
        "origin": "*"
      }
    }
  }
}
```

```
    }
}
}
```

5. Install AIO API Mesh plugin.

```
aio plugins:install @adobe/aio-cli-plugin-api-mesh
```

6. Create a Mesh endpoint using the previous configuration file to get a **Mesh endpoint** for the current workspace.

```
aio api-mesh create mesh.config.json
```

7. In your `.env` file, add the following property **API_MESH_URL**:

```
API_MESH_URL=<YOUR_API_MESH_URL>
```

Code your API Mesh integration

1. Add a new function named `callMesh` inside the `/web-src/src/utils.js`:

```
export async function callMesh(url, pageSize, currentPage) {
  const res = await fetch(url, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(
      {
        query: `query {
          products(
            search: ""
            pageSize: ${pageSize}
            currentPage: ${currentPage}
            sort: {name: ASC}
          ) {
            items {
              id,
              sku,
              name,
              type_id,
              created_at,
              updated_at,
              price {
                regularPrice {
                  amount {
                    value
                  }
                }
                pdf_file
              }
            }
          }
        }` 
      )
    })
  return await res.json();
}
```

2. Create a hook in `/web-src/src/hooks/useCommerceProducts.js` to call API Mesh:

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
import { useEffect, useState } from 'react'
```

```

import { callMesh } from '../utils'

export const useCommerceProducts = props => {
  const [isLoadingCommerceProducts, setIsLoadingCommerceProducts] = useState(true)
  const [commerceProducts, setCommerceProducts] = useState([])

  const fetchCommerceProducts = async () => {

    const commerceProductsResponse = await callMesh(process.env.API_MESH_URL, props.pageSize, props.currentPage)
    console.log(`%s products returned by Commerce REST API`, commerceProductsResponse.data.products.items.length);

    commerceProductsResponse.data.products.items.forEach((item) => {
      item.price = item?.price?.regularPrice?.amount?.value;
    });

    setCommerceProducts(commerceProductsResponse.error ? [] : commerceProductsResponse.data.products.items)
  }

  useEffect(() => {
    fetchCommerceProducts().then(() => setIsLoadingCommerceProducts(false))
  }, [])

  return { isLoadingCommerceProducts, commerceProducts }
}

```

3. Finally, in your Products component, replace the static calls with **useCommerceProducts** call to load the content from API Mesh:

```
const {isLoadingCommerceProducts, commerceProducts} = useCommerceProducts({...props, pageSize: 20, currentPage: 1})
```

Validation

Fetch products using API Mesh

Fetched products from Adobe Commerce

Name	SKU	Id	Price	Type ID	Created At	Updated At
Abominable Hoodie	MH09	196	69	configurable	2023-10-10 09:46:34	2023-10-10 09:46:34
Adrienne Trek Jacket	WJ08	1322	57	configurable	2023-10-10 09:47:09	2023-10-10 09:48:17
Advanced Pilates & Yoga (Strength)	240-LV08	49	18	downloadable	2023-10-10 09:46:02	2023-10-10 09:46:02
Aeon Capri	WP07	1867	48	configurable	2023-10-10 09:47:27	2023-10-10 09:47:27
Aero Daily Fitness Tee	MS01	564	24	configurable	2023-10-10 09:46:48	2023-10-10 09:46:48
Aether Gym Pant	MP11	873	74	configurable	2023-10-10 09:46:59	2023-10-10 09:46:59
Affirm Water Bottle	24-UG06	15	7	simple	2023-10-10 09:45:44	2023-10-10 09:45:44
Aim Analog Watch	24-MG04	36	45	simple	2023-10-10 09:45:53	2023-10-10 09:45:53
Ajax Full-Zip Sweatshirt	MH12	244	69	configurable	2023-10-10 09:46:35	2023-10-10 09:46:35
Ana Running Short	WSH10	2023	40	configurable	2023-10-10 09:47:30	2023-10-10 09:47:30
Angel Light Running Short	WSH06	1996	42	configurable	2023-10-10 09:47:29	2023-10-10 09:47:29
Antonia Racer Tank	WT08	1802	34	configurable	2023-10-10 09:47:23	2023-10-10 09:47:23
Apollo Running Short	MSH02	904	32.5	configurable	2023-10-10 09:46:59	2023-10-10 09:46:59
Arcadio Gym Short	MSH11	1021	20	configurable	2023-10-10 09:47:01	2023-10-10 09:47:01
Argus All-Weather Tank	MT07	700	22	configurable	2023-10-10 09:46:52	2023-10-10 09:46:52
Ariel Roll Sleeve Sweatshirt	WH09	1168	39	configurable	2023-10-10 09:47:05	2023-10-10 09:47:05
Artemis Running Short	WSH04	1973	45	configurable	2023-10-10 09:47:29	2023-10-10 09:47:29

----- Coffee break -----

Integrate PDF Embed API service

Goals

- Do a third-party API call against the [PDF embed API](#) to display a PDF file.
- Display a React Button to view the content of the PDF file.

