iZettle Partner Integrations

Solution Documentation

PayPal Global Professional Services

Version 0.9.1, 15 April 2021

Table of Contents

Glossary . . . 2
Document Version History . . . 3
About this Document . . . 5
1. Goals of this Integration . . . 6
2. iZettle Products . . . 7
3. User Stories . . . 8
3.1. E-commerce Platform iZettle Extension Discovery and Installation . . . 8
3.2. Connect iZettle Account . . . 8
3.3. Product Library Synchronization . . . 9
3.3.1. Scope . . . 9
3.3.2. Synchronization Options . . . 10
3.3.3. Single Source of Truth . . . 11
3.3.4. Initial Import . . . 11
3.3.5. Updates and Synchronization . . . 12
3.3.6. Handling of Synchronization Conflicts . . . 13
3.4. Inventory Synchronization . . . 13
3.4.1. Configurations . . . 14
3.4.2. Two-Way Synchronization . . . 14
3.4.3. Initial Import . . . 14
3.4.4. Updates and Synchronization . . . 14
3.4.5. Handling of Synchronization Conflicts . . . 14
3.5. Logging and Debugging . . . 15
3.6. Retrieve Transaction Information . . . 15
3.7. Non-Functional Requirements . . . 15
3.7.1. Performance and Behavior . . . 15
3.7.2. Tracking . . . 15
3.7.3. Partner Affiliation . . . 16
3.8. Other Use Cases . . . 16
4. Technical Specifications . . . 17
4.1. General API Features . . . 17
4.1.1. Partner Affiliation Implementation . . . 17
4.2. iZettle OAuth endpoint . . . 17
4.3. Onboarding, Authentication & Authorization . . . 17
4.3.1. OAuth 3rd-party flow . . . 18
4.3.2. API Keys . . . 22
4.3.3. Permission Scopes . . . 27
4.4. Account Information API . . . 27
4.4.1. Tax and VAT Settings Information . . . 27

4.4.2. Retrieve a Merchant's Account Information . . . 28
4.5. Product Library API . . . 30
4.5.1. Products Data Model . . . 31
4.5.2. Managing Products . . . 32
4.5.3. VAT and Sales Tax . . . 33
4.5.4. Discounts . . . 35
4.6. Image API . . . 36
4.6.1. Importing Images to iZettle . . . 36
4.7. Inventory API . . . 39
4.7.1. Inventory Lifecycle and Locations . . . 39
4.7.2. Managing Inventory Tracking . . . 40
4.7.3. Updating Inventory Balances . . . 44
4.8. Purchase API . . . 45
4.9. Pusher API (WebHooks) . . . 46
4.10. Swagger files . . . 46
4.11. Postman Collection . . . 46
4.11.1. Usage . . . 47
4.11.2. Collection Updates . . . 47
5. Integration . . . 48
5.1. Prerequisites . . . 48
5.1.1. Developer Account . . . 48
5.2. Languages to be supported . . . 48
5.3. Testing Environment . . . 48
5.4. Integration Milestones . . . 48
5.5. Integration Certification . . . 48
5.5.1. Post-Live Support . . . 48
6. Frequently Asked Questions . . . 49
7. Points of Contact . . . 50

Glossary

E-commerce Platform

Software and services that allows online businesses to manage their website, marketing, sales, and operations.

E-commerce Platform Merchant

A user of E-commerce Platform selling goods or services through their online stores.

iZettle Merchant

A user selling goods or services offline through iZettle.

iZettle Reader

iZettle's hardware device for accepting card and contactless payments.

iZettle Go

iZettle's POS apps for mobile devices (iOS/Android)

iZettle APIs

iZettle's REST APIs for partners to connect with iZettle functionalities such as e.g. iZettle's product library or inventory tracking.

E-commerce Platform iZettle Extension

Software and services to that enable an E-commerce Platform Merchant to accept offline payments with iZettle through an integrated experience.

Buyer

A customer of a merchant buying goods or services from the merchant on- or offline.

POS System

Point of Sale System – hard/software system to facilitate offline sales

Partner

E-commerce Platform partnering with iZettle/PayPal to offer iZettle payments to the E-commerce Platform Merchants.

Integration Partner

iZettle/PayPal partner who will implement, operate and maintain an iZettle Partner integration for an E-commerce Platform. The *Implementation Partner* may be the *Partner* role if the implementation is facilitated by the operator of the E-commerce Platform themselves. The *Implementation Partner* may also be e.g. a System Integrator that builds an implementation for an E-commerce Platform on behalf of PayPal/iZettle, the E-commerce Platform or a third-party.

Document Version History

0.9.1 (15 April 2021)

- Some changes were left out in the release build, fixed.
- Fixed broken reference links.

0.9.0 (14 April 2021)

- Updated information how to retrieve iZettle account settings for VAT/Tax handling.
- Added information how to handle Sales Tax.
- Fixed typos; minor wording improvements.

0.8.1 (25 March 2021)

- Added FAQ entry how to delete product library entries during testing.
- Added reference to Swagger specs for Product Library API.

0.8.0 (7 October 2020)

- Added Logging and Debugging use case.
- Added non-functional requirements for Performance and Behavior.
- Added clarifications to product price specifications (see Product Prices).
- Updated description of requirements and limitations for importing images (see Importing Image from URLs).
- Added tracking parameters to login flow URL specifications (see Authorization Web Flow and API Key Creation Flow).
- Clarified two-way nature of inventory data synchronization (see Inventory Synchronization).
- Fixed typos and spelling.
- Removed DRAFT mark.

0.7.2 (8 June 2020)

- Added list of available permission scopes (see Permission Scopes).
- Added specifications to include a partner identifier in a product's metadata when creating a product (see Partner Identifier).
- Added documentation for partner affiliation capabilities. A partner using API Keys no longer needs to retrieve a user's Client ID by decoding the API Key but will instead use a partner's Client ID (see Partner Affiliation, Partner Affiliation Implementation, and updates to the API Keys section).
- Provided additional information on required tracking capabilities (see Tracking).

0.7.1 (May 2020)

- Added subsections about API Key Management and behavior of revoked API Keys
- Added screenshots of API Key creation flow
- Fixed typos, smaller wording improvements

0.7 (May 2020)

- Added API Keys flow, distinction and additional technical details for both authorization methods
- Extended Connection and Initial Synchronization sections and added best practices section
- Added Account Information API information
- Added information how to use eTags when updating product information
- Various typos fixed and wordings improved for consistency.

0.6 (10 Feb 2020)

- Additional specifications for initial product import
- Fixed typos; overall wording improvements

0.5 (5 Feb 2020)

- First generic version

0.3 (20 Dec 2019)

- Added Inventory API specifications

0.2 (17 Dec 2019)

- Added FAQ section
- Further detailed inventory tracking use cases
- Added product library synchronization conflict handling use cases
- Product Library Images should be referenced by URL instead of Image Keys. Removed references to Image Keys and updated relevant documentation accordingly.

0.1 (30 Nov 2019)

- Initial version

About this Document

This document provides technical specifications and defines the engagement between PayPal and *Integration Partner* to successfully implement the integration solutions outlined in this document. This document will document the understanding of the requirements and shall support *Integration Partner* in going through the integration process.

This document is designed to be used by *Integration Partner's* engineering team as they plan and complete their integration with PayPal and iZettle. As the integration progresses this document may be updated to provide e.g. additional specifications and clarifications, address deficiencies, reduce confusion, and address changes to the integration plan.

PayPal will certify the integration before launching/release.

General Notes

- The information contained in this document is proprietary and confidential and must not be shared and/or distributed without explicit consent of PayPal.
- This document is in draft state and still undergoing frequent changes as requirements of the integration are defined. Once finalized, this statement will be removed.
- Screenshots of experiences displayed in this document are provided for orientation only and may differ from actual implementations.
- Sample API requests are provided for illustrative purposes and must not be used without proper precautions in a live environment. PayPal and iZettle assume no liability or responsibility for effects caused by the use of the provided sample API requests.

1. Goals of this Integration

Enable merchants of the E-commerce platform to accept offline payments through iZettle with an integrated experience.

TBD: Distinction between integrations into the core E-commerce platform vs. creating a plugin/extension to be added to a E-commerce platform's installation. The current version of this document discusses a plugin/extension integration.

To be completed

The iZettle Extension will inform and guide E-commerce Platform merchants through all steps necessary to enable taking offline payments with iZettle. For use cases not available through integration with iZettle APIs or other products, the extension will provide comprehensive information and help for E-commerce Platform merchants how to proceed. An iZettle account will be required to use the E-commerce Platform iZettle Extension. The extension will provide instructions for E-commerce Platform merchants how to create an iZettle account, and how to purchase one or more iZettle readers to be able to accept offline payments.

2. iZettle Products

The iZettle Card Reader makes it possible for anyone to take card payments – any time, anywhere, while iZettle's intuitive free point-of-sale system, iZettle Go, helps merchants get paid, tracks their sales and lets them keep an eye on their inventory stock levels.

Figure 1. iZettle Reader

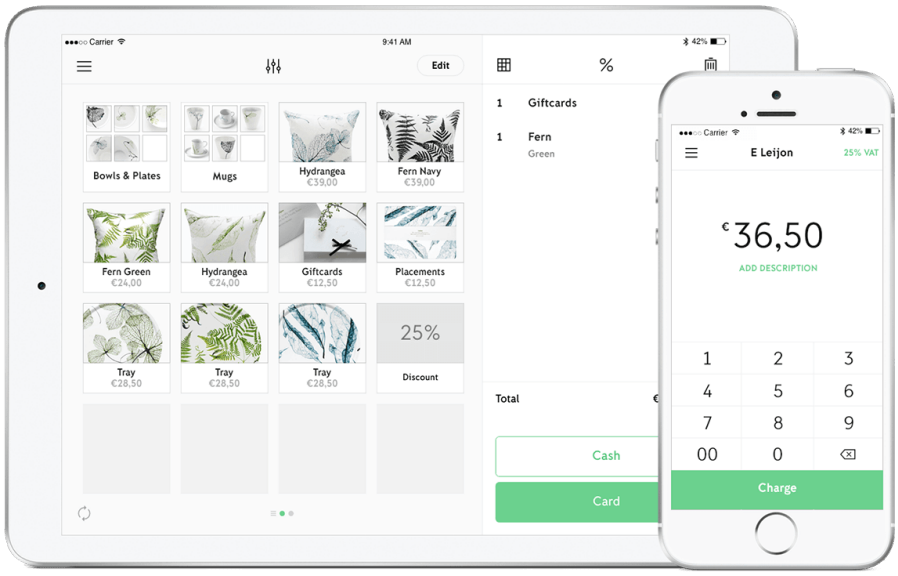


Figure 2. iZettle Go Apps

3. User Stories

3.1. E-commerce Platform iZettle Extension Discovery and Installation

> *As a merchant I want to be able to add iZettle functionalities to my E-commerce Platform setup.*

The E-commerce Platform iZettle Extension will be available for free through the distribution channels common for the E-commerce Platform, e.g. an extension store or for download. The E-commerce Platform iZettle Extension should be listed and easy to find in available extension directories. Merchants should be able to identify iZettle/PayPal as the publishers of the extension.

