SmartPass API Integration

CHANGELOG

Version	Date	Changes
v1.0	2026-10-09	Initial release.

SMARTPASS API INTEGRATION

SmartPass lets you issue digital passes, such as event tickets, membership cards, business cards or vouchers, that your users save to Apple Wallet and Google Wallet. Sparados generates and signs the passes for both platforms, hosts the wallet web services, keeps saved passes up to date on users' devices, and handles expiry and voiding. You only decide who gets a pass and what is printed on it.

The work is split between the Sparados Corporate Panel and the Sparados API:

- **Pass templates are created in the Corporate Panel** (SmartPass > Campaigns). A template defines the look of the pass, its texts and languages, the barcode format and which fields are filled per pass. Templates cannot be created or edited through the API; the API gives read access to them.
- **Passes can be issued in three ways:** in the Corporate Panel one by one (Issue pass), in the Corporate Panel in bulk from a CSV file (Bulk issue), or through the Sparados API, either one at a time or as a batch. This article describes the API.

SmartPass API allows partners to:

- list pass templates and read their field definitions
- issue a single pass or many passes in one batch
- deliver passes by Sparados email or through your own channel using add-to-wallet links and QR codes
- check whether a pass has been saved to Apple Wallet or Google Wallet
- update a pass that is already in the user's wallet
- void a pass or let it expire automatically

Only issued passes count toward your SmartPass pass limit. Updating a pass, sending wallet notifications and creating templates do not use the limit, and access to SmartPass API has no extra fee. SmartPass can be used on its own or together with Sparados virtual cards, for example to show a card budget on a pass next to the payment card in the same wallet. More about the product: sparados.com/smartpass.

Before using SmartPass API, the partner must complete onboarding with Sparados, sign the cooperation agreement, configure Mutual TLS authentication for the selected environment, and create at least one pass template in the Corporate Panel.

API Endpoints

API	BETA	PROD
SmartPass API, Passes (swagger)	https://sp-api.secure-verestro.dev	https://sp-api.secure-verestro.com

1. SERVER-TO-SERVER SECURITY

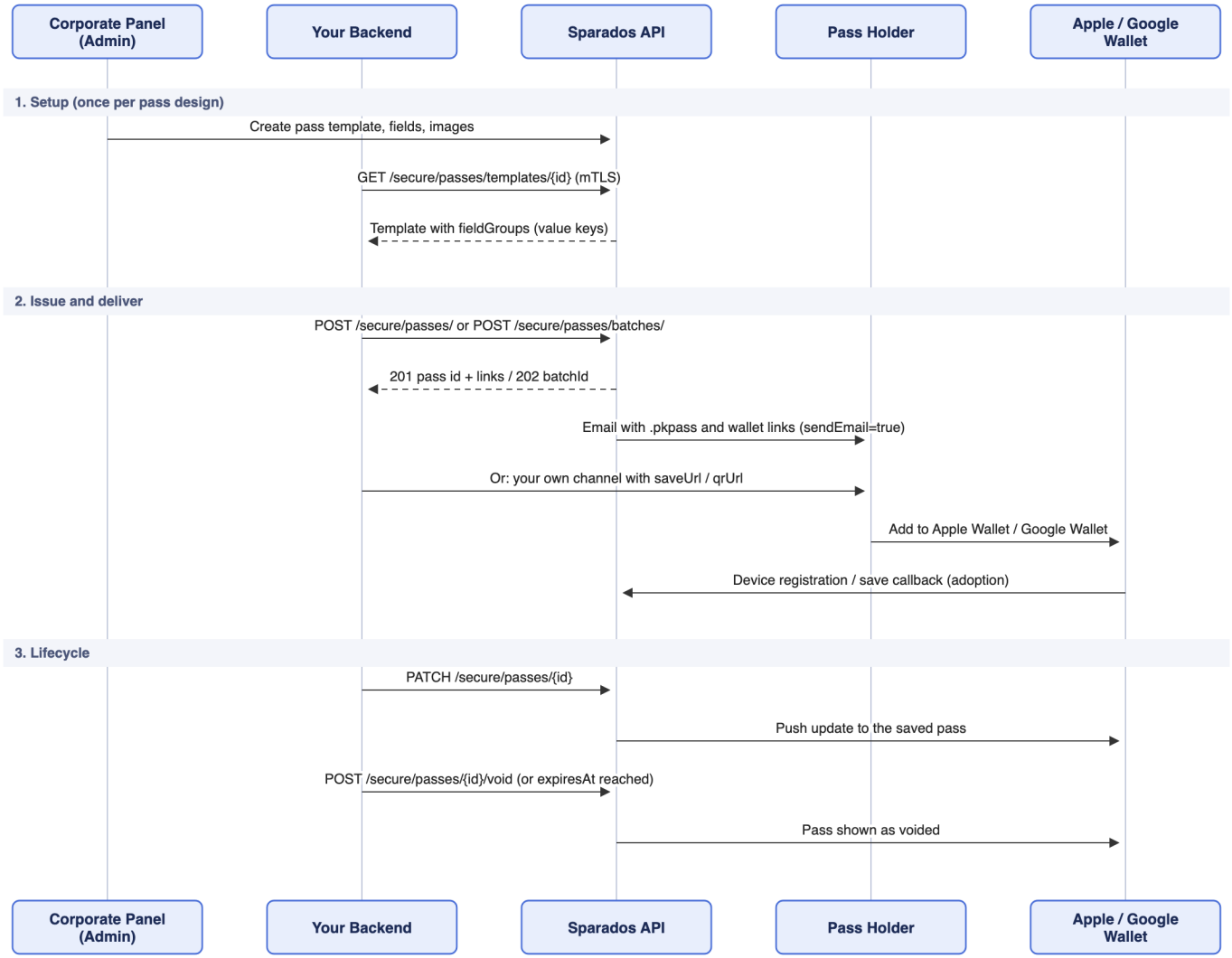
All SmartPass API requests require Mutual TLS (mTLS), the same as the other Sparados server-to-server APIs. Instructions for generating the CSR and configuring the client certificate are available in Connecting to server-to-server APIs.

