FRUITION

The future begins today...

USER GUIDE

Product Labels Manager

Fruition Consultancies
info@fruitionconsultancies.com

Contents

Introduction

Fruition Product Labels is an Adobe Commerce **App Builder** extension that lets merchants define product and category **labels and badges** from the Admin and serve them to the storefront through a **labels-by-SKU** API.

The solution combines an embedded **React** experience in Commerce Admin with **Adobe I/O Runtime** actions. Label rules and styling are stored in App Builder Files as a single configuration, so merchandising data stays with the app rather than scattered across theme code. In summary: it gives you **consistent, rule-based merchandising labels** without hard-coding badges in themes or redeploying front-end assets for every campaign tweak.

Who Should Use This App

This app is designed for:

- **Merchandising and e-commerce managers** who run promotions, sales, and seasonal messaging on specific products and need to turn labels on and off without developer involvement for each change.
- **Adobe Commerce administrators** who own catalog presentation and want Admin-native tooling to manage badges alongside the rest of commerce configuration.
- **Developers and integrators** building headless, PaaS / SaaS, or custom storefronts who need a stable contract for “give me labels for these SKUs” (product vs. category context) so the storefront can render badges from server responses.
- **Multi-store and B2B merchants** who must scope labels by store view and customer group. The app supports “all stores” and “all groups” in the UI; the API reflects that intent with resolved store and customer group identifiers so integrators can filter or display correctly per context.

Core Capabilities

- ### Label types
  Merchants can use text-only labels, predefined shapes (vector artwork generated server-side for the API), or uploaded images (stored and returned as data suitable for direct use in the browser).

- ### Placement and presentation
  Product page and category page each have their own settings: a position grid (nine cells), size, label text, text color and size, shape color where applicable, plus redirect URL, alt text, and related fields so accessibility and click-through can be controlled per context.

- ### Targeting and lifecycle
  Labels apply to a list of product SKUs. Each rule can include an optional date range (start and end), an active status, and a priority value. When hide if higher priority is enabled, a label can be suppressed if a higher-precedence (lower numeric priority) label is already shown for the same SKU, reducing visual clutter when many rules overlap.

- ### Scope
  Rules can be limited to store views and customer groups, or left broad when “all” is chosen. The storefront API returns resolved store view IDs and customer group IDs (including the “all” case), so consumers know which contexts the label is meant for.

- ### Configurable and grouped products
  A use for parent option supports showing label logic at the parent level where that matches how the catalog is merchandised.

- ### Storefront integration
  The primary integration is a POST request that accepts a list of SKUs and a mode for product vs. category listing context. The response supplies what renderers typically need—among other fields: image (when applicable), position, inline style hints, display text, redirect, tooltip, priority, scheduling fields, and scope metadata—so themes or headless clients can paint badges consistently.

How the App Fits into Your Commerce Stack

- Built natively on **Adobe App Builder**
- Designed to integrate seamlessly with **Adobe Commerce SaaS**
- API-first architecture supports:
  - Headless storefronts
  - AEM boilerplate based frontends
  - Custom UI implementations

Benefits for Your Business

- **Faster time-to-market** for promotions and badges, because many changes are configuration-only instead of full theme or pipeline deploys for every message update.
- **Consistent branding** through reusable shapes and governed upload assets, rather than one-off graphics embedded in different places.
- **Less routine work for developers**: business users can own day-to-day label updates while engineering focuses on **consuming one API** and presentation patterns.
- **Better relevance** through store view, customer group, and date scheduling—reducing wrong-audience or expired messaging on the storefront.
- **Cleaner customer experience** when many rules exist, thanks to **priority** and **hide-if-higher-priority** behavior that helps prevent overlapping or noisy badge stacks on the same product.

Prerequisites

Admin UI SDK – Installation & Configuration

Installation:

- **PaaS**
  - Please follow the instructions Install or update Adobe Commerce Admin UI SDK
  - Admin UI SDK version 3.0 or higher
- **SaaS**
  - Included already

IMS Module for Authentication

For PaaS: Please follow the instructions Adobe Identity Management Service (IMS) for Adobe Commerce

For SaaS: SaaS instances already include IMS configuration

Note: the request schemas, response schemas and mesh.json needs to be created in the api mesh app builder project before deploying the app to the workspace mentioned in below section of document.

