

Club Custom Integration Guide

Prepared for Numan. A working example implementation in Node.js is available at bitbucket.org/brandbassadoras/club-custom-integration-example, and the full endpoint specifications are published as Swagger at custom-integration.club.co/api.

Welcome to the Club Custom Integration documentation! The Club platform offers a suite of features to activate your community and drive revenue for your brand. To leverage these features, integrating your webshop is essential. Key benefits include:

- Enabling your community to create and share discount codes that drive sales to your webshop.
- Offering freebies to your community, encouraging them to share about your products on social media.
- Tracking orders generated by your community in a centralized dashboard, showcasing the value they bring.
- Recruiting customers to join your community and become brand advocates.

From a technical perspective, your webshop backend must expose secure endpoints for operations like creating discount codes and fetching order details.

This guide provides an overview of the endpoints you need to implement. For detailed specifications, refer to the Swagger documentation: <https://custom-integration.club.co/api>

Connecting Your Webshop to Club

To integrate your webshop with Club using API Key authentication, provide the following:

- **API URL:** The base URL of your webshop's API.
- **Authentication URL:** This can be the same as your API URL or a dedicated endpoint if using a third-party authentication service.
- **API Key:** A secure key for authenticating requests.

Integration Process

When you integrate your webshop with Club, we will:

1. Authenticate with your webshop using POST `/api-key/token` and retrieve an access token.
2. Retrieve store information and collections using GET `/store/*` endpoints.
3. Subscribe to a webhook to receive order updates via POST `/webhook/subscribe_webhook`.

Once connected, you can begin using features that require integration.

Authentication

API Key Authentication

Endpoint: POST /api-key/token **Purpose:** Authenticate using an API key to retrieve an access token.

Store Endpoints

The store endpoints provide information about the e-commerce integration, including the default currency, countries the platform ships to, available collections, and supported features.

Get Store Info

Endpoint: GET /store/info **Purpose:** Retrieves general information about the store. The store info endpoint returns a list of supported features, indicating the capabilities of the integration:

- write_discount_code: Required to enable integration. Allows creation of discount codes.
- read_discount_code: Required to enable integration. Allows retrieval of discount code information.
- custom_discount_codes: Required to enable integration. Allows user to specify the discount codes on creation.
- read_orders: Required to enable integration. Enables access to order information.
- webhook: Required to enable integration. Enables Club to receive updates on orders.
- write_orders: Not required but reserved for future order product features on custom integrations.
- read_customer: Not required. Used to activate customers from Club.

Get Collections

Endpoint: GET /store/collections **Purpose:** Retrieves a list of collections with optional pagination. Collections are used by the discount code feature to restrict the validity of a discount code to specific products.

Discount Code Management

The discount code endpoints allow you to create, retrieve, and delete discount codes. A discount code is typically a string that provides a percentage or fixed amount off on eligible products or collections. We need access to these endpoints to enable discount code and voucher creation for Club. Discount codes are used in features like:

1. Sales Challenges where an ambassador can earn a commission.
2. Gift Card withdrawal, where an ambassador earns and withdraws store vouchers.

3. Mission action integrations such as Order Products using vouchers.

Create a Discount Code

Endpoint: POST /discount-code **Purpose:** Creates a new discount code.

Get a Discount Code by ID

Endpoint: GET /discount-code/{id} **Purpose:** Retrieves details of a discount code by its ID.

Delete a Discount Code

Endpoint: DELETE /discount-code/{id} **Purpose:** Deletes a discount code by its ID.

Order Management

The orders endpoints allow you to read customer orders. Order information is used to check which orders were driven by a Club feature and to calculate rewards and commissions for ambassadors.

Get Orders

Endpoint: GET /order/orders **Purpose:** Retrieves a list of orders with optional pagination and filtering by date. Orders must be sorted by updated_at in ascending order. Make sure you only return the orders that are in a status you want to track.

Get Order by ID

Endpoint: GET /order/{id} **Purpose:** Fetches detailed information about a specific order by its ID.

Customer Management

The customer endpoints allow you to manage customer information. This data is used to enable the Recruitment features within the Club platform.

Get Customers

Endpoint: GET /customer/customers **Purpose:** Retrieves a list of customers with optional pagination. This feature supports:

- Automatically identifying customers who follow you on Club and adding them to your Customer Club.
- Sending recruitment emails to recent customers, inviting them to join your community.

If you wish to use the recruitment feature, implement this endpoint and include `read_customer` in the `supported_features` field of the GET `/store/info` response.

Webhook Management

Webhooks allow the Club platform to receive real-time notifications about order updates. By using webhooks, the load on the custom integration is reduced as polling the order endpoint is unnecessary.

Subscribe Webhook

Endpoint: POST `/webhook/subscribe_webhook` **Purpose:** Registers a webhook by providing a URL and a secret key. If a webhook with the same URL already exists, it updates the secret key.

Unsubscribe Webhook

Endpoint: POST `/webhook/unsubscribe_webhook` **Purpose:** Removes a webhook subscription using its ID.

Test Webhook

Endpoint: POST `/webhook/try_webhook` **Purpose:** Tests webhook functionality by sending a sample order to all subscribed webhook URLs.

Webhook Notifications

When a webhook is triggered, it sends a `WebhookBody` to the subscribed URL. The body includes:

- `webhook_id`: The ID of the webhook subscription.
- `data`: The updated order (`OrderDto`).

For security, each webhook request includes an HMAC signature in the `x-club-hmac-sha256` header. This signature is created using the secret key provided during subscription and can be used to verify the authenticity of the webhook payload.

Example of generating the signature:

```
const body: WebhookBody = {
  webhook_id,
  data: order,
};

const hmac = crypto
  .createHmac('SHA256', webhook.secret_key)
  .update(JSON.stringify(body), 'utf8')
  .digest('base64');
```

Data Standards

To ensure consistency and interoperability, this API adheres to the following international standards for representing various types of data:

Date and Time

All date and time values are represented using the ISO 8601:2004 standard. This format provides a standardized way to represent dates and times, making it easier to parse and understand across different systems.

Example: 2023-05-15T14:30:00.394 Z (May 15, 2023, at 2:30 PM UTC)

Currencies

Currency codes follow the ISO 4217 standard. This three-letter code system is widely recognized and used to represent currencies in financial transactions and data exchange.

Example: USD (United States Dollar), EUR (Euro), JPY (Japanese Yen)

Countries

Country codes adhere to the ISO 3166-1 alpha-2 standard. This two-letter code system provides a concise and universally recognized way to represent countries.

Example: US (United States), GB (United Kingdom), JP (Japan)

By following these standards, we ensure that data exchanged through this API is consistent, easily interpretable, and compatible with a wide range of systems and applications.

Errors

The API uses HTTP status codes to indicate the success or failure of requests. The following status codes are used:

- 200 OK: The request was successful.
- 201 Created: The request was successful and a resource was created.
- 400 Bad Request: The request was invalid.
- 401 Unauthorized: The request was not authenticated.
- 404 Not Found: The requested resource was not found.
- 500 Internal Server Error: The server encountered an error.

For each status code we expect to return a JSON body with the following fields:

- `code`: A string that uniquely identifies the error. Check the swagger documentation for the error codes inside each endpoint.
- `message`: A human-readable description of the error.

For All Apis we support the following generic error codes:

- AUTH_ERROR: The request was not authenticated.
- BAD_REQUEST: The request was invalid.
- INTERNAL_SERVER_ERROR: The server encountered an error.

For detailed API specifications, visit the Swagger documentation: [Swagger API Documentation](#).