The E-commerce Platform iZettle Extension should not require other extensions, non-standard versions and/or settings for the E-commerce Platform installation and components thereof. No modifications to the E-commerce Platform merchant's installation should be required for installation and operation of the extension.

3.2. Connect iZettle Account

> *As a merchant I want to be able to connect my iZettle account with my E-commerce Platform account.*

After installation of the E-commerce Platform iZettle Extension, the E-commerce Platform merchant should be guided through the necessary steps to connect their iZettle account with the merchant's online store:

1. **Inform**:
    - Inform the E-commerce Platform merchant about the requirement of an iZettle account and on the necessary steps to onboard for such an account.
    - Advise the E-commerce Platform merchant how to order an iZettle Reader device if necessary.
2. **Connect**:
    - Provide a way to connect the merchant's iZettle account to their E-commerce Platform setup. The merchant will need to enter their iZettle account credentials and agree to the necessary access permissions.
3. **Initial Product Synchronization**:
    - Let the merchant start the initial product synchronization. The merchant will be requested to provide any necessary configuration and setup options as part of this step, e.g. wether the merchant intends to synchronize product prices.
4. **Manage Connection**:
    - Display the account connection status.
    - Allow the merchant to disconnect their connected iZettle account at any time.

5. **Guide**:
    - Guide the merchant though the necessary steps to enable synchronization of product libraries and inventory tracking.

3.3. Product Library Synchronization

> *As a merchant I want to be able to access and use my product library for both my online and offline sales.*

The iZettle Go POS application will display product items that have been set up in the iZettle product library. The iZettle product library can be edited within the iZettle Go POS application and through the iZettle web interface, as well as through the iZettle Connect APIs.

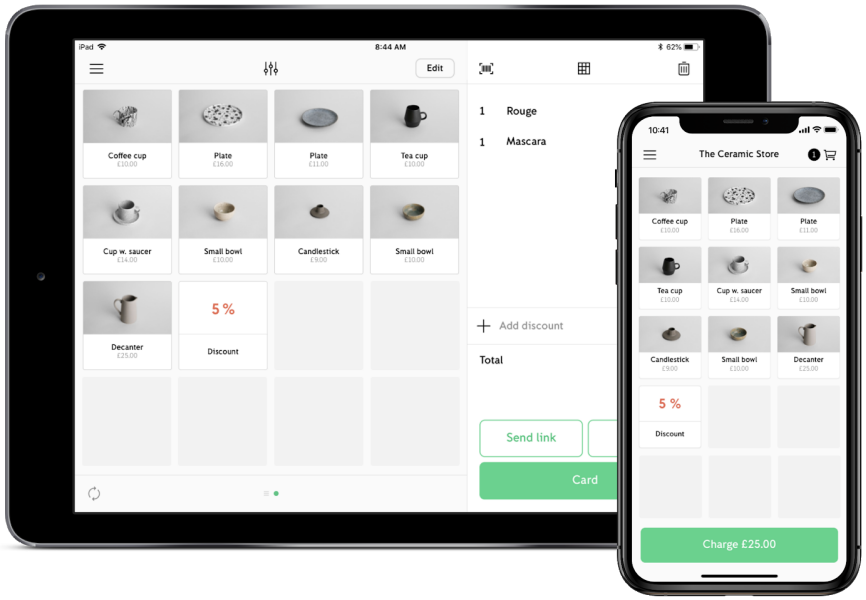


Figure 3. iZettle Product Library

The E-commerce Platform iZettle Extension should provide the necessary means for a merchant to use their product library for both online and offline sales. The E-commerce Platform iZettle Extension should implement methods to synchronize E-commerce Platform's and iZettle product libraries, and to keep them synchronized across changes and updates on both sides.

3.3.1. Scope

The following entities are to be synchronized between E-commerce Platform's and iZettle product libraries:

- Products
- Product Variants
- Product Images
- Discounts

The implementation will need to take care of any necessary mapping between the data models of both libraries.

Best Practices

Larger synchronization tasks (e.g. initial imports of big product libraries) may take some time and increase the possibility of interruptions or other failures before completion. The partner's integration should plan for such failures and implement robust retry and recovery mechanisms to avoid broken states such as an partially imported product library.

3.3.2. Synchronization Options

> *As a merchant I want to be able to specify which product information is shared between online and offline product libraries.*

Example: a merchant offers products at different prices for online vs. offline sales.

The E-commerce Platform iZettle Extension should allow the merchant to specify all relevant parameters for the product library synchronization use case, e.g.

- whether online/offline product prices should be synchronized
- which product images are to be shared for online/offline use

3.3.2.1. Price Synchronization

The merchant should be offered an option wether product data should be synchronized to the target library with or without prices. If the merchant selects to not synchronize price data, prices will have to be set manually in the target library. Future updates should not overwrite manually entered price data in this case as long as price synchronization is disabled for the product.

In addition to an initial global setting the partner may chose to implement enabling price synchronization on product level.

In the case of a **currency mismatch** (product currency does not match currency of the merchant's iZettle account) prices should not be synchronized but will have to be set manually. In this case the merchant should be notified and provided guidance how to complete product setup.

VAT and Sales Tax Check

In the case that the merchant intends to synchronize product prices, the integration needs to ensure that correct tax settings are applied when exchanging product data. As part of the setup flow, the integration must therefore retrieve the tax settings for the merchant's iZettle account settings (see Tax and VAT Settings Information) and match them to the merchant's settings in the E-Commerce Platform. In case of a mismatch, the merchant should be informed and, if possible, should be provided available options to adjust the settings.

The integration will need to check if the merchant uses net or gross prices on both E-Commerce

Platform and the merchant's iZettle account (see Price Types). In case of a mismatch (gross prices used on one system, net prices on the other), the integration should either:

- if possible, take care of the necessary conversion (this will most likely only be possible for accounts using VAT); or
- inform the merchant about the mismatch and allow for adjustments to be performed (e.g. by recommending that the merchant updates their iZettle account settings).

In case of a mismatch, the merchant must be made of the implications (e.g. margins may change) and should be explicitly requested to confirm going forward despite the mismatch.

If the merchant's iZettle account uses sales tax (see Taxation Types), the integration should check if any default sales tax rates are set up in the merchant's iZettle account (see Sales Tax Rates) before creating products in the iZettle Product Library. If this is not the case, the merchant should be informed and asked to complete their iZettle account setup before continuing. It should be made clear to the merchant that otherwise no sales tax information will be applied to the exported products and may have to be set manually by the merchant in the iZettle Backoffice.

3.3.3. Single Source of Truth

A single source of truth should be defined for the synchronization of the E-commerce Platform/iZettle product libraries. Ideally, any product item updates are mastered only within this source, and all changes occurring in this data source will be (automatically) synchronized to the other library.

Integration Partner and PayPal will define which of the product libraries (iZettle vs. E-commerce Platform) should be considered the source of truth, or whether the decision should be left as a choice to the merchant.

3.3.4. Initial Import

> *As a merchant I want to be able to use my product library for both my online and offline sales.*

The E-commerce Platform iZettle Extension should allow an initial import between E-commerce Platform's and iZettle product libraries. The direction of the import will depend on the source of truth. For importing large product libraries an appropriate asynchronous mechanism with meaningful progress information should be provided.

3.3.4.1. Import Options

The initial import should allow the merchant to define wether

- all data of the library that is not the source of truth should be deleted before importing items of the other library,
- data of the source of truth library should be added to the other library, or (optionally)
- data of the two libraries should be merged and distributed to both libraries.

The merchant should be informed and warned about any destructive and irreversible actions such as deleting existing products of a library, see .

Best Practices

Only available options should be displayed to the merchant, based on the merchant's current scenario:

- Don't offer to synchronize data from the source library if the source library is empty.
- Don't offer deleting the target library if the target library is empty.

Unavailable options may still be displayed but marked inactive, e.g. by disabling a radio button option.

If no synchronization options are available the merchant should still be informed how future products will be synchronized.

Best Practices

The merchant should be shown a summary of their chosen settings for review before starting the initial synchronization.

3.3.4.2. Duplicates

If possible the import process should recognize the presence of identical products in both libraries and allow appropriate actions, e.g. to merge information.

3.3.4.3. Reversibility

> *As a merchant I want to reverse my setup of the E-commerce Platform iZettle Extension.*

Merchants should be able to try out all functionalities of the extension without extensive cleanup operations in case they don't intend to continue using the extension. It therefore should be possible to easily reverse the initial import process where possible, e.g. to easily remove all product information that has been added during the import.

It may not be possible to restore the initial state, e.g. if one of the libraries has been deleted during import. The merchant should be informed and warned about any such actions and options.

3.3.4.4. Next Steps

After the initial import the merchant should be informed about how future changes to the product libraries will be synchronized.

3.3.5. Updates and Synchronization

> *As a merchant I want changes made to my product library to be available for both my online and offline sales.*

Updates made to either of the product libraries should be automatically (or manually) synchronized and made available in both product libraries.

3.3.6. Handling of Synchronization Conflicts

> *As a merchant I want to be supported with keeping my online/offline product libraries consistent and synchronized, and want to be able to resolve potential conflicts.*

The E-commerce Platform iZettle Extension should provide means for the merchant to

a. be informed about discrepancies that permit synchronization of E-commerce Platform's and iZettle product libraries, and to

b. resolve any pending synchronization issues gracefully.

3.4. Inventory Synchronization

> *As a merchant I want to be able to track my inventory for both online and offline sales.*

iZettle provides inventory tracking for items in the iZettle product library. Sales through iZettle will automatically be tracked and will update the inventory accordingly. Inventory balances can be managed from within the iZettle Go POS application, through the iZettle web interfaces, and through the iZettle Connect APIs. Inventory tracking can be enabled individually and independently for each item in the iZettle product library.