Project Setup

Installing from App Page

This step walks you through installing the app for **Adobe Commerce** via the Adobe Exchange marketplace.

1. Go to Adobe Exchange, select "Experience Cloud" and search for **Product Labels Manager**.

2. Click button **Buy** or **Install** and follow the prompts. The application will be automatically deployed to your Adobe App Builder environment.

Please refer to the page https://developer.adobe.com/developer-distribution/experience-cloud/docs/guides/discoverAndManage/app-builder-discover

Setting up the required configurations

- Adobe Commerce Base URL

  Note: When configuring the COMMERCE_BASE_URL environment variable, the format differs between PaaS and SaaS:

  For PaaS (On-Premise/Cloud):

  - Must include your base site URL + /rest/ suffix
  - Example: https://[environment-name].us-4.magentosite.cloud/rest/

  For SaaS:

  - Must be the REST API endpoint provided by Adobe Commerce
  - Example: https://na1-sandbox.api.commerce.adobe.com/[tenant-id]/

  Make sure to use your actual environment name or tenant ID in the URL **and the URL should end in a slash(/)**. The examples above use placeholder values.

Please provide below configurations

- OAUTH_CLIENT_ID
- OAUTH_CLIENT_SECRETS
- OAUTH_TECHNICAL_ACCOUNT_ID
- OAUTH_TECHNICAL_ACCOUNT_EMAIL
- OAUTH_IMS_ORG_ID
- COMMERCE_CONSUMER_KEY (PaaS, OAuth, if IMS Auth isn't being used)
- COMMERCE_CONSUMER_SECRET (PaaS, OAuth, if IMS Auth isn't being used)
- COMMERCE_ACCESS_TOKEN (PaaS, OAuth, if IMS Auth isn't being used)
- COMMERCE_ACCESS_TOKEN_SECRET (PaaS, OAuth, if IMS Auth isn't being used)

Note:

For PaaS:

Case1: With IMS Auth - Given that the IMS module will be enabled for use with Admin UI SDK, the OAuth credentials could also be used to authenticate with PaaS. This process requires a Commerce instance with Adobe Identity Management Service (IMS) for Adobe Commerce configured.

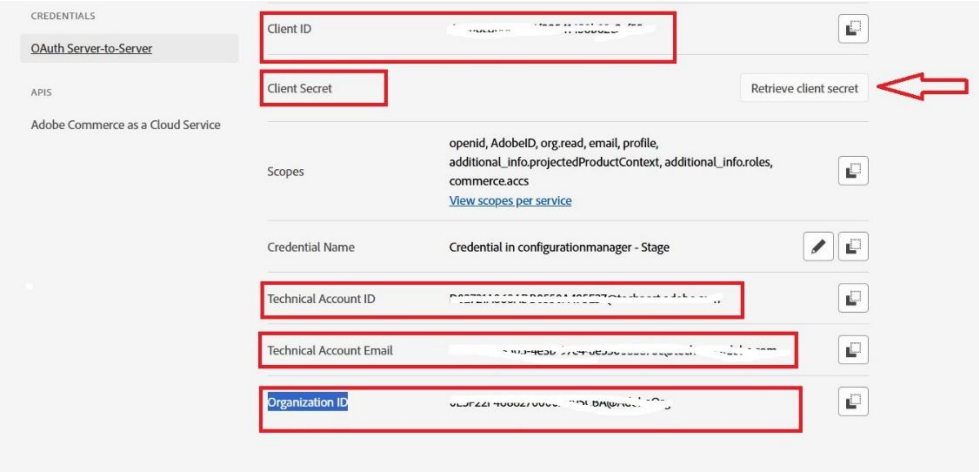
Case2: If IMS Auth isn't being used, but OAuth is used – The values (COMMERCE_CONSUMER_KEY, COMMERCE_CONSUMER_SECRET, COMMERCE_ACCESS_TOKEN, COMMERCE_ACCESS_TOKEN_SECRET) can be retrieved from a

Commerce integration, please refer to
https://experienceleague.adobe.com/en/docs/commerce-admin/systems/integrations