2. HOW SMARTPASS WORKS

How it works

1. An administrator creates a pass template in the Corporate Panel: layout, colors, logo and icon, texts in the supported languages, barcode format, and the fields shown on the pass. Each field is either the same for everyone, or filled per pass under a value key such as `holderName`.
2. Your backend reads the template with `GET /secure/passes/templates/{id}` to learn which value keys each pass must provide.
3. Your backend issues passes with `POST /secure/passes/` (one pass) or `POST /secure/passes/batches/` (many passes).
4. The pass reaches the holder either by an email sent by Sparados (`sendEmail: true`), or through your own channel using the add-to-wallet links and QR codes returned by the API.
5. The holder adds the pass to Apple Wallet or Google Wallet. Sparados records this as adoption, which you can read from the API.
6. Later you can update the pass with `PATCH /secure/passes/{id}`, and the change is pushed to the holder's device, or void it with `POST /secure/passes/{id}/void`. A pass with `expiresAt` is voided automatically when that time passes.

Sequence diagram



3. KEY CONCEPTS

Concept	Corporate Panel	Description
Template	Campaign	Design and structure of a pass. Created and edited only in the Corporate Panel. Identified by `templateId`.
Pass	Pass	One pass issued to one holder, identified by `id`. The pass `id` is also the pass serial number used by Apple Wallet and Google Wallet.
Batch	Batch	A bulk issuing job created from a list of rows (API) or a CSV file (Corporate Panel). Processed in the background.
Field values	Filled per pass	Per-pass values sent in `fieldValues`, keyed by the value keys defined in the template.
Adoption	Wallets (Apple / Google icons)	Whether the holder has saved the pass to Apple Wallet or Google Wallet.

Concept	Corporate Panel	Description
Void	Void	Final deactivation of a pass. The pass stays on the list and is shown as voided in the wallet.

Field groups

Fields of a template are placed in zones. The zone decides where the field is displayed on the pass:

Zone (API)	Corporate Panel	Where it is displayed
`header`	Header	Small text in the top right corner, next to the logo.
`primary`	Primary	Large text, typically the holder's name.
`secondary`	Secondary	Row below the primary fields.
`auxiliary`	Auxiliary	Extra row below the secondary fields.
`back`	Back / details	Back of the pass in Apple Wallet, details view in Google Wallet.