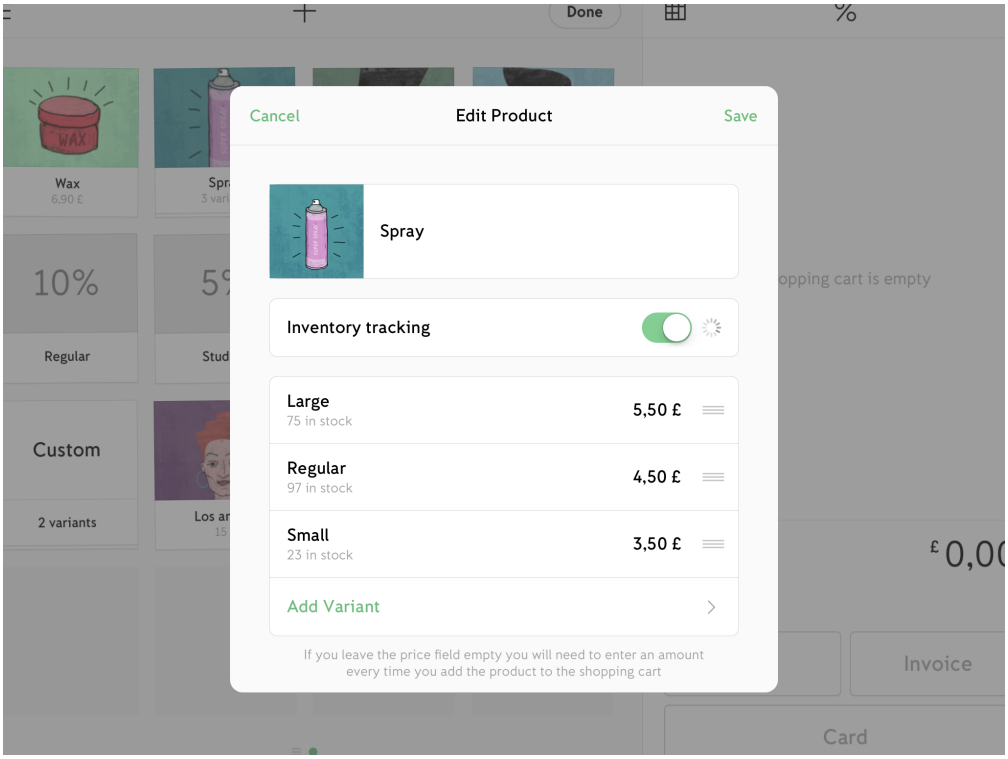


Figure 4. Enabling inventory tracking for a product

The E-commerce Platform iZettle Extension should provide the necessary means for a merchant to track their product inventory across online and offline sales. The E-commerce Platform iZettle Extension should implement methods to synchronize E-commerce Platform's and iZettle inventories, and to keep them synchronized across changes and updates on both sides.

iZettle tracks inventory on product level only. Inventory tracking for individual variants is not supported by iZettle at this time.

3.4.1. Configurations

> *As a merchant I want to be able to specify for which products to track inventory status online and offline.*

Example: a merchant offers both services (no inventory tracking) and products (with inventory tracking) offline.

The E-commerce Platform iZettle Extension should allow the merchant to specify which products should be enabled for offline inventory tracking. Enabling/disabling a product's inventory tracking in the iZettle backoffice should be properly reflected in the other platform.

3.4.2. Two-Way Synchronization

Product inventory data has to be synchronized in both directions and should always be reflected correctly with the same value on both platforms. Updating of inventory data should be immediate and not require e.g. scheduled synchronization runs.

3.4.3. Initial Import

> *As a merchant I want to be able to use my inventory tracking for both my online and offline sales.*

The E-commerce Platform iZettle Extension should allow an initial import between E-commerce Platform's and iZettle product inventories, depending on the direction of the initial product data import.

3.4.4. Updates and Synchronization

> *As a merchant I want my sales and other changes made to my inventory to be tracked in both directions across my online and offline sales.*

Updates made to either one of the inventories should be automatically (or manually) synchronized and made available to both platforms. E.g. if a product starts with a stock amount of 10 items and the merchant sells two products online and one via iZettle, the total stock amount should show a remaining stock count of 7 on both platforms.

Updates include:

- Selling of products online and offline
- Re-stocking of inventory
- Voiding of inventory

3.4.5. Handling of Synchronization Conflicts

> *As a merchant I want to be supported with keeping my online/offline inventories consistent and synchronized, and want to be able to resolve potential conflicts.*

The E-commerce Platform iZettle Extension should provide means for the merchant to

a. be informed about discrepancies that permit synchronization of E-commerce Platform's and iZettle inventory tracking, and to
b. resolve any pending synchronization issues gracefully.

3.5. Logging and Debugging

> *As an integrator I want to be able to debug a merchant's installation of the iZettle Extension.*

The integration should provide an easy way for a merchant to find and view status information and history of any actions performed by the iZettle Extension. In particular, the merchant should be able to provide information about errors occurring during past actions to the Partner to allow for triaging and debugging issues. The Partner may also decide to provide automated ways to submit or receive this information.

Best Practices

In addition to logging information from the iZettle Extension itself it may make sense to collect information additional information about the merchant's e-commerce platform, the underlying system software etc. all in one place.

3.6. Retrieve Transaction Information

> *As a merchant I want to be able to see aggregated sales/payments reports for both my in-store sales through iZettle and my online sales.*

3.7. Non-Functional Requirements

3.7.1. Performance and Behavior

Usage of the E-commerce Platform iZettle Extension must not impact the performance of the merchant's online store. Performance-intensive processes must be executed asynchronously and must not block the merchant's or a customer's experience. Long-running processes (e.g. an initial export of a large number of products) must be able to run in the background and should implement a notification mechanism that informs the merchant about the process status (completion, errors) of the process.

The merchant should be able to cancel long-running processes and, if possible, revert a partially executed process.

TBD: Performance benchmarking.

3.7.2. Tracking

The number of downloads, installations and completed/active configurations of the E-commerce Platform iZettle Extension should be tracked and made available to PayPal and iZettle teams.

In addition, the E-commerce Platform iZettle Extension should allow insight into a user's behavior and usage of the extension, e.g. the number of tracked products and ratio of synchronized vs. omitted products.

Explicit KPI definitions and tracking measures are to be defined as part of the individual integration.

3.7.3. Partner Affiliation

The E-commerce Platform iZettle Extension will implement measures that allow association of all generated data exchange and traffic with iZettle's systems to the partner.

3.8. Other Use Cases

iZettle offers and supports a number of other use cases which are not in scope of this integration, e.g.

- Checkout
- Orders
- Customer management
- Gift cards
- Managing multiple iZettle accounts

Development of the E-commerce Platform iZettle Extension should take into account that additional use cases may need to be supported in the future.

4. Technical Specifications

iZettle E-commerce integrations will use the iZettle REST APIs.

Online documentation for the iZettle APIs can be found at https://github.com/iZettle/api-documentation. There is also a list of Frequently Asked Questions.

4.1. General API Features

4.1.1. Partner Affiliation Implementation

iZettle E-commerce integration need to implement measures for iZettle to be able to associate all generated API traffic from the extension to a partner. As identifying value a Client ID owned the partner is to be used.

The association of traffic to the partner is implemented by

a. adding a header with a partner's Client ID to every API request, and

b. providing the partner's Client ID to the Retrieve Access Token from API Key request (if applicable).

4.1.1.1. Partner Affiliation Header

iZettle E-commerce integrations are required to set a dedicated HTTP header value for each request to iZettle's APIs:

```
X-iZettle-Application-Id: 6adde977-c34d-4de1-99b2-f6ed3e65431a
```

The header value in the example line is to be replaced with the partner's Client ID (UUID v1).

4.2. iZettle OAuth endpoint

See the iZettle OAuth online documentation for detailed information.

4.3. Onboarding, Authentication & Authorization

Authentication and authorization of iZettle merchants will leverage iZettle's OAuth 2.0 framework implementation.

iZettle supports multiple authentication schemes to be selected depending on the integration model:

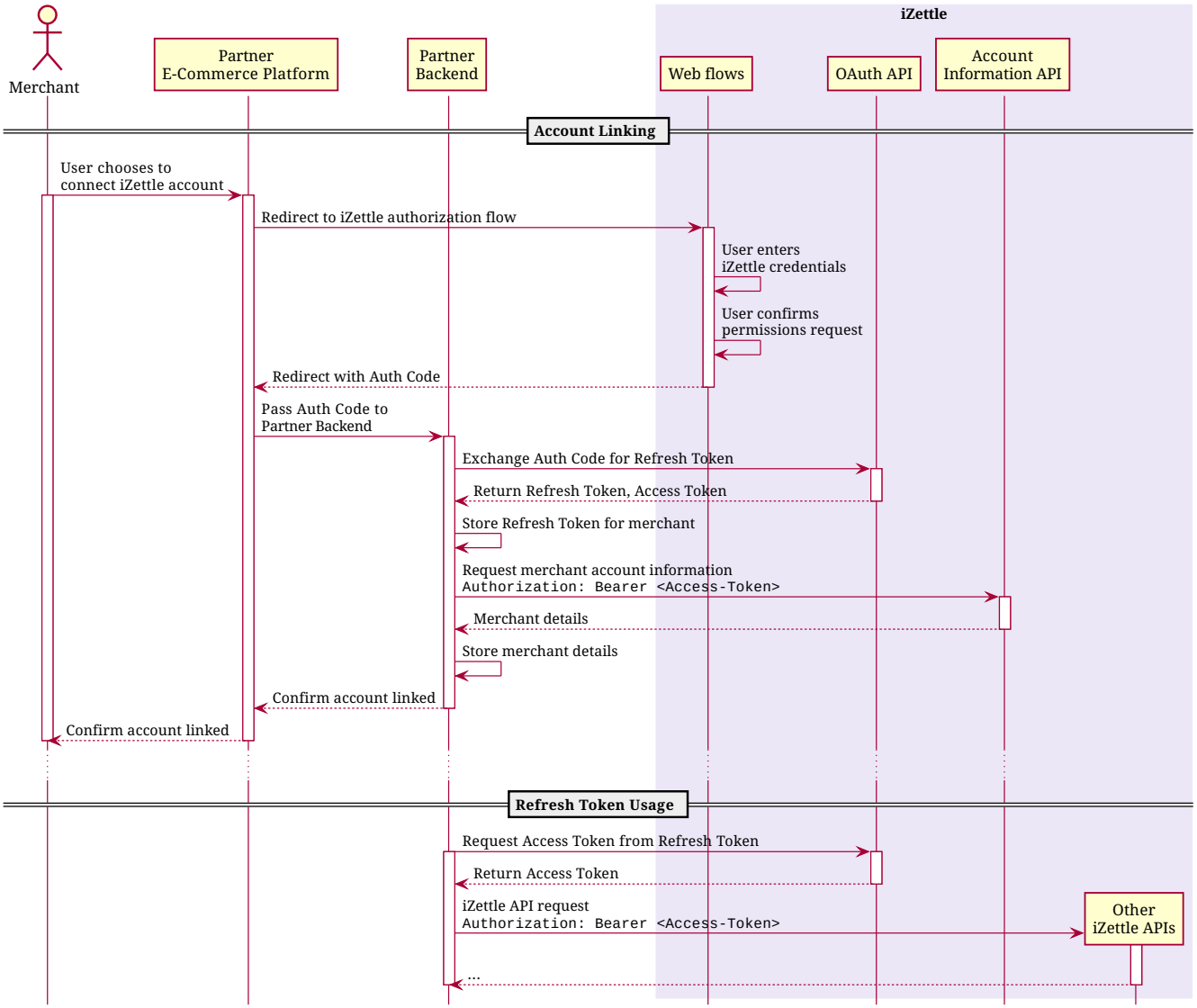
- Partner-hosted installations (SaaS): OAuth 3rd-party
- Merchant-hosted installations: API Keys

Password Grant authentication schemes (collection of a merchant's iZettle username and password by the partner) should not be used for iZettle partner integrations. Use API Keys instead.