Add PDF Embed API support

Configure PDF Embed API for Production environment

- Select the Production environment.
- Click on Add Service → API and **select PDF Embed API** from the available options.

Add an API

Add a REST API to your project to access Adobe services and products. Browse this API list and customize. Note that many services are only available through paid license or subscription. If you believe you should have access to a disabled service, please contact your Adobe sales representative.

The screenshot shows a list of APIs categorized into several sections:

- Creative Cloud Libraries**: Shows a note that applications are closed. It includes a "View docs" link.
- I/O Management API**: Shows a "View docs" link.
- Primetime Ad Insertion**: Shows a note about managing AdRules. It includes a "View docs" link.
- Globalization Content Service**: Shows a note about translation partners. It includes a "View docs" link.
- I/O Events**: Shows a "View docs" link.
- Lightroom Services**: Shows a note about photo editing features. It includes a "View docs" link.
- PDF Embed API**: This section is highlighted with a blue border. It shows a note about a free client-side JavaScript API for embedding PDF documents. It includes a "View docs" link and a checked checkbox.

A "Next" button is visible at the bottom right of the page.

3. Enter the allowed domain base URL, it should be your App Builder URL, following this example:

To fill out the form, get the `AIO_runtime_namespace` environment value from your .env file and append ".adobeio-static.net" to it.

The screenshot shows the configuration page for the PDF Embed API:

- API Key**: A note states that some Adobe services require authorization but do not require authentication. It explains that an API key is the only client credential required for these services. A "Learn more" link is provided.
- Allowed domain ***: A text input field contains the value "675172-bcnworkshoppablomor-stage.adobeio-static.net". Below the input field is a note: "The PDF Embed API supports a single allowed domain."

At the bottom right, there are "Back" and "Save configured API" buttons.

4. After saving the configuration API, you will be redirected to the PDF Embed API page.

5. From the PDF Embed API page, get the API Key and replace the variable `SERVICE_PDF_EMBED_API_CLIENT_ID` value from your .env file with the API key (*client id*).

Configure PDF Embed API for the Local environment (Optional)

Only to be done if you plan to run your application locally (`aio app run`)

1. Go to <https://documentservices.adobe.com/dc-integration-creation-app-cdn/main.html?api=pdf-embed-api>
2. Fill the form with "**BCNWorkshop-Local-<YourName>**" name and **localhost** as a domain.

PDF Embed API > Get Credentials

Adobe PDF Embed API Credentials

Get unlimited access for free

Create new credentials All fields are required

Credentials Name i

bcnworkshopElocal



Provide a unique descriptive name for the credentials

8

Application Domain i

localhost

www.example.com

 By creating credentials, you are agreeing to our [developer terms](#).[Create credentials](#)Have existing credentials? [Manage in developer console](#)3. Finally, click on **Create credentials** and add the client ID generated to your .env file as SERVICE_PDF_EMBED_API_CLIENT_ID.a. Note: **Do not replace the previous PDF Embed API key generated for the Production workspace, instead keep both keys in your .env file and activate only the key needed on each case. Keep in mind this note before deploying the code.**

Code your PDF Embed API integration

1. Import PDF viewer into your `/web-src/index.html`:

```
<script type="text/javascript" src="https://acrobatservices.adobe.com/view-sdk/viewer.js"></script>
```

2. Create a new file `/web-src/src/libraries/ViewSDKClient.js` to interact with the PDF Embed API service:

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
class ViewSDKClient {
    constructor() {
        this.readyPromise = new Promise((resolve) => {
            if (window.AdobeDC) {
                resolve();
            } else {
                /* Wait for Adobe Acrobat Services PDF Embed API to be ready */
                document.addEventListener("adobe_dc_view_sdk.ready", () => {
                    resolve();
                });
            }
        });
        this.adobeDCView = undefined;
    }

    ready() {
        return this.readyPromise;
    }

    previewFile(divId, filename, viewerConfig) {
        const config = {
            clientId: process.env.SERVICE_PDF_EMBED_API_CLIENT_ID,
        };
        if (divId) { /* Optional only for Light Box embed mode */
            /* Pass the div id in which PDF should be rendered */
            config.divId = divId;
        }
        /* Initialize the AdobeDC View object */
        this.adobeDCView = new window.AdobeDC.View(config);
    }
}
```

```

const url = new URL(filename);

/* Invoke the file preview API on Adobe DC View object */
return this.adobeDCView.previewFile({
    content: {
        location: {
            url: url.toString(),
        },
    },
    metaData: {
        fileName: filename,
        id: crypto.randomUUID().toString(),
    }
}, viewerConfig);
}

export default ViewSDKClient;

```

3. Import the ViewSDKClient library into your Products React component

```
import ViewSDKClient from "../libraries/ViewSDKClient";
```

4. Then, inside your Products React component, add a button handler to **handle the click on the "View PDF" button** at the beginning of the component.