For SaaS: SaaS instances already include IMS configuration

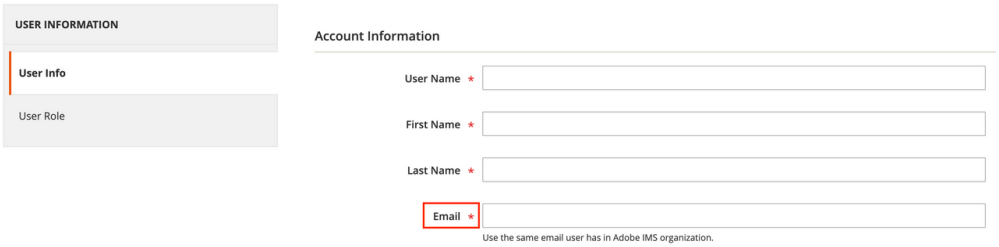
Use the following steps to create OAuth credentials for App Builder authentication:

- Go to Adobe Developer Console
  - https://developer.adobe.com/console
  - Log in with your Adobe ID (same org where your project is created).
- Access your IMS credentials through the Adobe Developer Console. Select any project and workspace within the IMS organization. Then click OAuth Server-to-Server in the side-navigation menu.
- NOTE: These credentials are automatically populated in Configure OAuth Server-to-Server Credential.
  - OAUTH_CLIENT_ID=<client id> (Found in Developer Console → your Project → Credentials → **Client ID**.)
  - OAUTH_CLIENT_SECRETS=<client secrets> (Found in Developer Console → your Project → Credentials → **Client Secrets**.)
  - OAUTH_TECHNICAL_ACCOUNT_ID=<technical account id> (Found in Developer Console → your Project → Credentials → **Technical Account ID**.)
  - OAUTH_TECHNICAL_ACCOUNT_EMAIL=<technical account email> (Found in Developer Console → your Project → Credentials → **Technical Account Email**.)
  - OAUTH_SCOPES=<scopes>
  - OAUTH_IMS_ORG_ID=<ims org> (Found in Developer Console → your Project → Credentials → **Organization ID**.)

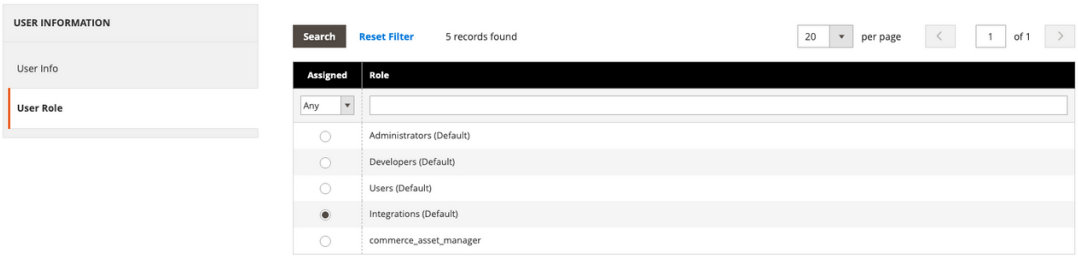


- Provide the technical account with access to the Commerce instance:
  - **SaaS** only The technical account is automatically created and associated with the Commerce instance once the first request is made using the OAuth credentials.
  - **PaaS** only Add a technical account with server-to-server credentials to the Commerce Admin with the appropriate permissions using the Admin User Creation Guide.

When associating the user with the account, find your Technical Account email as a part of generated IMS credentials with following pattern: ***<technical-account>@techacct.adobe.com*** and use that value in the Email field during user creation:



On the User Role tab, select the role that provides all necessary permissions for API integrations.

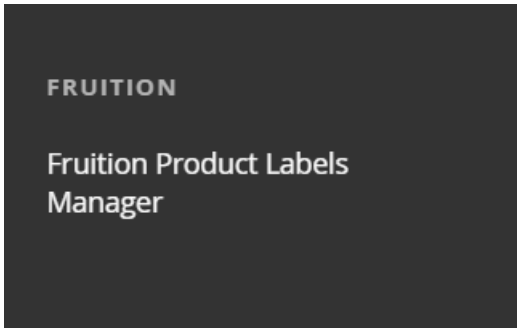


Locate Configurations in Admin Panel

Configuration:

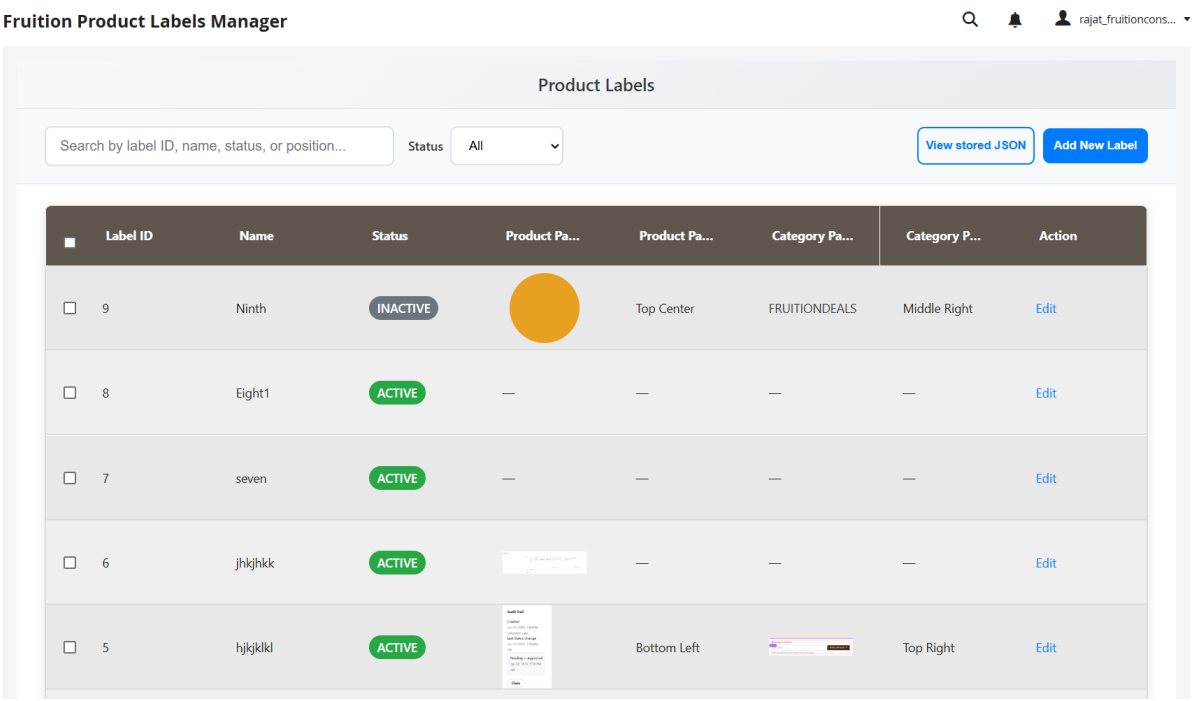
Please follow the instructions Admin configuration and testing

Once the App is installed and configured via Admin SDK UI, you will be able to see the Menu with name “Fruition”, when you click on it, a sub menu will appear “Fruition Product Labels Manager”.



Landing on Product Labels Manager

When you click on **Fruition Product Labels Manager**, the **Configuration Manager screen** opens. From here, you can **add, edit, delete, or view** existing labels.