4.3.1. OAuth 3rd-party flow

For partner-hosted integrations iZettle supports an OAuth 2.0 three-legged authentication flow with Authorization Code grant:

1. The merchant will be redirected to an iZettle login and permissions flow.
2. The merchant logs in with their iZettle credentials and confirms the permission request.
3. The browser will be redirected to a predefined URI in the partner's environment. The redirection will include an Authorization Code.
4. The partner will submit the Authorization Code to iZettle's OAuth endpoint and receives in exchange a long-lived Refresh Token representing the permission of the specific merchant (the lifetime of an iZettle Refresh Token is currently set to 180 days). The partner will store the Refresh Token with the merchant's user record.
5. The partner will retrieve relevant information about the merchant's iZettle account (e.g. the merchant's name). The partner will store the retrieved information with the merchant's user record.
6. The partner confirms to the merchant that the merchant's iZettle account has been successfully linked.
7. For any future requests to iZettle APIs to be made on behalf of the merchant, the partner will first retrieve an OAuth Access Token from the merchant's stored Refresh Token. The Access Token will be used in a Bearer auth scheme for all API requests.



4.3.1.1. Authorization Web Flow

To initiate the authorization flow to access an iZettle account, redirect the merchant to the iZettle authorization flow:

```
https://oauth.izettle.com/authorize?response_type=code&scope=<ScopeList>
&client_id=<PartnerClientID>&redirect_uri=<ReturnURL>&state=<StateValue>
&<TrackingParameters…>
```

with the following parameters (all mandatory):

Parameter	Description	Example
`ScopeList`	Space-separated list of Permission Scopes to request	`READ:PRODUCT WRITE:PRODUCT`
`PartnerClientID`	The Partner's ClientID of the (public) application (UUID v1)	`6adde977-c34d-4de1-99b2-f6ed3e65431a`

`ReturnURL`	The redirection target URI defined by the E-commerce partner in the setup if the iZettle application	`https://www.example.com/izettle/return`
`State`	A transparent parameter defined by the partner that will be passed through and included upon redirection when the flow is completed. Use this to e.g. identify a merchant's session or ID in the partner's environment. Make sure to apply url encoding when providing more complex data structures.	`abc123678-222`
`TrackingParameters…`	Additional parameters to enable tracking of partner-facilitated signups. Actual parameters will be provided by the PayPal/iZettle integration partners.	`utm_source=local_partnership&utm_medium=ecommerce&utm_campaign=theBestEcomStore`

Example:

```
https://oauth.izettle.com/authorize?response_type=code&scope=READ:PRODUCT%20WRITE:PRODUCT&client_id=6adde977-c34d-4de1-99b2-f6ed3e65431a&redirect_uri=https%3A%2F%2Fwww.example.com%2Fizettle%2Freturn&state=abc123678-
?utm_source=local_partnership&utm_medium=ecommerce&utm_campaign=theBestEcomStore
```

4.3.1.2. Exchange Auth Code for Refresh Token

After the merchant has completed the authentication flow the merchant's browser will be redirected to the registered Return URL together with an Auth Code. Use the `/token` endpoint with `authorization_code` grant to exchange the Auth Code against a long-lived Refresh Token.

The Exchange Auth Code request only has to be performed a single time after a merchant has given consent.

Note that the Exchange Auth Code response will also include an Access Token which can be immediately used to authenticate API requests on behalf of the merchant. The returned value `expires_in` is the remaining lifetime of the Access Token in seconds.

Sample Request

```
POST /token HTTP/1.1
Host: oauth.izettle.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
redirect_uri=https%3A%2F%2Fwww.example.com%2Fizettle%2Freturn&
code=ccf62ef9-99b2-4e9d-dba3-9c3f3c532992&
client_id=6adde977-c34d-4de1-99b2-f6ed3e65431a
client_secret=6adde977-99b2-408e-c34d-1727b14a398d
```

Sample Response

```
HTTP/1.1 200 Ok
Content-Type: application/json

{
    "access_token": "eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q",
    "refresh_token": "IZSEC90590831-93a5-4289-ee37-a17824c0fea1",
    "expires_in": 7200
}
```

4.3.1.3. Retrieve Access Token from Refresh Token

Use the `/token` endpoint with the `refresh_token` grant to retrieve a short-lived Access Token from the merchant's Refresh Token.

The response value `expires_in` is the remaining lifetime of the Access Token in seconds.

Sample Request

```
POST /token HTTP/1.1
Host: oauth.izettle.com
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&
refresh_token=IZSEC90590831-93a5-4289-ee37-a17824c0fea1&
client_id=6adde977-c34d-4de1-99b2-f6ed3e65431a&
client_secret=6adde977-99b2-408e-c34d-1727b14a398d
```

Sample Response

```
HTTP/1.1 200 Ok
Content-Type: application/json

{
    "access_token": "eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q",
    "refresh_token": "IZSEC90590831-93a5-4289-a178-ee0fea13724c",
    "expires_in": 7200
}
```

4.3.2. API Keys

A partner's iZettle API credentials must never be exposed to any other party. The OAuth 3rd-party flow is therefore not recommended in merchant-hosted environments. Instead, iZettle allows merchants to create API Keys to be shared with the partner. The API Key authorization scheme does not require a partner to distribute their iZettle credentials with their code base.

iZettle's API Keys are scoped merchant credentials that a merchant can create and revoke from the iZettle account pages. Partners can redirect merchants to a flow to create a new API Key, and set the following parameters:

- A suggested name for the API Key, for the merchant to recognize in the list. This could be e.g. the partner's product name.
- A list of requested permission scopes.

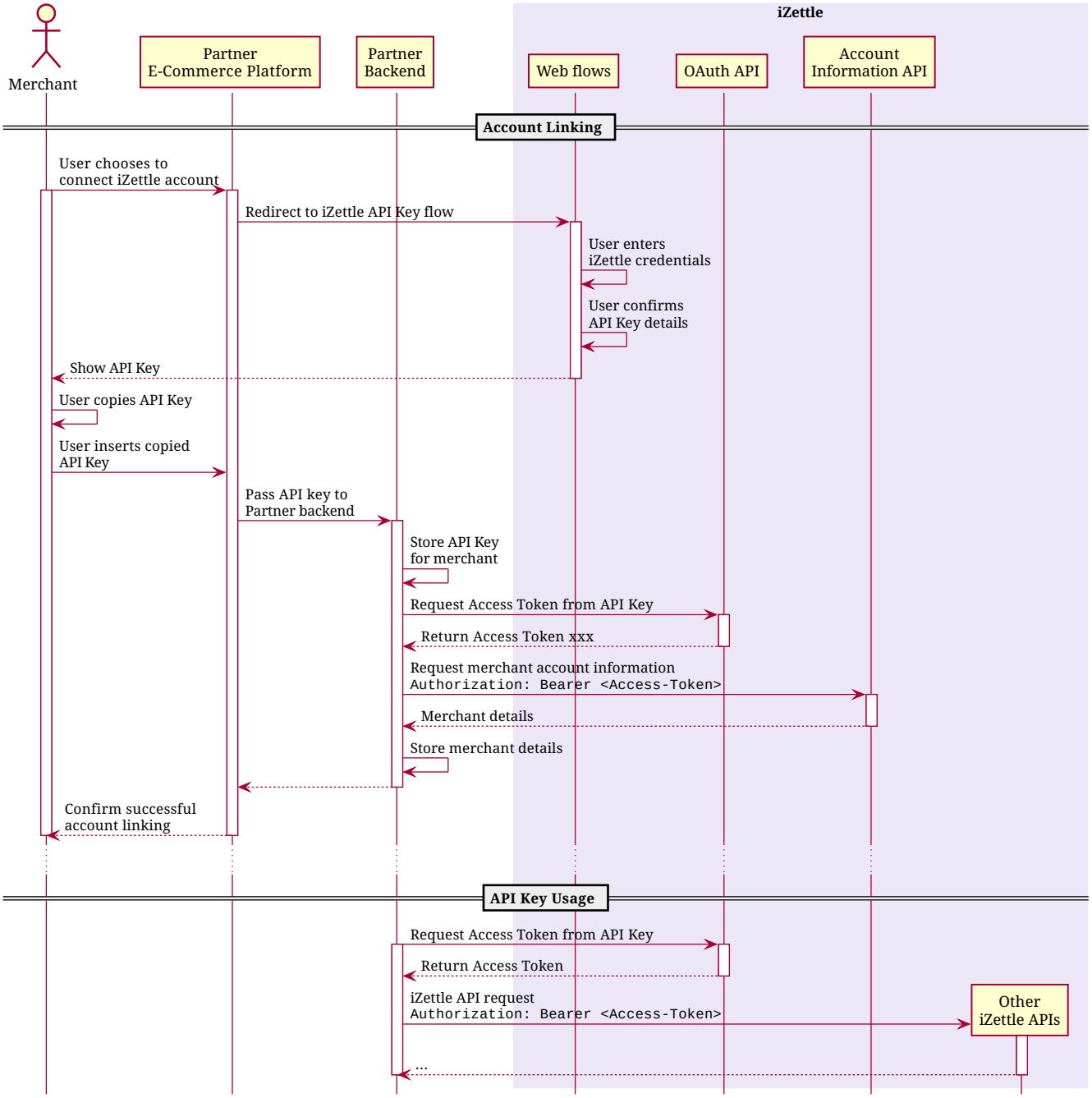
Please note that the API Key creation flow is not a fully integrated experience in the sense that it will not automatically return the merchant to the partner's environment. Instead, the merchant will have to copy the created API Key and manually enter it into the partner's environment. The API Key creation flow should therefore be opened in a new browser tab or window.