```

const handleOnPress = (pdfFilename) => {
    if (pdfFilename === '') {
        return;
    }

    const viewSDKClient = new ViewSDKClient();
    viewSDKClient.ready().then(() => {
        viewSDKClient.previewFile("", pdfFilename, {
            embedMode: "LIGHT_BOX"
        });
    });
};

```

5. Add a new column to the table to display the PDF button:

```
{name: 'Attachments', uid: 'pdf_file'},
```

6. Within the same React component, add a button to trigger the PDF Embed API call inside the <TableBody> loop.

```

if (columnKey === 'pdf_file' && content) {
    content = (
        <Button type="button" variant={'primary'} onPress={() => handleOnPress(product[columnKey])}>View PDF</Button>
    );
}

```

7. Finally, add a layer to embed the PDF content. The content of the PDF file will be opened inside this layer

```
<div id="container" className="light-box-container"></div>
```

Validation

List of products with PDF button

Fetched products from Adobe Commerce

SKU	Name	Status	Price	Type ID	Created At	Updated At	Attachments
24-MB01	Joust Duffle Bag	1	34.3	simple	2023-10-10 09:45:34	2023-10-16 14:44:27	View PDF
24-MB04	Strive Shoulder Pack	1	32.01	simple	2023-10-10 09:45:35	2023-10-13 14:49:11	View PDF
24-MB03	Crown Summit Backpack	1	38	simple	2023-10-10 09:45:36	2023-10-10 09:45:36	
24-MB05	Wayfarer Messenger Bag	1	45	simple	2023-10-10 09:45:37	2023-10-10 09:45:37	

After clicking on the View PDF button



References

- <https://developer.adobe.com/document-services/docs/overview/pdf-embed-api/>

Deal with Commerce Events using the Eventing Framework

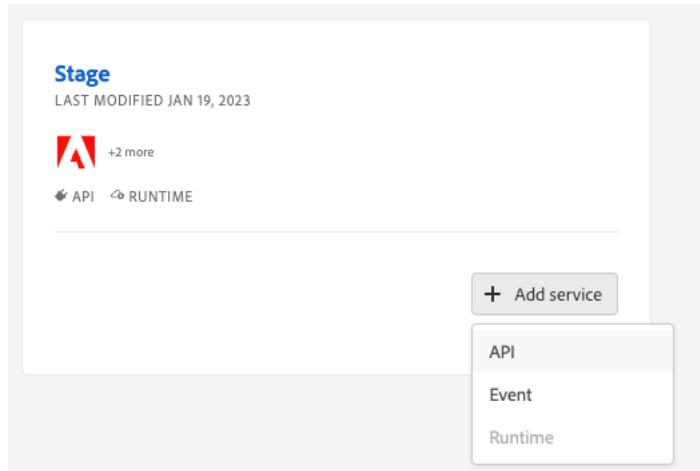
Goals

- Handle Commerce events.
- Create a runtime action that gets triggered by a product updated Commerce event:
 - Calls Randomizer API to generate a random number.
 - Call back to Commerce API to store the generated number as a catalog attribute for the product updated.

Add I/O Events support

Configure the required I/O Services in Production

1. In your workspace, click the **Add service** pop-up menu and select **API**.



2. On the **Add an API** page, filter on **Adobe Services** and select **I/O Management API**. Then click **Next**.
Note: This service is required to add the appropriate authentication scopes used by App Builder.

Add an API

Add a REST API to your project to access Adobe services and products. Browse this API list and customize. Note that many services are only available through paid license or subscription. If you believe you should have access to a disabled service, please contact your Adobe sales representative.

Filter by product



Adobe Experience Platform



Adobe Sensei



Adobe Services



Creative Cloud



Document Cloud



Experience Cloud

View by Available to you

Adobe Commerce with Adobe ID

Requires Adobe review

Adobe Commerce with Adobe ID allows you to configure Adobe Commerce to use Adobe ID for authentication and authorization.

[View docs](#)

Adobe Status API

Adobe Status API's provide detailed information and real time updates about Adobe cloud products and services outage, disruption and maintenance...

[View docs](#)

Globalization Content Service

Globalization Content Service (GCS) is a set of micro-services which enable content globalization for individuals, SMBs and Enterprises.

[View docs](#)

I/O Events

[View docs](#)

I/O Management API



[View docs](#)

Sign In with Adobe

Requires Adobe review

Sign In With Adobe allows you to sign in Adobe users and get a unique ID for them.

[View docs](#)

User Management API

[View docs](#)

Next

3. On the **Configure API** page, select the **OAuth Server-to-Server** option, fill the credential name and click **Save configured API**.

Configure API

API

I/O Management API

[View Docs](#)

Credentials

OAuth Server-to-Server

Choose type of authentication you need

Server-to-Server authentication allows your application to call Adobe services on behalf of the application or your enterprise organization. User authentication allows your application to call Adobe services on behalf of a signed-in user. Choose your credential type -