Add a Label

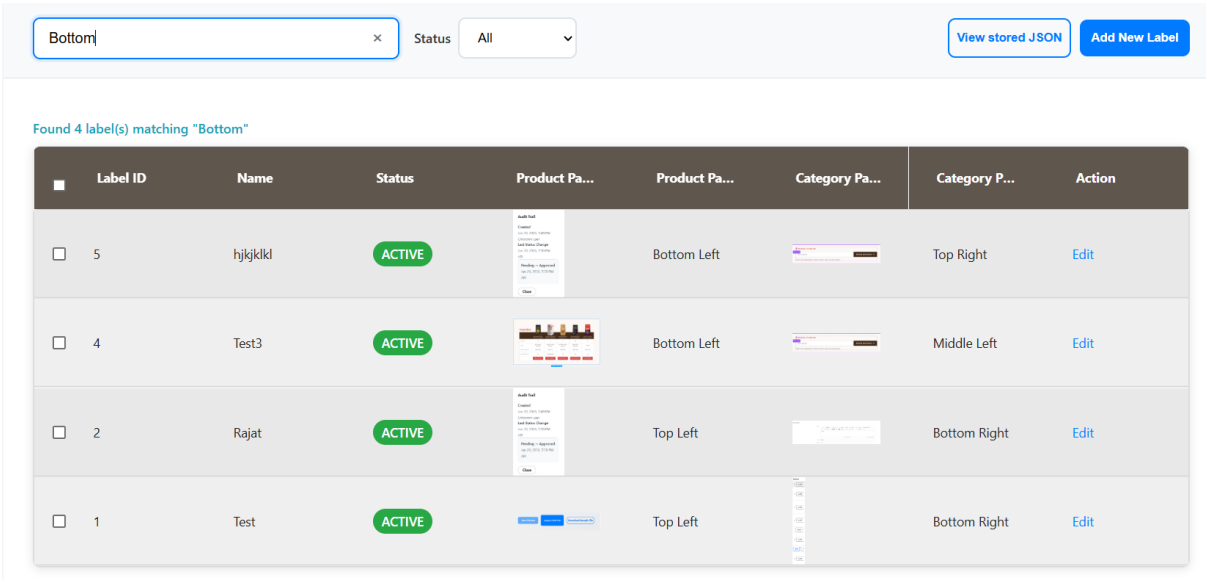
To add a new configuration:

- Enter the **Name as mandatory fields.**
- Specify Show From, Show To, Priority and Status as optional fields.
- Select the appropriate **Store / Customer Group** where this configuration should apply.
- Product Page Configuration – Select Label Type, Label Position, Text, Color, Size, URL etc
- Category Page Configuration – Select Label Type, Label Position, Text, Color, Size, URL etc
- Product Conditions - Use flexible conditions to specify which products the label should or shouldn't be displayed. All Product Attributes will be listed to pick and choose along with some specific conditions like Stock Status, Qty, Is New etc.

Search and Listing of Labels

User can search existing configurations using filters such as **Label Name, Status, Position**.

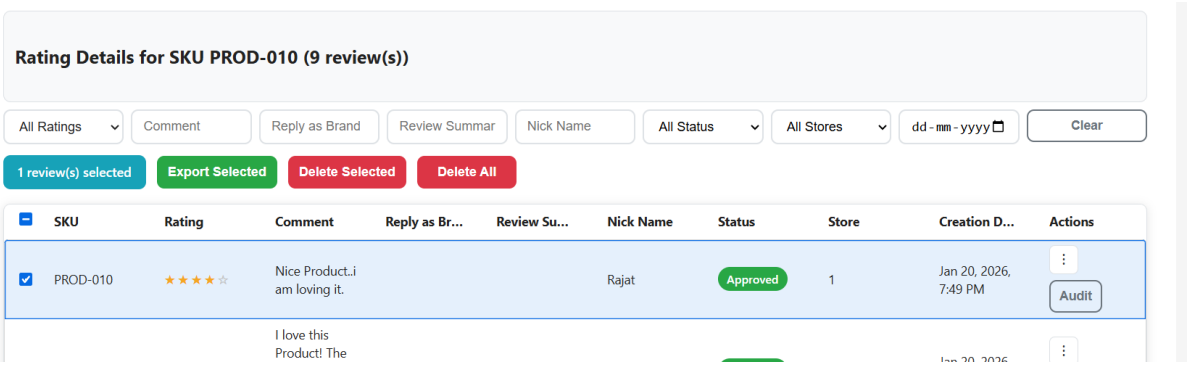
- If no filters are applied, the system will display **all available labels**.



Delete Label

To delete a Label:

1. Select the checkbox next to the labels you want to remove.
2. You can select multiple label(s) if needed.
3. Click the **Delete Selected** button.
4. Confirm the deletion when prompted.