4.3.2.1. API Key flow

1. The partner redirects the merchant to the API Key creation flow in a separate window or browser tab.
2. The merchant logs in with their iZettle credentials, reviews the API Key name and requested permission scopes, and confirms the API Key creation.
3. The merchant copies API Keys from the browser window.
4. The merchant goes back to the partner's environment and pastes the copied API Key into the partner's iZettle Extension setup process.
5. The partner will store the merchant's Client ID and API Key with the merchant's user record.
6. The partner retrieves an OAuth Access Token from the merchant's stored API Key.
7. The partner will retrieve relevant information about the merchant's iZettle account (e.g. the merchant's name) using the retrieved Access Token. The partner will store the retrieved information with the merchant's user record.
8. The partner confirms to the merchant that the merchant's iZettle account has been successfully

linked.

9. For any future requests to iZettle APIs to be made on behalf of the merchant, the partner will first retrieve an OAuth Access Token from the merchant's stored API Key. The Access Token will be used in a Bearer auth scheme for all iZettle API requests.



4.3.2.2. API Key Creation Flow

To create an iZettle API Key the API Key Creation Flow should be opened for the merchant in a new window or browser tab.

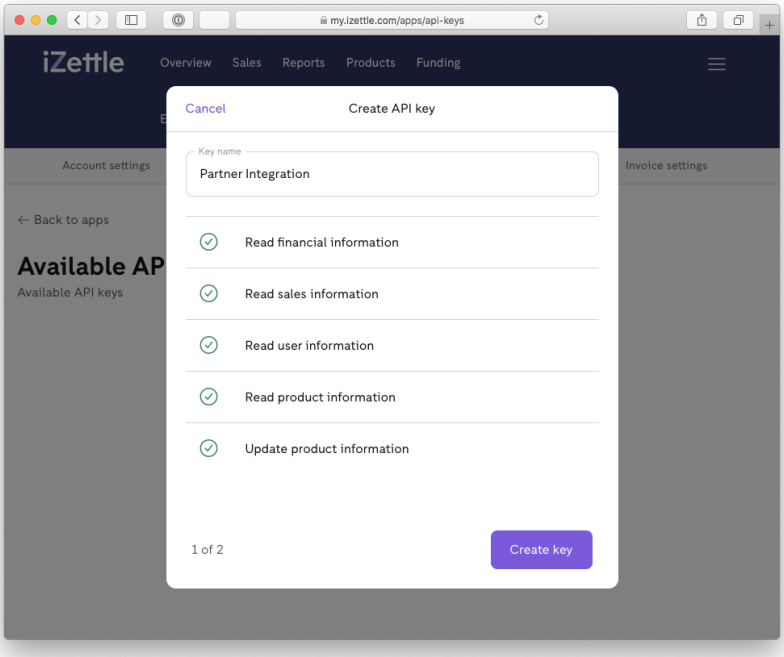


Figure 5. API Key Creation Flow – Step I

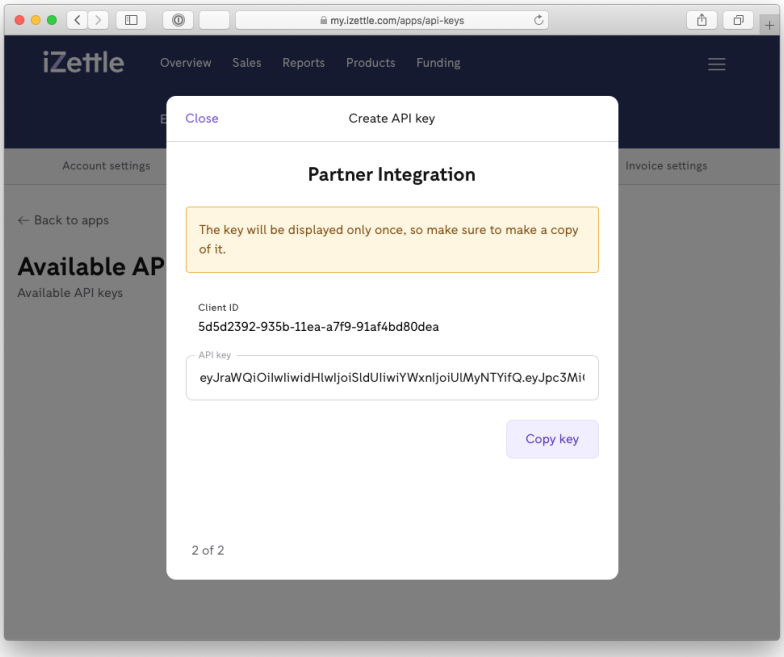


Figure 6. API Key Creation Flow – Step II

The partner redirects the merchant to the following URL:

```
https://my.izettle.com/apps/api-
keys?name=<APIKeyName>&scopes=<ScopeList>&<TrackingParameters…>
```

with the following parameters:

Parameter	Description	Example

`ScopeList`	Space-separated list of Permission Scopes to request	`READ:PRODUCT WRITE:PRODUCT`
`APIKeyName`	A suggested name for the API Key for the merchant to recognize (optional)	`Partner Integration`
`TrackingParameters…`	Additional parameters to enable tracking of partner-facilitated signups. Actual parameters will be provided by the PayPal/iZettle integration partners.	`utm_source=local_partnership&utm_medium=ecommerce&utm_campaign=theBestEcomStore`

Example:

```
https://my.izettle.com/apps/api-keys?name=Partner%20Integration&scopes=READ:FINANCE%20READ:PURCHASE%20READ:USERINFO%20READ:PRODUCT%20WRITE:PRODUCT&utm_source=local_partnership&utm_medium=ecommerce&utm_campaign=theBestEcomStore
```

4.3.2.3. Retrieve Access Token from API Key

Use the `/token` endpoint with an JWT Bearer assertion grant to retrieve a short-lived Access Token from the merchant's API Key.

The response value `expires_in` is the remaining lifetime of the Access Token in seconds.

Sample Request

```
1 POST /token HTTP/1.1
2 Host: oauth.izettle.com
3 Content-Type: application/x-www-form-urlencoded
4
5 grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&
6 assertion=eyJraWQiOiIwIiwidHlwIjoiSl...y9V15QKjn4ZgKRumYb_ikw&
7 client_id=6adde977-c34d-4de1-99b2-f6ed3e65431a
```

In the above example the `client_id` value is to be replaced with the partner's Client ID (UUID v1) to allow for Partner Affiliation Implementation.

Sample Response

```
HTTP/1.1 200 Ok
Content-Type: application/json

{
    "access_token": "eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q",
    "expires_in": 7200
}
```

4.3.2.4. API Key Revocation

A merchant may revoke an issued API Key at any time from the API Keys section of their their account page.

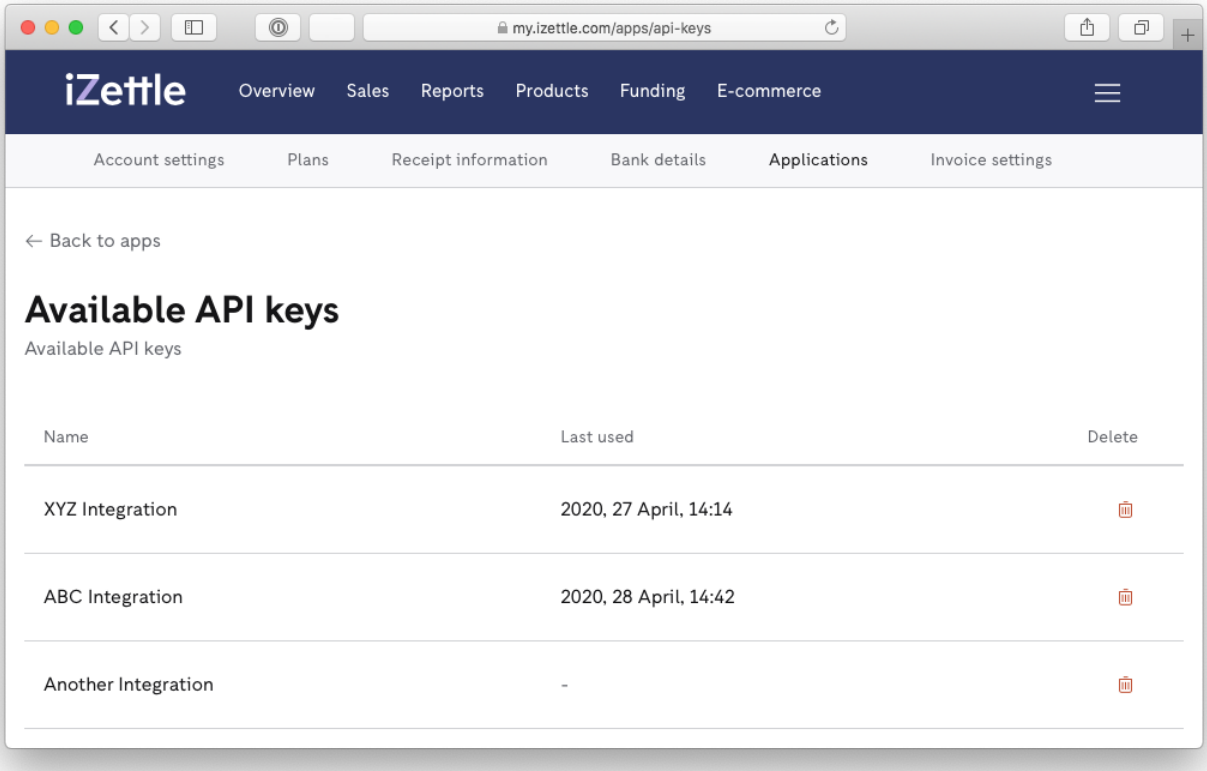


Figure 7. API Key Management Page

In case a merchant has revoked an API Key, or if the API Key has otherwise expired or is invalid, the Retrieve Access Token from API Key request will fail with an HTTP status 400 and an error as follows:

Sample Response for invalid (revoked/expired) API Key

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "invalid_grant",
    "error_description": "Invalid assertion JWT"
}
```

In the case of a revoked API Key the partner needs to guide the merchant through the API Key creation flow again.

4.3.3. Permission Scopes

The following permission scopes are available for iZettle partner integrations:

- `READ:USERINFO` – allows retrieving metadata about the merchant's iZettle account
- `READ:PRODUCT` – to retrieve product library and inventory data
- `WRITE:PRODUCT` – allows creating and updating products and inventory data
- `READ:FINANCE` – permits retrieving an iZettle merchant's account balances and transactions
- `READ:PURCHASE` – to read information about a merchant's incoming purchases made through iZettle

For partner integrations, at least the scopes `READ:USERINFO`, `READ:PRODUCT` and `WRITE:PRODUCT` should be requested.

For a phased implementation approach it may make sense to start with requesting a larger scope set than implemented in the first iterations, to avoid merchants having to re-consent once the implementation adds new features.

4.4. Account Information API

The Account Information API provides information about a merchant's iZettle account, such as the merchant's (store) name, their accepted currency etc.

4.4.1. Tax and VAT Settings Information

The Account Information API also provides information wether the merchant's iZettle account is set up to uses VAT, sales taxes, or no taxation at all for product prices. It also defines if a product's prices are to be considered including or excluding taxes.

Refer to the available information and data fields in the sample requests below.

4.4.1.1. Taxation Types

Integrations should use the tax settings information provided by the Account Information API to determine which VAT/tax model to use when synchronizing products and their prices.

`taxationType == NONE`

The account doesn't apply VAT or sales tax to the product prices.

`taxationType == VAT`

This account uses country-specific VAT rates. The response will also include a field `vatPercentage` with the default VAT rate for the iZettle merchant's country.

`taxationType == SALES_TAX`

This account uses sales tax rates that are set up and managed by the merchant in the iZettle Backoffice. Information about the available sales tax rates in the merchant's iZettle account can be retrieved with the Product Library API's Sales Tax Rates endpoints.

The previously used field `usesVAT` is redundant with `taxationType == VAT`, it is recommended to check the `taxationType` value instead.

4.4.1.2. Price Types

The Account Information API response also includes information about the user's settings for the price structure of the iZettle product library:

`taxationMode == INCLUSIVE`

Product prices in the iZettle Product Library are inclusive taxes (net prices)

`taxationMode == Exclusive`

Product prices in the iZettle Product Library are exclusive taxes (gross prices)

Note that this setting only provides information *how* product prices are stored. It in particular does not make a statement wether the merchant collects taxes at all, or which tax rates will be applied to which product (see VAT and Sales Tax).

4.4.2. Retrieve a Merchant's Account Information

Sample Request

```
1 GET /api/resources/organizations/self HTTP/1.1
2 Host: secure.izettle.com
3 Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
```

Sample Response (German Account with VAT)

```
HTTP/1.1 200 Ok
Content-Type: application/json

{
    "uuid": "0497dde4-e04e-11e9-81af-0fbace9c2068",
    "name": "Julian Dreißig",
    "receiptName": "JULIAN DREISSIG",
    "city": "BERLIN",
    "zipCode": "10823",
    "address": "HAUPTSTR. 1",
    "addressLine2": "",
    "legalName": "JULIAN DREISSIG",
    "legalAddress": "HAUPTSTR. 1",
    "legalZipCode": "10823",
    "legalCity": "BERLIN",
    "legalState": "",
    "phoneNumber": "+491771111111",
    "contactEmail": "jdreissig@paypal.com",
    "receiptEmail": "jdreissig@paypal.com",
    "legalEntityType": "COMPANY",
    "legalEntityNr": "xyz123123",
    "vatPercentage": 19.0,
    "country": "DE",
    "language": "de",
    "currency": "EUR",
    "created": "2019-09-26T11:08:50.064+0000",
    "ownerUuid": "049a25a4-e04e-11e9-805f-a3f6990e3d99",
    "organizationId": 12312312,
    "customerStatus": "ACCEPTED",
    "usesVat": true,
    "customerType": "NonLimitedCompany",
    "timeZone": "Europe/Berlin"
}
```

Sample Response (US Account with Sales Tax)

```
HTTP/1.1 200 Ok
Content-Type: application/json

{
    "uuid": "b21c8722-6a53-11eb-9191-e0167ec9f3e1",
    "name": "Demo Six Last",
    "receiptName": "Demo Six",
    "city": "Fremont",
    "zipCode": "94538",
    "address": "39455 Albany Cmn",
    "addressLine2": "",
    "legalName": "Demo Six Inc",
    "legalAddress": "39455 Albany Cmn",
    "legalZipCode": "94538",
    "legalCity": "Fremont",
    "legalState": "CA",
    "contactEmail": "uslaunche2e+n6@gmail.com",
    "legalEntityType": "UNKNOWN",
    "legalEntityNr": "734738494",
    "country": "US",
    "language": "en",
    "currency": "USD",
    "created": "2021-02-08T21:22:06.797+0000",
    "ownerUuid": "b21e1d94-6a53-11eb-98d0-56942810512a",
    "organizationId": 69999150,
    "customerStatus": "RESTRICTED",
    "usesVat": false,
    "customerType": "Partnership",
    "taxationType": "SALES_TAX",
    "taxationMode": "EXCLUSIVE",
    "timeZone": "America/Los_Angeles"
}
```

4.5. Product Library API

A merchant's iZettle product library defines the product items to be made available the merchant for selling in the iZettle Go apps. The iZettle Product Library API allows to add, manage and delete items in a merchant's iZettle product library.

The most recent and detailed technical information about the iZettle Product Library API can be found in the following places:

- The iZettle Product Library API Swagger specs
- The iZettle Product Library API documentation

Product images to appear in the iZettle product library will be managed with the Image API.

4.5.1. Products Data Model

4.5.1.1. Product Variants

Product variants define the product varieties you are offering. Each product has at least one variant. Multiple product variants can either be a list of distinct options, or may be defined along chosen dimensions defined by Product Variant Options.

4.5.1.2. Product Variant Options

Product variant options define the variant dimensions and available values a product may come in. E.g. apples may come in different colours (red, green) and in different sizes (small, medium, large) – the product variant options in this case would be

- "Size" with values `small`, `medium` and `big`
- "Colour" with values `red` and `green`.

A product can have up to three product variant options. Please note that the merchant does not need to define all value combinations as variants. In the above example, a merchant may only offer "small green" and "big red" apples.

If a product has product variant options set, the product definition will return a section `variantOptionDefinitions` with the definitions of the applicable options for this product.

4.5.1.3. Product Images

The iZettle Go app can display images for products and product variants. All images to be used within the iZettle Go app have to be made available to the iZettle system through the Image API before assigning them to products and product variants.

Images are assigned to a product through their URL on iZettle servers (as returned by the Image API).

A product can have none or a single image assigned.

4.5.1.4. Product Prices

Product and Product Variant prices are specified as an integer value in cents (e.g. `200` to represent €2.00) and the currency code. The currency must match the merchant's iZettle account currency. iZettle also allows to specify a product price per unit (e.g. per kilogram). The unit can be freely specified by setting the parameter `unitName` on the main product – the unit name will be applied to all of the product's variants. Setting `unitName` to `null` disables unit specification of the product's price.

4.5.1.5. Barcodes

Setting a product's barcode allows the merchant to add products to a user's basket by scanning the product's barcode with iZettle's barcode scanning feature.

4.5.1.6. VAT and Sales Tax Rates

VAT and tax rates can be applied for each product in the iZettle Product Library. Depending on the account type, tax is handled in different ways and a product may use or contain different fields. See section VAT and Sales Tax for further information.

4.5.1.7. External Reference

The `externalReference` field can be used to store an external identifier e.g. the product's ID in an eCommerce system.

4.5.1.8. Partner Identifier

Products created through partner integrations need to be marked as "external". This way future versions of the iZettle backoffice will be able to distinguish these products and e.g. prevent editing the product data outside of the partner's environment.

```
...
"metadata": {
    "inPos": true,
    "source": {
        "name": "<Partner Identifier>",
        "external": true
    }
}
...
```

The `<Partner Identifier>` value is specific to each integration and will be assigned by iZettle during the integration.

4.5.2. Managing Products

Products in the iZettle Library can be created, updated and deleted through the iZettle Product Library API. Product entries can be created individually or by importing a set of multiple .

4.5.2.1. Product Updates

To avoid synchronization conflicts, iZettle's Product API requires sending an eTag containing the product's latest modification timestamp as the `If-Match` header. A product update request will only be executed if the provided timestamp equals the product's latest modification date and time – otherwise the request will be rejected.

A product's latest modification timestamp will be included as `ETag` header field in the response to a "Retrieve Single Product" request.

Sample request: Retrieve Single Product

```
GET /organizations/self/products/cef7c7f0-11fb-11ea-897e-3f3dcb3e9faa
Â HTTP/1.1
Host: products.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json
ETag: "E8433CA91335FA4C61852E0B5A40AF31"

{
    "uuid": "cef7c7f0-11fb-11ea-897e-3f3dcb3e9faa",
    "name": "Apple",
    ...
}
```

Note that the eTag value typically *includes* the surrounding quotation marks.

Sample request: Update Product

```
PUT /organizations/self/products/v2/cef7c7f0-11fb-11ea-897e-
Â 3f3dcb3e9faa HTTP/1.1
Host: products.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
If-Match: "E8433CA91335FA4C61852E0B5A40AF31"

{
    "name": "Banana"
}
```

To be completed

4.5.2.2. Bulk Import

Import a list of products to the iZettle product library is done asynchronously. After the initial submission of the product definitions, the iZettle Product Library API will return an identifier for this specific import. The status of a submitted import can be requested with a separate API call.

Typical processing time for an import of e.g. 300 items is around 5 seconds. It is recommended to poll for status updates with a frequency of not less than 2 seconds and should be stopped once an import reaches a final state.

4.5.3. VAT and Sales Tax

Depending on the iZettle merchant's account country, different options are available for handling

sales tax and VAT rates for products in the iZettle Product Library. See Tax and VAT Settings Information for how to retrieve which options apply to the individual merchant's account.

Also depending on the merchant's iZettle account properties and settings, a product's price values set in the iZettle Product Library may either include or exclude taxes (gross/net values). An integration must ensure that correct net or gross price values are used when creating products in the iZettle Product Library, and that a merchant is warned about any inconsistencies between the E-Commerce Platform's and the iZettle account's tax handling that may affect the merchant.

Some of the iZettle account's tax settings can be adjusted by the merchant and may therefore change over time (e.g. the gross/net setting for US accounts). It is therefore recommended to retrieve these settings on a regular basis and always determine the current status when products in the iZettle Product Library are to be created by the integration.

4.5.3.1. No Tax Accounts

If the merchant's iZettle account does not support any taxes (`taxationType == NONE`, see Taxation Types) no additional information needs to be provided when creating products in the iZettle Product Library.

4.5.3.2. VAT accounts

In VAT countries (`taxationType == VAT`, see Taxation Types), the products in the iZettle Product Library must set the VAT percentage value (field `vatPercentage`) to either zero or to a valid VAT rates of the merchant's country. E.g. for Germany the available VAT rates are 19%, 10.7%, 7%, 5%, and 0. Trying to submit an invalid VAT percentage value will result in an error of type `VAT_NOT_ALLOWED_IN_COUNTRY`.

VAT percentage values are to be provided without the percentage sign and may contain a single decimal place, e.g.