SERVER-TO-SERVER AUTHENTICATION

OAuth Server-to-Server

SERVER-TO-SERVER AUTHENTICATION

DEPRECATED

Service Account (JWT)

This credential allows you to use industry standard OAuth2.0 libraries to generate access tokens using the OAuth 2.0 client credentials grant type. [Learn more](#)

This credential involves creating a JSON Web Token (JWT) and exchanging that JWT for an access token. [Learn more](#)

OAUTH2.0 CLIENT CREDENTIALS

Credential name *

Credential name

Name your credential to identify it easily on the Admin Console under Users > API Credentials. The credential name must be unique to your organization. You can modify it later on the credential details screen.

[Return to Browse](#)

[Save configured API](#)

4. On the front page of your workspace, click the **Add service** pop-up menu and select **API**.

Project overview

Activity Log

Workspaces:

test

+ Add service

API

Event

Runtime

CREDS

OAuth Server-to-Server

APIS

I/O Management API

Get started

Read through our get started guides to understand the steps for working with this API.

[View documentation](#)



Try it out

Experiment with this API in Postman and get a feel for how it works.

[Download for Postman](#)

Connected credentials

OAuth Server-to-Server

CREDENTIAL

API KEY (CLIENT ID)

Copy

Generate access token

5. On the **Add an API** page, filter by **Experience Cloud** and select **Adobe I/O Events for Adobe Commerce**. OAuth Server-to-Server should be determined by default.

The screenshot shows the 'Configure API' dialog. In the top left, there's a gear icon and the title 'Configure API'. Below it, a tree view shows 'API' expanded, with 'Adobe I/O Events for Adobe Commerce' selected. Underneath, there are 'View Docs' and 'Credentials' sections. The 'Credentials' section has 'OAuth Server-to-Server' selected, with a status bar indicating it's an 'Existing' credential. A note explains that server-to-server authentication allows calling Adobe services on behalf of the application or organization. A credential named 'OAuth Server-to-Server' is listed, created by 'BCN Workshop Pablo Moreno - Stage'. At the bottom, there are 'Return to browse' and 'Save configured API' buttons.

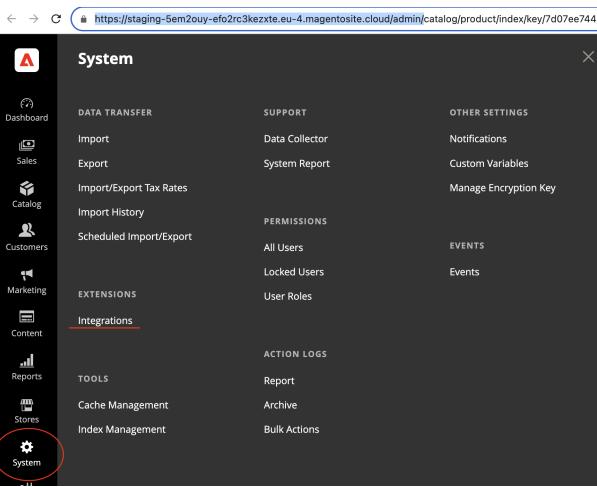
Then click Next and **Save configured API**.

6. Finally, you will need to **download the workspace configuration file**. Go to the overview page of your workspace and click the Download All button in the top-right corner.

The screenshot shows the workspace overview page. At the top, it displays 'AEP Partner Shared Training Sandbox's projects > BCN Workshop Pablo Moreno > Stage'. To the right are buttons for 'Activate email alerts', 'Download all' (which is circled in red), and 'Delete Workspace'. The main area is divided into sections: 'Project overview' (MODIFIED Oct 13, 2023, CREATED Oct 9, 2023, DESCRIPTION No description, EMAIL ALERTS STATUS Paused), 'Workspaces:' (Stage, + Add service), 'Products & services' (I/O Management API, Adobe I/O Events for Adobe Commerce), and 'CREDENTIALS' (OAuth Server-to-Server). The 'OAuth Server-to-Server' section shows a credential with an API key ('b733a4eb8a544f5d94d630d00089c54d') and a 'Copy' button. There are also 'Generate access token' and three-dot ellipsis buttons.

Populate .env configuration

1. Go to the commerce instance (<https://staging-5em2ouy-efo2rc3kezxe.eu-4.magentosite.cloud/admin/>).
2. Navigate to System → Integrations



3. Click on Edit button to the **BCN Workshop Oct** integration

BCN Workshop Oct	⚠ Integration not secure	Active	Reauthorize	Edit
------------------	---	--------	-------------	--

4. You will find the **Commerce credentials** under the Integration Details section. Use them to populate the .env file variables prefixed with **COMMERCE_***.
 a. Add the Commerce URL <https://staging-5em2ouy-ef02rc3kezxe.eu-4.magentosite.cloud/> to the **COMMERCE_BASE_URL** as well.
 5. Populate the .env variables prefixed with **RANDOMIZER_*** to allow calls to the 3rd party API that is going to be used from the event handler.