Field source

Each field has a `source` that decides where its value comes from:

source.type	Corporate Panel	Description
`static`	Same for everyone	The text is set in the template (`source.value`, per language) and shared by all passes.
`field_values`	Filled per pass	Each pass supplies its own value in `fieldValues` under `source.key`. In the Corporate Panel CSV this key is also the column name.

Example field definitions returned in `fieldGroups`:

```
{
  "primary": [
    {
      "key": "holder",
      "label": { "en": "Holder" },
      "source": { "type": "field_values", "key": "holderName" },
      "notifyOnChange": false
    }
  ],
```

```
  "secondary": [
    {
      "key": "event",
      "label": { "en": "Event" },
      "source": { "type": "static", "value": { "en": "Sparados Developer Day" } },
      "notifyOnChange": false
    }
  ]
}
```

When `notifyOnChange` is `true`, the holder receives a wallet notification whenever the value of this field changes. `changeMessage` can define the notification text per language.

4. VALUES USED IN SMARTPASS API

Detailed Swagger documentation is available at the end of this document.

Date format

All dates use ISO 8601 date-time format in UTC:

```
2026-12-31T23:59:00Z
```

The Corporate Panel displays dates in the local time zone of the administrator, so `2026-12-31T23:59:00Z` is shown as `2027-01-01 00:59` in Poland in winter.

Identifiers

Templates, passes and batches are identified by UUIDs.

Languages

Texts such as `organizationName`, `passTitle`, field labels and static values are returned as objects keyed by language code, for example `{ "en": "Holder", "pl": "Posiadacz" }`. Each template has a `defaultLanguage`, used when a translation is missing, and a list of `supportedLanguages`. The `language` parameter used when issuing passes selects the language of the email sent to the holder.

Barcode formats

The barcode format is set in the template: `QR`, `PDF417`, `AZTEC` or `CODE128`. The content of the barcode is set per pass in `barcodeMessage`, and an optional text displayed under the barcode in `barcodeAltText`. A pass without `barcodeMessage` has no barcode.

Layouts

The API returns the template layout as one of: `boardingPass`, `storeCard`, `eventTicket`, `coupon`, `generic`. The Corporate Panel currently creates templates with the `eventTicket` layout.

Pagination and sorting

List endpoints use cursor-based pagination. Pass the `nextCursor` value from the previous response as `cursor` to fetch the next page. When `nextCursor` is `null`, there are no more results.

Parameter	Type	Description
cursor	string	Opaque cursor from a previous response's `nextCursor`.
limit	integer	Max results per page. Default 20, up to 100. For batch rows: default 100, up to 1000.
orderBy	string	Sorting field. Default `createdAt`. Available values depend on the endpoint.
orderDirection	string	`asc` or `desc`. Default `desc`.

Errors

When a template, pass or batch does not exist, the API returns `404 Not Found`:

```
{
  "reason": "RESOURCE_NOT_FOUND",
  "detail": "Pass not found"
}
```

5. PASS TEMPLATES

Templates are created in the Corporate Panel. Through the API you can list them and read their structure.

Template API Methods

Method	Endpoint	Description
GET	/secure/passes/templates/	Retrieves the list of pass templates.
GET	/secure/passes/templates/{id}	Retrieves details of a specific template, including `fieldGroups`.

LIST TEMPLATES

```
GET /secure/passes/templates/
```

Query parameters

Field	Type	Description
name	string	Filter by template name.
layout	string	Filter by layout: `boardingPass`, `storeCard`, `eventTicket`, `coupon`, `generic`.
createdAtFrom / createdAtTo	string (date-time)	Filter by creation date.
updatedAtFrom / updatedAtTo	string (date-time)	Filter by last update date.
orderBy	string	`createdAt`, `updatedAt`, `name`, `id`. Default `createdAt`.
orderDirection, cursor, limit		See Pagination and sorting.

GET TEMPLATE DETAILS

```
GET /secure/passes/templates/{id}
```

Returns the template with its `fieldGroups`. Every key referenced by a field with source type `field_values` must be sent in the `fieldValues` of each pass issued from this template.