```
...
"vatPercentage": "10.7",
...
```

The field `createWithDefaultTax = true` can be used to have the country's default VAT rate applied to the product. Note that only one of the `createWithDefaultTax` and `vatPercentage` fields can be specified when creating products.

`taxRates` and `taxExempt` fields will be ignored when creating products for VAT accounts.

4.5.3.3. Sales Tax accounts

In countries using sales tax (`taxationType == SALES_TAX`, see Taxation Types), an iZettle merchant will be able to setup multiple sales tax rates in the iZettle Backoffice. The merchant can mark one or multiple of the created sales tax rates as "default" to be assigned automatically to all

newly created products in the iZettle Backoffice.

The field `vatPercentage` must not be set if the merchant's account uses sales tax!

Assigning Default Sales Tax Rates

Include the optional field `createWithDefaultTax = true` when creating a new product in the iZettle Product Library to have the merchant's default tax rates assigned to the product. This is the same behavior as implemented by the iZettle Backoffice. When creating products via the iZettle Product Library API, this behavior needs to be explicitly defined:

Assigning Explicit Sales Tax Rates

Product definitions for accounts using sales tax can include zero, one, or multiple references to any tax rates available in the merchant's account, e.g.

```
...
"taxRates": ["2201006c-4c2c-14a6-8511-f4381dc391f7"],
"taxExempt": false,
...
```

Note that both `taxRates` and `taxExempt` parameters are optional.

Tax Exempt Products

A product in the iZettle Product Library can be marked to be exempt from sales taxes by setting the optional field `taxExempt = true`. Note that this setting is independent of the sales tax rates that are applied to the product: regardless of the assigned sales tax rates, no tax rates will be applied to this product's price for purchases through iZettle.

4.5.3.4. Tax Settings

The iZettle Product Library API provides an endpoint to retrieve just the iZettle account's tax settings. This is the same information as returned by the Account Information API, it is up to the integration which endpoint to use. See the "Taxes" endpoints in the Product Library API Swagger Definition for more details.

4.5.3.5. Sales Tax Rates

The iZettle Product Library API provides additional endpoints to create and manage a merchant's sales tax rates. See the "Taxes" endpoints in the Product Library API Swagger Definition for more details.

4.5.4. Discounts

A merchant can define discounts to be applied to a user's basket. To apply a discount, the merchant will select and add the discount like a product to the buyer's basket.

A merchant can create, list, update and delete discount entries via the iZettle Product Library API.

4.5.4.1. Create Discount

Discounts can be defined as fixed amounts (e.g. "5 €") or as percentage values (e.g. "10 %").

Sample request: create percentage-based discount

```
POST /organizations/self/discounts HTTP/1.1
Host: products.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
Content-Type: application/json

{
    "uuid": "ecb22118-fc97-1d5b-9e9e-0e7ab5fe7ebf",
    "name": "Club Members",
    "description": "10% off for Club Members",
    "percentage": "10.0",
    "externalReference": "CLUB-MEMBERS-10-PERCENT"
}
```

Sample response

```
HTTP/1.1 201 Created
```

To be completed

4.6. Image API

Product images to be visible within the iZettle Go app will be managed with the iZettle Image API.

4.6.1. Importing Images to iZettle

All images to be used for the iZettle product library have to be uploaded or imported into iZettle first. Images can be either imported from existing URLs or uploaded as image files.

Images imported to iZettle are referenced by their URLs. Please note that image URLs to use as reference for the iZettle product library have to be hosted by iZettle and start with `"https://image.izettle.com/…`.

Note that the Image API may provide the image in one or multiple resolutions. As of now, the first entry of the returned URLs of an imported image should be used.

4.6.1.1. Importing Image from URLs

Images can be imported to iZettle by providing a URLs to the image. Note that the provided image URL must be reachable by iZettle's servers and e.g. cannot be located within closed networks or on a local system. In particular, it may not be possible for iZettle to load images from a locally run test system.

Image URLs to be imported by iZettle must use the `https` protocol and have to be properly encoded (no non-ASCII characters in the path).

```
POST /v2/organizations/self/products HTTP/1.1
Host: image.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
Content-Type: application/json

{
    "imageFormat": "JPEG",
    "imageUrl": "https://picsum.photos/id/301/320/240"
}
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "imageLookupKey": "BJfd5OBOEemBrw-6zpwgaA-F1EGGBqgEeq0Zcced6LHlQ",
    "imageUrls": [
        "https://image.izettle.com/product/BJfd5OBOEemBrw-6zpwgaA-
Â F1EGGBqgEeq0Zcced6LHlQ.jpeg"
    ]
}
```

4.6.1.2. Add Image from File

Image files may be uploaded directly to iZettle:

Sample request: upload image file

```
POST /organizations/self/discounts HTTP/1.1
Host: products.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
Content-Type: multipart/form-data; boundary=----MultipartBoundary

--MultipartBoundary
Content-Disposition: form-data; name="file"; filename="apple.jpg"
Content-Type: image/jpeg

... (binary content) ...
--MultipartBoundary--
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "imageLookupKey": "BJfd5OBOEemBrw-6zpwgaA-wZMJqhqgEeqmn14A0Mlvow",
    "imageUrls": [
        "https://image.izettle.com/product/BJfd5OBOEemBrw-6zpwgaA-
Â wZMJqhqgEeqmn14A0Mlvow.jpeg"
    ]
}
```

4.6.1.3. Add Multiple Images via URLs

Sample request

```
POST /v2/organizations/self/products/bulk HTTP/1.1
Host: image.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
Content-Type: application/json

{
    "imageUploads": [
        {
            "imageFormat": "JPEG",
            "imageUrl": "https://picsum.photos/id/300/320/240"
        },
        {
            "imageFormat": "JPEG",
            "imageUrl": "https://picsum.photos/id/301/320/240"
        }
    ]
}
```

Sample response

In the following response, iZettle was able to import one of the provided images but failed to import another one.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "uploaded": [
        {
            "imageLookupKey": "BJfd5OBOEemBrw-6zpwgaA-
Â  v_jxfhSVEeqQbOh92zNC7Q",
            "imageUrls": [
                "https://image.izettle.com/product/BJfd5OBOEemBrw-
Â  6zpwgaA-v_jxfhSVEeqQbOh92zNC7Q.jpeg"
            ],
            "source": "https://picsum.photos/id/907/320/240"
        }
    ],
    "invalid": [
        "https://picsum.photos/id/11/320/240"
    ]
}
```

4.7. Inventory API

iZettle inventory handling and tracking can be managed with the iZettle Inventory API.

4.7.1. Inventory Lifecycle and Locations

iZettle tracks inventory by moving product items between so-called "Locations". Generally speaking a Location is a bucket tracking individual balances for a number of products. Typically, an iZettle merchant has a single Location corresponding to the merchant's storeroom or warehouse. iZettle locations corresponding to such a physical store are of type `STORE`.

The iZettle systems also knows a number of virtual Location types for which no balances are tracked:

`SOLD`
: Items that have been sold are moved to this Location.

`BIN`
: Move items to this location to remove/void them from inventory.

`SUPPLIER`
: Restocked items are moved into inventory from this location type.

> *To be completed*: Add Lifecycle diagram

4.7.1.1. Retrieve Location UUIDs

To access the UUIDs of the merchant's Locations, request a list of all Location data.

Sample request

```
GET /organizations/self/locations HTTP/1.1
Host: inventory.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "uuid": "258c7d57-0493-12ad-9897-64f5689308e5",
        "type": "STORE",
        "name": "Location 1573831448",
        "description": "System generated",
        "default": true
    },
    {
        "uuid": "bf6044b8-e04e-11e9-ac5d-b31e359f4e34",
        "type": "SUPPLIER",
        "name": "SUPPLIER",
        "description": "System generated",
        "default": false
    },
    {
        "uuid": "bf6045e4-e04e-11e9-a598-d3d649f988ff",
        "type": "SOLD",
        "name": "SOLD",
        "description": "System generated",
        "default": false
    },
    {
        "uuid": "bf6045f8-e04e-11e9-ad8a-3d9e0fb8149d",
        "type": "BIN",
        "name": "BIN",
        "description": "System generated",
        "default": false
    }
]
```

4.7.2. Managing Inventory Tracking

4.7.2.1. Enabling Inventory Tracking

Inventory tracking in iZettle is enabled on a product basis and can be enabled/disabled at any time.

Sample request

Inventory tracking for a product in the merchant's STORE location is enabled by the following request:

```
POST /organizations/self/inventory HTTP/1.1
Host: inventory.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
Content-Type: application/json

{
    "productUuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
}
```

Sample response

The response will contain product variants with non-zero inventory balances and the UUID of the Location the product is tracked in.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
    "variants": [
        {
            "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
            "locationType": "STORE",
            "productUuid": "c0aec1a0-e04e-11e9-9ee1-d7179a457a09",
            "variantUuid": "457a1240-1290-11ea-b72a-9775414fee2b",
            "balance": "10"
        }
    ]
}
```

4.7.2.2. Disabling Inventory Tracking

Similarly to Enabling Inventory Tracking, the tracking of a particular product can be stopped at any time.

Sample request

Inventory tracking for a product in the merchant's STORE location is disabled by a DELETE request with the product's UUID:

```
1 DELETE /organizations/self/inventory/products/3fa85f64-5717-4562-b3fc-
Â 2c963f66afa6 HTTP/1.1
2 Host: inventory.izettle.com
3 Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
```

Sample response

```
1 HTTP/1.1 204 No Content
```

4.7.2.3. Accessing Inventory Balances

Tracked product inventory balances can be accessed individually or as list of all tracked items for the Location.

Sample request: list all tracked product balances

Provide the Location's UUID to the following request:

```
1 GET /organizations/self/inventory/locations/258c7d57-0493-12ad-9897-
Â 64f5689308e5 HTTP/1.1
2 Host: inventory.izettle.com
3 Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
```

Sample response: list all tracked product balances

The response will contain the STORE Location ID and the product variants with non-zero inventory balances.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
    "trackedProducts": [
        "c6ccd657-7588-1fb2-ad13-96abaab57c1c",
        "c0aec1a0-e04e-11e9-9ee1-d7179a457a09",
        "ef04428c-45f6-181b-958f-170c91b5ff4b"
    ],
    "variants": [
        {
            "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
            "locationType": "STORE",
            "productUuid": "c0aec1a0-e04e-11e9-9ee1-d7179a457a09",
            "variantUuid": "457a1240-1290-11ea-b72a-9775414fee2b",
            "balance": "10"
        }
    ]
}
```