```
RANDOMIZER_URL=https://675172-bcnworkshoprandom-stage.adobeio-static.net/api/v1/web/randomizer-app-builder/eventRegister
RANDOMIZER_FIRSTNAME=<Your name>
RANDOMIZER_LASTNAME=<Your last name>
RANDOMIZER_COMPANY_NAME=<Your company name>
RANDOMIZER_API_KEY=1WantT0Random!ze
```

Handling the event in code

1. Go to the source code of your project.
2. Create an *index.js* javascript file under the *actions/randomizer* directory.

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
import {getCommerceOauthClient} from "../oauthlib";

const { Core } = require('@adobe/aio-sdk')
const { ErrorResponse, checkMissingRequestInputs } = require('../utils')
import axios from 'axios';

export async function main(params) {
    const logger = Core.Logger('main', { level: params.LOG_LEVEL || 'info' });

    try {
        const requiredParams = ['COMMERCE_BASE_URL', 'RANDOMIZER_URL', 'RANDOMIZER_API_KEY', 'RANDOMIZER_FIRSTNAME', 'RAN
        const requiredHeaders = [];
        const errorMessage = checkMissingRequestInputs(params, requiredParams, requiredHeaders);

        if (errorMessage) {
            return ErrorResponse(400, errorMessage, logger);
        }

        const res = await axios.post(params.RANDOMIZER_URL, {
            "firstname": params.RANDOMIZER_FIRSTNAME,
            "lastname": params.RANDOMIZER_LASTNAME,
            "company_name": params.RANDOMIZER_COMPANY_NAME,
            "api_key": params.RANDOMIZER_API_KEY
        });

        const randomizedContent = await res.data;

        if (!randomizedContent?.random) {
```

```

        throw new Error("Random number not returned");
    }

logger.info("Random number returned: %s", randomizedContent.random);

const oauth = getCommerceOauthClient(
    {
        url: params.COMMERCE_BASE_URL,
        consumerKey: params.COMMERCE_CONSUMER_KEY,
        consumerSecret: params.COMMERCE_CONSUMER_SECRET,
        accessToken: params.COMMERCE_ACCESS_TOKEN,
        accessTokenSecret: params.COMMERCE_ACCESS_TOKEN_SECRET
    },
    logger
)

const attributeOptions = await oauth.get('products/attributes/random_numbers/options');

logger.info("Got custom attribute options: %s", JSON.stringify(attributeOptions).toString());

// Add random number to custom attributes if not already there
const option = attributeOptions.find(o => o.label === randomizedContent.random.toString());

if (!option) {
    await oauth.post('products/attributes/random_numbers/options', {
        option: {
            "value": randomizedContent.random.toString(),
            "label": randomizedContent.random.toString()
        }
    });
    logger.info("Random number added to custom attributes: %s", randomizedContent.random);
}

return {
    statusCode: 200,
    body: {
        "random": randomizedContent.random
    }
}
} catch (error) {
    logger.error("Cannot process catalog updated event: ", error);
    return errorResponse(500, error, logger);
}
}
}

```

3. Add the following config as a child of ***runtimeManifest*** property in your *app.config.yaml* file:

```

packages:
  BcnAppBuilderWorkshop:
    license: Apache-2.0
    actions:
      randomizer:
        function: actions/randomizer/index.js
        web: 'yes'
        runtime: 'nodejs:16'
        inputs:
          COMMERCE_BASE_URL: $COMMERCE_BASE_URL
          COMMERCE_CONSUMER_KEY: $COMMERCE_CONSUMER_KEY
          COMMERCE_CONSUMER_SECRET: $COMMERCE_CONSUMER_SECRET
          COMMERCE_ACCESS_TOKEN: $COMMERCE_ACCESS_TOKEN
          COMMERCE_ACCESS_TOKEN_SECRET: $COMMERCE_ACCESS_TOKEN_SECRET
        LOG_LEVEL: debug
        RANDOMIZER_FIRSTNAME: $RANDOMIZER_FIRSTNAME
        RANDOMIZER_LASTNAME: $RANDOMIZER_LASTNAME
        RANDOMIZER_COMPANY_NAME: $RANDOMIZER_COMPANY_NAME
        RANDOMIZER_URL: $RANDOMIZER_URL
        RANDOMIZER_API_KEY: $RANDOMIZER_API_KEY
    annotations:
      require-adobe-auth: true
      final: true

```

Configure the required event provider in Production

1. Deploy your code to the Production environment, so it's required in order to associate the event received to a handler.

aio app deploy

- From your workspace. Click the **Add service** pop-up menu and select **Event**.

The screenshot shows the 'Products & services' interface. On the left, there is a sidebar titled 'Workspaces:' with a dropdown set to 'Stage'. Below it are three buttons: '+ Add service', 'API', 'Event' (which is highlighted in blue), and 'Runtime'. To the right, a large central area is titled 'I/O Management API' with a sub-section 'API'. At the bottom right of this area is a small '...' button.

- On the **Add events** page, select the event provider "**Commerce Events**" to handle events triggered by Adobe Commerce. Then click **Next**.