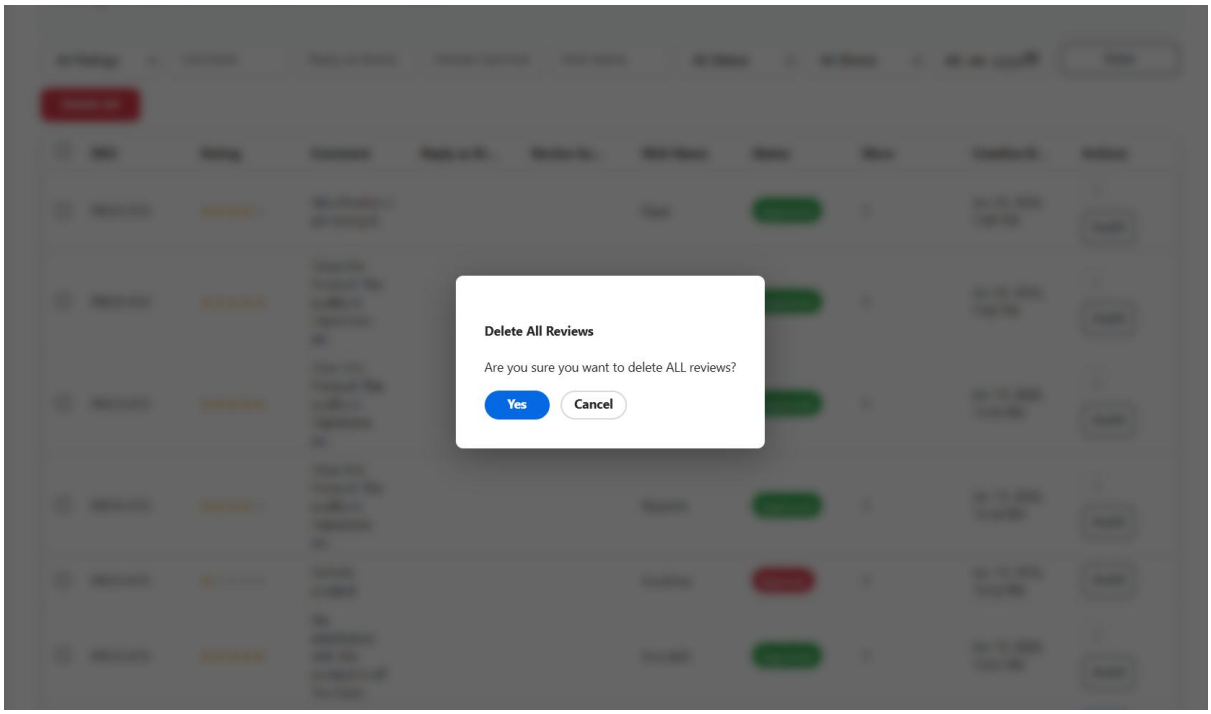


Delete All Labels

To remove all existing labels in one step:

1. Click the **Delete All** button.
2. Confirm the deletion when prompted.

⚠️ **Note:** This action is irreversible and will permanently delete all labels from the system.



API Endpoints

Base URL

https://<ACTION_URL>.adobeio-static.net/api/v1/web/productlabel-manager/

<ACTION_URL> should be replaced with the actual deployed URL.

Get Product Labels

This endpoint gets labels data from the Product Labels Manager by passing array of skus or specific sku. It is a simple HTTP POST request that returns the data posted to system.

HEADER

x-gw-ims-org-id: <OrganizationID>

Authorization: Bearer <Auth Token>

Request

Endpoint: https://<ACTION_URL>.adobeio-static.net/api/v1/web/ productlabel-manager/labelsBySku

Method: POST

Response

- **Status Code**: 200 OK
- **Content-Type**: application/json

Request Body

The request contains the following structure:

- **JSON**

```
{
  "skus": ["ADB150", "Pant", "Shorts", "Test"],
  "mode": "CATEGORY"
}
```

- **Required Fields**
  - `skus` (array): Product SKUs
  - `mode` (string): value should be CATEGORY or PRODUCT

Response Body

The response contains the following structure:

- **JSON**

```
{
  "data": {
    "fruitionLabelProvider": [
      {
        "items": [
          {
            "label_id": 4,
            "name": "Test3",
            "status": "ACTIVE",
            "mode": "CATEGORY",
            "label_type": "upload_image",
            "image": "...",
            "alt_tag": "",
            "position": "middle-left",
            "txt": "",
            "text_color": null,
            "text_size": null,
            "shape_key": null,
            "shape_color": null,
            "label_size": null,
            "size": "",
```

```
            "style": "",
            "priority": 0,
            "hide_if_higher_priority": false,
            "is_visible": true,
            "product_id": 3,
            "sku": "ADB150",
            "conditions_count": 1,
            "customer_group_ids": "0,1,2,3",
            "store_views": [1, 0, 2, 3],
            "show_from": null,
            "show_to": null,
            "redirect_url": "",
            "tooltip": null,
            "use_for_parent": false
          }
        ]
      }
    ]
  }
}
```