Example response:


```
{
  "id": "6f1c2a9e-4b7d-4c1e-9a3f-2d5e8b7c1a40",
  "corporationId": "1b2c3d4e-5f60-4718-8293-a4b5c6d7e8f9",
  "name": "Developer Day 2026",
  "layout": "eventTicket",
  "organizationName": { "en": "Sparados" },
  "passTitle": { "en": "Developer Zone Pass" },
  "backgroundColor": "#0B1D33",
  "foregroundColor": "#FFFFFF",
  "labelColor": "#8FB4FF",
  "barcodeFormat": "QR",
  "emailTemplate": null,
  "defaultLanguage": "en",
  "supportedLanguages": ["en"],
  "fieldGroups": {
    "header": [],
    "primary": [
      {
        "key": "holder",
```

```
        "label": { "en": "Holder" },
        "source": { "type": "field_values", "key": "holderName" },
        "notifyOnChange": false
      }
    ],
    "secondary": [
      {
        "key": "event",
        "label": { "en": "Event" },
        "source": { "type": "static", "value": { "en": "Sparados Developer Day" } },
        "notifyOnChange": false
      }
    ],
    "auxiliary": [],
    "back": []
  },
  "logoUrl": "https://<assets-host>/logo.png",
  "iconUrl": "https://<assets-host>/icon.png",
  "stripImageUrl": null,
  "createdAt": "2026-10-09T14:29:00Z",
  "updatedAt": "2026-10-09T14:29:00Z",
  "activePassCount": 2,
  "adoptedPassCount": 0,
  "totalPassCount": 3
}
```


Response fields

Field	Type	Description
id	string (uuid)	Template ID. Use it as `templateId` when issuing passes.
name	string	Internal template name (Campaign name in the Corporate Panel). Never shown to pass holders.
layout	string	Pass layout.
organizationName	object	Issuer name per language. Apple: shown in notifications. Google: small title at the top of the pass.
passTitle	object, nullable	Card heading per language. Google Wallet only.
backgroundColor	string, nullable	Background color, Apple and Google.

Field	Type	Description
foregroundColor / labelColor	string, nullable	Text and label colors, Apple Wallet only.
barcodeFormat	string, nullable	`QR`, `PDF417`, `AZTEC` or `CODE128`.
emailTemplate	string, nullable	Custom email template for the issued-pass email. `null` uses the default email.
defaultLanguage / supportedLanguages	string / array	Default language and all languages the texts are available in.
fieldGroups	object	Field definitions per zone: `header`, `primary`, `secondary`, `auxiliary`, `back`. See Key concepts.
logoUrl / iconUrl / stripImageUrl	string, nullable	Public URLs of the template images.
activePassCount	integer	Number of active passes issued from this template.
adoptedPassCount	integer	Number of active passes saved to Apple Wallet or Google Wallet.
totalPassCount	integer	Number of all passes issued from this template, including voided.

6. ISSUE A PASS

To issue a single pass, use:

```
POST /secure/passes/
```

To issue many passes from one template, use `POST /secure/passes/batches/` instead of calling this endpoint once per pass. See section 11.

Request body example:

```
{
  "templateId": "6f1c2a9e-4b7d-4c1e-9a3f-2d5e8b7c1a40",
  "userEmail": "john.smith@example.com",
  "fieldValues": {
    "holderName": "John Smith"
  },
  "barcodeMessage": "TICKET-000123",
  "barcodeAltText": "TICKET-000123",
  "expiresAt": "2026-12-31T23:59:00Z",
  "language": "en",
  "sendEmail": true
```

```
}
```

Request body fields:

Field	Type	Description
templateId*	string, uuid	ID of the template the pass is issued from.
userEmail*	string, email	Email address of the pass holder. Used for the issued-pass email and for searching passes.
fieldValues	object	Per-pass values keyed by the value keys of the template. Must contain every key referenced by a `field_values` field. Default `{}`.
barcodeMessage	string, nullable	Data encoded in the barcode. Without it the pass has no barcode.
barcodeAltText	string, nullable	Text displayed under the barcode.
expiresAt	string (date-time), nullable	Expiry time of the pass. When it passes, the pass is voided automatically. Omit for a pass that does not expire.
language	string	Language of the issued-pass email.
sendEmail	boolean	When `true`, Sparados emails the holder the `.pkpass` file and add-to-wallet links for both platforms. Default `false`.

Additional properties are not accepted by the API.