The screenshot shows the 'Add events' page. At the top, there is a header with a back arrow and the title 'Add events'. Below it, a sub-header says 'Subscribe to events to get near real-time updates on content and data from Adobe services and products.' A 'Filter by product' section shows icons for Creative Cloud, Experience Cloud, and Experience Platform. The main area contains several event providers listed in cards:

- Adobe Express Docs**: Receive notifications and respond to changes made to Adobe Express Documents.
- Adobe Illustrator Cloud Documents**: Adobe Illustrator Cloud Document Events.
- Asset Compute**: Asset Compute Events.
- Cloud Manager Events**: Cloud Manager events provider (gain insight into actions and status emitted by the Cloud Manager CI/CD pipeline). This card has a checked checkbox next to its name.
- Commerce Events**: Commerce Eventing Framework. This card has a checked checkbox next to its name.
- Creative Cloud Assets**: Creative Cloud Asset Events Provider.
- Creative Cloud Libraries**: Receive notifications and respond to changes made to Creative Cloud Libraries.
- Data Hygiene**: Data lifecycle management tool to delete and update records as well as to set data expiration dates, eliminating data once it is expired.
- Globalization Content Service**: GCS Events for consuming translation related content.

At the bottom right of the page is a 'Next' button.

- Select the commerce instance from which you want to receive events.

The screenshot shows the 'Configure event registration' page. At the top, there is a header with a back arrow and the title 'Configure event registration'. On the left, there is a sidebar with two sections: 'Event Providers' (which is expanded) and 'Instances' (which is collapsed). Under 'Event Providers', there is a single item: 'Commerce Events'. Under 'Instances', there is one item: 'staging-bcn-workshop-provider' with the identifier 'DX_COMMERCER_EVENTS'. In the main area, there is a section titled 'Choose your instance' with the sub-instruction 'Select the Commerce Events instance from which you want to receive notifications. In the next step, you can choose which events you'd like to receive notifications about.' Below this, there is a table with columns 'SUBSCRIBE' (with a checked checkbox), 'INSTANCE ID' (containing 'staging-bcn-workshop'), and 'NAME' (containing 'staging-bcn-workshop-provider'). At the bottom of the page are 'Return to Browse' and 'Next' buttons.

- Select the events to subscribe to. In this case, we are interested in the catalog updated event. Then click **Next**.

Configure event registration

- Event Providers
 - Commerce Events
- Instances
 - staging-bcn-workshop-provider
DX_COMMERCER_EVENTS
- Subscribed Events
 - Observer event catalog_product_save_after
STAGING-BCN-WORKSHOP-PROVIDER

Choose event subscriptions

Select which events you'd like to receive notifications about from your chosen event providers. In the next steps, you can decide how to receive these, how to create multiple events registrations, and more.

SUBSCRIBE	NAME	PROVIDER	INSTANCE	DESCRIPTION
<input checked="" type="checkbox"/>	Observer event catalog_product...	Commerce Events	staging-bcn-workshop-provider	Observer event catalog_product_save_after

[Return to Browse](#) [Back](#) [Next](#)

6. Select *Oauth Server-to-Server* as the authentication method, selected by default.

Configure event registration

- Event Providers
 - Commerce Events
- Instances
 - staging-bcn-workshop-provider
DX_COMMERCER_EVENTS
- Subscribed Events
 - Observer event catalog_product_save_after
STAGING-BCN-WORKSHOP-PROVIDER
- Credentials
 - OAuth Server-to-Server
 - Credential name: OAuth Server-to-Server
 - Credential in BCN Workshop Pablo Moreno - Stage

Choose type of authentication you need

Server-to-Server authentication allows your application to call Adobe services on behalf of the application or your enterprise organization. User authentication allows your application to call Adobe services on behalf of a signed-in user. Choose your credential type -

SERVER-TO-SERVER AUTHENTICATION

OAuth Server-to-Server Existing

This credential allows you to use industry standard OAuth2.0 libraries to generate access tokens using the OAuth 2.0 client credentials grant type. [Learn more](#)

[OAUTH2.0 CLIENT CREDENTIALS](#)

[Return to Browse](#) [Next](#)

7. Update the **Event registration name** and **Event registration description** fields. In the **How to receive events** section, under **OPTION 2**, select the runtime action you created in the previous section and click **Save configured events**.

Note. If there are no actions in the Runtime action dropdown, review the step where you deployed the application

Configure event registration

- Instances
 - staging-bcn-workshop-provider
DX_COMMERCER_EVENTS
- Subscribed Events
- Credentials
- Event registration details

Event registration name * Randomizer event registry

Event registration description This event will trigger the Randomizer service and return value to Commerce|

How to receive events

Journaling
Adobe's Journaling API collects all events for a specific time period. A unique endpoint will be generated after this registration is saved.
[Learn more](#)

Add an optional way to receive events
After the registration is saved, it will have an auto-generated Journaling URL. You also have the option to add a webhook URL or Runtime action to receive notifications about events.

OPTION 1

Webhook
Register a webhook by specifying a webhook URL and how often you'd like to receive the events. Each event will result in a HTTP request to this URL, notifying your application

OPTION 2

Runtime action
Select a Runtime action for where to consume your events. If you don't have Runtime set up, [view our documentation](#) to learn more about Runtime and how to deploy your first action.