Notes

- The fruitionLabelProvider array groups labels **by product**. Each object in the array corresponds to one product, and its items array contains all labels assigned to that product.
- An **empty items array** ("items": []) indicates the product exists in the query scope but has no labels assigned.
- The image field contains a full **Base64-encoded data URI** and can be large in size. Consider lazy-loading or caching on the frontend.
- customer_group_ids "0,1,2,3" corresponds to: 0 = Not Logged In, 1 = General, 2 = Wholesale, 3 = Retailer (default Adobe Commerce groups).
- show_from / show_to support label scheduling. When both are null, the label is permanently active (subject to status).

Field	Type	Description
label_id	Integer	Unique identifier for the label
Name	String	Display name of the label
Status	String	Active state of the label. Values: ACTIVE, INACTIVE

Mode	String	Display context. Values: CATEGORY, PRODUCT
label_type	String	Type of label rendering. Values: upload_image, text_only, null
Image	String	Base64-encoded image data URI. null if no image
alt_tag	String	Alt text for the label image
Position	String	Position of label on product image. Values: top-left, top-right, middle-left, middle-right, bottom-left, bottom-right or empty
Txt	String	Text content displayed on the label (used when label_type is text_only)
text_color	String	Hex color code for label text
text_size	Integer	Font size for label text in pixels
shape_key	String	Key identifier for predefined label shape
shape_color	String	Hex color code for label background shape
label_size	Integer	Overall size of the label in pixels
Size	String	Custom size override
Style	String	Additional CSS style overrides
Priority	Integer	Display priority when multiple labels apply. Higher value = higher priority
hide_if_higher_priority	Boolean	If true, hides this label when a higher-priority label exists on the same product
is_visible	Boolean	Whether the label is currently visible on the storefront
product_id	Integer	ID of the product this label is assigned to
Sku	String	SKU of the product this label is assigned to
conditions_count	Integer	Number of conditions configured for label assignment rules
customer_group_ids	String	Comma-separated list of customer group IDs the label is visible to
store_views	Array of Integer	List of store view IDs where the label is displayed
show_from	String	Schedule start date (ISO 8601). null if no schedule
show_to	String	Schedule end date (ISO 8601). null if no schedule
redirect_url	String	URL to redirect when label is clicked
Tooltip	String	Tooltip text shown on hover
use_for_parent	Boolean	If true, label is also applied to the parent configurable product

Instructions for Integrating an EDS / Headless Storefront with app

Below are steps to integrate this app with an EDS Storefront or Headless App:

1. **Setting up EDS storefront –** Please follow the steps mentioned in the link https://experienceleague.adobe.com/developer/commerce/storefront/get-started/create-storefront/
2. **API Exposure from the App** – The Product Labels Manager exposes product labels data via APIs. – This is already there in App Builder App. Refer to section API Endpoints in this document.
3. **Consumption in the EDS / Headless Storefront** – An EDS / Headless storefront consumes these APIs and could render the product labels.
    a. Consume the API
    b. Call API in the block like product-details.js
    c. Render the product Labels at desired place

Note:

- The steps under "Consumption in the EDS / Headless Storefront" are just one example of how to use the app's configurations.
- the request schemas, response schemas and mesh.json needs to be created in the api mesh app builder project before deploying the app to the workspace

Configuring a Mesh

The labelsbySKU endpoint can be added as a source to a mesh, the mesh can securely store the auth credentials needed for requests to the endpoint. Please refer to the link for more information Command Reference

Sample mesh configuration snippet for the endpoint:

```
{
 "meshConfig": {
  "sources": [
   {
    "name": "productLabelManager",
    "handler": {
     "JsonSchema": {
      "baseUrl": "<ACTION_URL>/api/v1/web/productlabel-manager/ ",
      "operations": [
```

```
        {
          "type": "Query",
          "field": "labelsBySku",
          "path": "/labelsBySku",
          "method": "POST",
          "responseSchema": "./responseSchema/labelsBySku.json"
        }

      ],
      "operationHeaders": {
        "Authorization": "{context.headers['authorization']}",
        "x-gw-ims-org-id": "{context.headers['x-gw-ims-org-id']}",
        "Content-Type": "application/json"
      }
    }
  }
 ]
}
}
```