Sample request: balances for specified product

Provide the Location's UUID and a product UUID to the following request:

```
GET /organizations/self/inventory/locations/258c7d57-0493-12ad-9897-
Â 64f5689308e5/products/c0aec1a0-e04e-11e9-9ee1-d7179a457a09 HTTP/1.1
Host: inventory.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
```

Sample response: list all tracked product balances

The response will contain the STORE Location ID and the product variants with non-zero inventory balances.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
    "variants": [
        {
            "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
            "locationType": "STORE",
            "productUuid": "c0aec1a0-e04e-11e9-9ee1-d7179a457a09",
            "variantUuid": "457a1240-1290-11ea-b72a-9775414fee2b",
            "balance": "10"
        }
    ]
}
```

4.7.3. Updating Inventory Balances

A product's inventory balance is updated by transferring an inventory amount for a product variant from one Location to another.

Purchase

Move from `STORE` to `SOLD`

Restocking

Move from `SUPPLIER` to `STORE`

Voiding inventory

Move from `STORE` to `BIN`

Note that multiple updates can be made with a single request.

Inventory balance update requests will trigger webhook event notifications for balance updates similar to purchase events. To be able to distinguish and identify events caused by the partner's explicit update requests a parameter `externalUuid` may be provided in the update request. This parameter will be passed through and included in the webhook event payload.

4.7.3.1. Purchase Updates

Purchases are tracked by moving inventory from the merchant's `STORE` Location to the `SOLD` Location.

Sample request

```
PUT /organizations/self/inventory HTTP/1.1
Host: inventory.izettle.com
Authorization: Bearer eyJraWQiOiIxN...R5Y6FDNTva7esJ5Q
Content-Type: application/json

{
    "changes": [
        {
            "productUuid": "c0aec1a0-e04e-11e9-9ee1-d7179a457a09",
            "variantUuid": "457a1240-1290-11ea-b72a-9775414fee2b",
            "change": "1",
            "fromLocationUuid": "258c7d57-0493-12ad-9897-
Â  64f5689308e5",
            "toLocationUuid": "bf6045e4-e04e-11e9-a598-d3d649f988ff"
        }
    ],
    "externalUuid": "947d712a-892d-11ea-bc55-0242ac130003"
}
```

Sample response

The response contains the updated inventory balance of the tracked Location.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
    "variants": [
        {
            "locationUuid": "258c7d57-0493-12ad-9897-64f5689308e5",
            "locationType": "STORE",
            "productUuid": "c0aec1a0-e04e-11e9-9ee1-d7179a457a09",
            "variantUuid": "457a1240-1290-11ea-b72a-9775414fee2b",
            "balance": "9"
        }
    ]
}
```

4.8. Purchase API

Transaction reports can be accessed through the iZettle Purchase API.

To be completed

4.9. Pusher API (WebHooks)

iZettle supports actively sending notifications via WebHooks for a number of events, e.g. when the inventory balance changes. Subscriptions to notifications for such events can be managed with the iZettle Pusher API.

For iZettle, registered listeners belong to the ClientID they have been created with. That is, WebHook listeners can only be listed or deleted with an access token that belongs to the same ClientID as the one used to create the WebHook listener.

An API Key always refers to a unique ClientID. A WebHook listener can only be viewed and/or modified with the same API Key.

WebHook listeners registered with a specific are automatically deleted when an API Key has been revoked.

To be completed

4.10. Swagger files

For most of iZettle's APIs, Swagger/OpenAPI schemes are provided and can be accessed in the browser through the following links:

- Swagger documentation for iZettle Product Library API
- Swagger documentation for iZettle Image API
- Swagger documentation for iZettle Inventory API
- Swagger documentation for iZettle Purchase API
- Swagger documentation for iZettle Pusher (WebHooks) API

4.11. Postman Collection

PayPal maintains a collection of sample API requests to be used with the Postman tool. Import the collection from the following link (use the "Import from Link" option):

```
https://www.getpostman.com/collections/d027ad33b18ab187167d
```

The provided Postman collection has been developed and assembled specifically for this integration, and is intended to be used for purposes of this integration only. All information contained in the provided collection is confidential and must not be distributed and/or shared without PayPal's explicit permission.

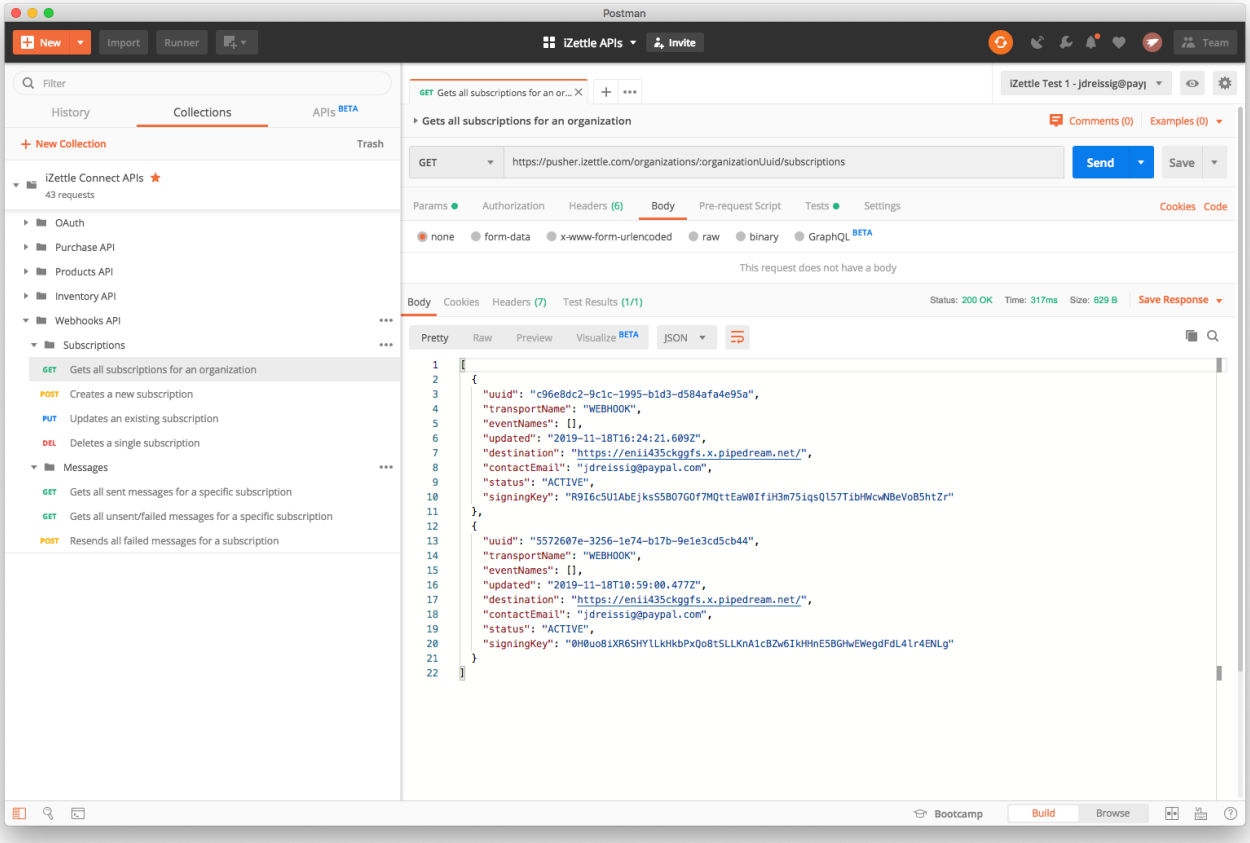


Figure 8. Postman Sample Collection

4.11.1. Usage

For full functionality the provided Postman collection expects specific environment variables to be set. Create one or more Postman Environments with the following environment variables:

Variable Name	Sample Value	Description
`client_id`	`6adde977-c34d-4de1-99b2-f6ed3e65431a`	The partner's iZettle API Client ID.
`client_secret`	`6adde977-99b2-408e-c34d-1727b14a398d`	The partner's iZettle API Client Secret.
`return_url`	`https://example.com/izettle/return`	The first of the Return URLs the partner has set up for their iZettle API Client.

The sample calls of the Postman collection are configured to work seamlessly with each other: e.g. when requesting an Access Token with the provided sample request, the retrieved token will automatically be stored in the `access_token` environment variable for use by subsequent requests of the collection. Please see the implementation of the individual sample request for the full list of required and affected environment variables.

4.11.2. Collection Updates

PayPal will continue to update and improve the provided Postman collection on an ongoing basis. To update a local copy of the collection simply re-import the collection from the above link.

5. Integration

5.1. Prerequisites

TBD: Distinction between different integration schemes (SaaS, self-hosted) needed for this section.

5.1.1. Developer Account

Registration for iZettle API credentials can be found at https://developer.izettle.com/register. *Integration Partner* will create and maintain an iZettle "Public Application" to be used for connecting iZettle merchant accounts.

5.2. Languages to be supported

To be completed

5.3. Testing Environment

To be completed

5.4. Integration Milestones

To be completed

5.5. Integration Certification

To be completed

5.5.1. Post-Live Support

To be completed

6. Frequently Asked Questions

Products created via API are not shown in the iZettle Go app

Check that the defined currency of a product's or product variant's price matches the currency of the user's account.

My generated UUID is not accepted when generating an entity with the iZettle API.

iZettle expects time-based version 1 UUIDs as identifiers for entities in the iZettle product library. As a workaround if only random/pseudo-random version 4 UUID generators are available it may be possible to simply change the version indicator of a generated v4 UUID to indicate v1 by setting the high byte of 7th octet to value `0x01`:
`ef63cf54-e04e-4`**`4`**`11e9-a453-353536383939` → `ef63cf54-e04e-`**`1`**`1e9-a453-353536383939`
Note that this workaround may not be sufficiently consistent for production use.

How can I delete all products in the iZettle library

iZettle's bulk deletion API method can be used to delete multiple products. Due to the gemeral limitation of a URL's maximum length, up to 46 products can be deleted at a time. To clean up a large number of products during testing the provided Postman Collection includes a tool suite "Runner: Delete All Products" that can be used with a Postman Runner to iteratively delete all products in a library.

7. Points of Contact

- PayPal Solution Engineering: Julian Dreissig
- Partner Manager: …
- Technical Account Manager: …
- …

> *To be completed*