Runtime Action: BcnAppBuilderWorkshop/_secured_rand...

[Return to Browse](#) [Back](#) [Save configured events](#)

At this point, you are ready to visit the Commerce admin (<https://staging-5em2ouy-ef02rc3kezxt.eu-4.magentosite.cloud/admin>) and update a product to trigger an event.

Validation

Event reaching your project

The screenshot shows the 'Randomizer event registry' page in the AEP Partner Shared Training Sandbox. The top navigation bar includes 'Edit Events Registration' and 'Delete Events Registration'. Below the navigation, there are tabs for 'Project overview', 'Registration Details', 'Debug Tracing' (which is selected), and 'Event Browser'. Under 'Activity Log', there is a dropdown menu set to 'Stage' and a search/filter section with 'All traces' and '+ Add filters' buttons. A 'Refresh list' button is also present. The main area displays a table of event logs:

DELIVERY TIME (UTC)	EVENT CODE	EVENT PROVIDER	RESPONSE CODE
> 2023/10/16 14:44:36	Observer event catalog_product_save_after	staging-bcn-workshop-provider	200
> 2023/10/16 10:11:35	Observer event catalog_product_save_after	staging-bcn-workshop-provider	200
> 2023/10/16 10:10:39	Observer event catalog_product_save_after	staging-bcn-workshop-provider	200

Getting a successful response

This screenshot shows a detailed view of an event response from the 'Randomizer event registry' page. The interface is similar to the previous one, with 'Edit Events Registration' and 'Delete Events Registration' buttons at the top. The 'Debug Tracing' tab is selected. The event log table shows a single entry with a delivery time of '2023/10/16 14:44:36', event code 'Observer event catalog_product_save_after', provider 'staging-bcn-workshop-provider', and response code '200'. Below the table, a 'Request' and 'Response (200)' section is expanded. The 'Request' section shows the raw HTTP request headers, which include various CORS-related headers like 'Access-Control-Allow-Headers', 'Access-Control-Allow-Methods', and 'Access-Control-Allow-Origin'. The 'Response' section shows the raw JSON response body: '{ "random": 1276 }'. A note indicates 'REQUEST COMPLETED IN: 4082MS'.

Product attribute created

Verify that the received number in the previous step is created in the Commerce Attribute page for the **random_numbers** attribute.

The screenshot shows the 'random_numbers' attribute page in the Adobe Commerce interface. At the top, there is a header with a search bar, a user icon, and a save button labeled 'Save Attribute'. Below the header, there are two tabs: 'ATTRIBUTE INFORMATION' (selected) and 'Attribute Properties'. The 'ATTRIBUTE INFORMATION' tab contains sections for 'Properties' (with 'Default Label' set to 'Random numbers'), 'Manage Labels', and 'Storefront Properties'. The 'Attribute Properties' tab contains sections for 'Catalog Input Type for Store Owner' (set to 'Dropdown') and 'Values Required' (set to 'No'). Below these tabs is a section titled 'Manage Options (Values of Your Attribute)'. It shows a table with columns for 'Is Default', 'Admin', 'Default Store View', and 'Delete'. There is one option listed: '1276'. A 'Delete' button is visible next to the option.

References

- <https://developer.adobe.com/commerce/extensibility/events/project-setup/>

Render App Builder extension in Commerce using Admin UI SDK

Goals

- Display a new option in the Adobe Commerce menu linking to our App Builder extension.
- Display our App Builder extension inside Adobe Commerce layout.

Publish the App Builder project

- Note: If you were using the PDF Embed API key generated for the local environment, switch back to your production API key or you won't be able to the PDFs in the commerce UI.

1. Add Nodejs library **uix-guest** to the package.json file to allow communication between Adobe Commerce and our App Builder extension.

```
"@adobe/uix-guest": "^0.8.2"
```

2. Run this command to install the new package

```
npm install
```

3. Fill in the ATTENDEE_FULLNAME environment variable:

Note: This value will be used as part of the new menu option that will be added to the Adobe Commerce menu.

```
ATTENDEE_FULLNAME=<Your name>
```

4. Create **install.yaml** file in the root of your project. This file is used to create and configure remote Developer Console resources.

```
$schema: http://json-schema.org/draft-07/schema
$id: https://adobe.io/schemas/app-builder-templates/1

categories:
  - action
  - ui

extensions:
  - extensionPointId: commerce/backend-ui/1
```

5. Create **extension-manifest.yml** file at the root of your project. This file contains data about your extension that allows Adobe Experience Platform to properly consume it.

```
{
  "name": "bcn-app-builder-workshop",
  "displayName": "BCN App Builder Workshop",
  "description": "Dive into the extensibility capabilities of Adobe Commerce",
  "platform": "web",
  "id": "bcn-app-builder-workshop-app",
  "version": "1.0.0"
}
```