< ACTION_URL> - Replace with your URL https://.adobeio-static.net/api/v1/web/productlabel-manager

requestSchema/labelsBySku.json

```
{
 "$schema": "http://json-schema.org/draft-07/schema#",
 "title": "Labels by SKU success response",
 "description": "200 JSON body from productlabel-manager/labelsBySku (data.fruitionLabelProvider from buildFruitionLabelProvider).",
 "type": "object",
 "required": ["data"],
 "properties": {
```

```
  "data": {
    "type": "object",
    "required": ["fruitionLabelProvider"],
    "properties": {
      "fruitionLabelProvider": {
        "type": "array",
        "description": "One entry per requested SKU, in the same order as the request skus array.",
        "items": {
          "type": "object",
          "required": ["items"],
          "properties": {
            "items": {
              "type": "array",
              "items": { "$ref": "#/definitions/labelItem" }
            }
          },
          "additionalProperties": false
        }
      }
    },
    "additionalProperties": false
  }
 },
 "additionalProperties": false,
 "definitions": {
  "labelItem": {
    "type": "object",
    "description": "Single label row for storefront rendering (see actions/lib/label-provider.js
buildProviderItem).",
    "properties": {
      "alt_tag": { "type": "string" },
```

"customer_group_ids": {

"type": "string",

"description": "Comma-separated Commerce customer group ids; all groups when config scoped to all."

},

"image": {

"type": ["string", "null"],

"description": "Data URL for upload_image or select_shape; null for text_only."

},

"is_visible": { "type": "boolean" },

"label_id": { "type": ["integer", "string"] },

"name": { "type": "string" },

"position": {

"type": "string",

"description": "Kebab-cased grid position e.g. top-left, middle-center."

},

"product_id": { "type": ["integer", "null"] },

"redirect_url": { "type": "string" },

"size": {

"type": "string",

"description": "Label size without px suffix in API output."

},

"style": {

"type": "string",

"description": "Inline CSS fragment (font-size, color, --label-shape-color)."

},

"txt": { "type": "string" },

"sku": { "type": "string" },

"mode": { "type": "string", "enum": ["PRODUCT", "CATEGORY"] },

"label_type": {

"anyOf": [

```
        { "type": "null" },
        {
          "type": "string",
          "enum": ["text_only", "select_shape", "upload_image"]
        }
       ]
      },
      "shape_key": { "type": ["string", "null"] },
      "shape_color": { "type": ["string", "null"] },
      "text_color": { "type": ["string", "null"] },
      "text_size": { "type": ["string", "null"] },
      "label_size": { "type": ["string", "null"] },
      "tooltip": { "type": ["string", "null"] },
      "store_views": {
        "type": "array",
        "description": "Resolved Commerce store view ids; all store views when config scoped to all.",
        "items": { "type": "integer" }
      },
      "priority": { "type": "number" },
      "show_from": { "type": ["string", "null"], "format": "date" },
      "show_to": { "type": ["string", "null"], "format": "date" },
      "hide_if_higher_priority": { "type": "boolean" },
      "use_for_parent": { "type": "boolean" },
      "status": { "type": ["string", "null"] },
      "conditions_count": { "type": "integer" }
    },
    "additionalProperties": true
   }
  }
}
```

Examples of Creating Secret File

Create a YAML file, such as secrets.yaml, to define your secrets. The file name must end with the yaml or yml file extension. Each line in the files defines a different secret.

Example:

authorization: ${authorization_token}

x-gw-ims-org-id: ${x-gw-ims-org-id}

Create or update your mesh secrets

When you create or update a mesh that you want to include secrets in, add the --secrets flag followed by the path to your secrets file. If you do not provide the secrets file when updating a mesh that has secrets, the secrets` values are overwritten by their literal references.

Create:

aio api-mesh create mesh.json --secrets secrets.yaml

Update:

aio api-mesh update mesh.json --secrets secrets.yaml

For more information about mesh secret, please visit Secrets management

Support

In case of any questions, please contact support@fruitionconsultancies.com