6. In the **/web-src/src/components** directory, you should create the **Constants.js** file

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/

module.exports = {
  extensionId: 'bcn_app_builder_workshop_app' + process.env.ATTENDEE_FULLNAME.replace(' ', '_').toLowerCase(),
  rootElementId: 'backend-ui',
};
```

7. In the **/web-src/src/components** directory, you should create the **ExtensionRegistration.js** file

```
/*
```

Copyright 2023 Adobe. All rights reserved.
This `file` is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this `file` except in compliance with the License. You may obtain a copy
of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.

*/

```
import { register } from '@adobe/uix-guest'
import { extensionId } from './Constants'

export default function ExtensionRegistration() {
  init().catch(console.error);
  return <></>;
}

const init = async () => {
  await register({
    id: extensionId,
    methods: {
      menu: {
        getItems() {
          return [
            {
              id: `${extensionId}`,
              title: 'App Builder Workshop ' + process.env.ATTENDEE_FULLNAME,
              parent: 'Magento_Backend::marketing'
            }
          ]
        },
        page: {
          getTitle() {
            return 'App Builder Workshop ' + process.env.ATTENDEE_FULLNAME
          }
        }
      }
    }
  })
}
```

8. Add ExtensionRegistration to the **Routes** in the App.js file

```
<Route path={'index.html'} element={<ExtensionRegistration />} />
```

9. Deploy the App into the Production workspace:

- Deploy your code to Prod:

```
→ <you-project-path> × aio app deploy
```

- Check if your deployed application is working properly. Retrieve the `AIO_runtime_namespace` environment value from your `.env` file and append "`.adobeio-static.net`" to it,
Ex: `675172-bcnworkshopjoao.adobeio-static.net`

10. Publish the App in the Developer Console

- Go into the Production workspace and select **Workspace Overview** on the left menu.

Project overview

PROJECT TEMPLATE

App Builder

MODIFIED

Oct 10, 2023

By Pablo Moreno

CREATED

Oct 9, 2023

By pablo moreno

DESCRIPTION

No description

EMAIL ALERTS STATUS

● Active

Activity Log

Workspaces:

Choose a workspace... ▾

Workspaces

+ Add workspace

Production

LAST MODIFIED OCT 9, 2023



↳ RUNTIME

● In Development

+ Add service

Stage

LAST MODIFIED OCT 10, 2023



+2 more

↳ API ↳ RUNTIME

+ Add service

Resources

API documentation

Learn more about Adobe APIs with documentation for each product.

Authentication documentation

Learn how to get the right authentication for your integration.

Adobe I/O Events

Streamline workflows, improve marketing performance, and more.

Adobe Developer App Builder

Build and deploy custom web apps that extend Adobe Experience Cloud solutions.

b. Click on **Submit for Approval**.c. Scroll down to fill out the App Submission form and **press Submit**.*You can get the App icon image from the /resources/ directory in your source code.*

App Submission Details
ALL FIELDS ARE REQUIRED

App Details
This will be visible to people using or reviewing your app. The app name will be used for the custom URL for your app.

App title *

App description *

Contact email *

App Icon
You can add an icon so people can easily identify your app. Please upload a PNG or JPG file that's 512x512 px and no larger than 100 kb in size.

App icon *

Note to Reviewer(s)
If you have any further notes or context for your app's reviewer, like login information, please include that here.

Note to reviewer *

Submit

d. Your app will be submitted for approval. At this point, **ask any of the organizers for approval**.

Production
LAST MODIFIED OCT 10, 2023

+2 more

↳ API ↳ RUNTIME

● Published

+ Add service

e. Once the app has been approved, you will receive an e-mail "Adobe Developer App Builder App Approved" meaning that your application has been published. Also, in your Workspaces Overview, you should notice that the **status has changed to Published**.

11. Finally, go to the Commerce admin instance to verify that a **new menu option is available under the Marketing tab**.

The screenshot shows the Adobe Commerce Admin interface. On the left, there's a sidebar with various navigation options like Dashboard, Sales, Catalog, Customers, Marketing, Content, Reports, Stores, System, and Find Partners & Extensions. The Marketing tab is selected. In the main content area, there's a 'Marketing' header with three columns: PROMOTIONS, COMMUNICATIONS, and USER CONTENT. Under COMMUNICATIONS, there are links for Email Templates, Newsletter Templates, All Reviews, Pending Reviews, Newsletter Queue, Newsletter Subscribers, and Email Reminders. The 'Email Reminders' link is circled in red. Below this is a section for PRIVATE SALES with links for Events, Invitations, SEO & SEARCH, URL Rewrites, Search Terms, Search Synonyms, and Site Map. At the bottom of the page, there's a message: 'We couldn't find any records.' To the right, there's a yellow bar with a 'Reload Data' button and a 'System Messages: 1' notification. A blue button at the bottom right says 'Go to Advanced Reporting'.

12. Clicking on the new option, the one with the name of the App you defined, should **display your App Builder extension embedded in Commerce**.

References

<https://developer.adobe.com/commerce/extensibility/admin-ui-sdk/troubleshooting/>

Sin etiquetas