Example response (`201 Created`):

```
{
  "id": "9c4e7b1a-2f3d-4a5b-8c6d-7e8f9a0b1c2d",
  "corporationId": "1b2c3d4e-5f60-4718-8293-a4b5c6d7e8f9",
  "templateId": "6f1c2a9e-4b7d-4c1e-9a3f-2d5e8b7c1a40",
  "templateName": "Developer Day 2026",
  "userEmail": "john.smith@example.com",
  "fieldValues": { "holderName": "John Smith" },
  "barcodeMessage": "TICKET-000123",
  "barcodeAltText": "TICKET-000123",
  "expiresAt": "2026-12-31T23:59:00Z",
  "googleObjectId": "<issuerId>.9c4e7b1a-2f3d-4a5b-8c6d-7e8f9a0b1c2d",
  "status": "active",
```

```
  "voidedAt": null,
  "adoption": {
    "apple": { "devices": [] },
    "google": { "savedAt": null, "deletedAt": null }
  },
  "createdAt": "2026-10-09T14:31:00Z",
  "updatedAt": "2026-10-09T14:31:00Z",
  "links": {
    "apple": {
      "saveUrl": "https://<pass-host>/passes/9c4e7b1a-2f3d-4a5b-8c6d-7e8f9a0b1c2d.pkpass",
      "qrUrl": "https://<pass-host>/passes/9c4e7b1a-2f3d-4a5b-8c6d-7e8f9a0b1c2d/qr/apple"
    },
    "google": {
      "saveUrl": "https://pay.google.com/gp/v/save/<signed-token>",
      "qrUrl": "https://<pass-host>/passes/9c4e7b1a-2f3d-4a5b-8c6d-7e8f9a0b1c2d/qr/google"
    }
  }
}
```

Response fields

Field	Type	Description
id	string (uuid)	Pass ID, also the pass serial number. Store it to update or void the pass later.
templateId / templateName	string	Template the pass was issued from.
userEmail	string	Pass holder's email.
fieldValues	object	Per-pass field values.
barcodeMessage / barcodeAltText	string, nullable	Barcode content and text under the barcode.
expiresAt	string (date-time), nullable	Expiry time, or `null` if the pass does not expire.
googleObjectId	string, nullable	ID of the pass object in Google Wallet.
status	string	`active` or `voided`.
voidedAt	string (date-time), nullable	When the pass was voided.
adoption	object	Wallet save status. See section 8.

Field	Type	Description
links	object	Add-to-wallet links (`saveUrl`) and QR code images (`qrUrl`) for Apple and Google. `null` for a platform the pass was not issued on. Returned by `POST /secure/passes/` and `GET /secure/passes/{id}`.
createdAt / updatedAt	string (date-time)	Creation and last update time.

7. DELIVERING THE PASS TO THE HOLDER

There are two ways to get the pass to the holder. You can use one or both.

Email sent by Sparados

Set `sendEmail: true` when issuing. Sparados sends an email to `userEmail` with the `.pkpass` file attached and add-to-wallet links for Apple Wallet and Google Wallet. The email uses the template's `emailTemplate` if one is configured, otherwise the default Sparados email, in the language selected by `language`.

Your own channel

Use the `links` returned by `POST /secure/passes/` or `GET /secure/passes/{id}`:

Link	Use
links.apple.saveUrl	Opens or downloads the pass for Apple Wallet. Show it as an Add to Apple Wallet button on iOS.
links.google.saveUrl	Opens the Google Wallet save page. Show it as an Add to Google Wallet button on Android.
links.apple.qrUrl / links.google.qrUrl	Image of a QR code that leads to the matching `saveUrl`. Useful on desktop, on printed materials or on a screen at an event, where the user scans it with a phone.

The links and QR codes are public and scoped to a single pass. Share them only with the pass holder.

8. TRACKING WALLET ADOPTION

The `adoption` object shows whether the holder has saved the pass:

Field	Type	Description
adoption.apple.devices	array	Apple devices on which the pass is saved. Each item contains `deviceLibraryIdentifier` and `registeredAt`. Empty when the pass is not in Apple Wallet.

Field	Type	Description
adoption.google.savedAt	string (date-time), nullable	When the pass was saved to Google Wallet.
adoption.google.deletedAt	string (date-time), nullable	When the pass was removed from Google Wallet.

To find passes that were or were not saved, use `GET /secure/passes/?adopted=true` or `adopted=false`. The filter considers active passes only. A pass counts as adopted when it is registered on an Apple device or saved and not deleted in Google Wallet. Per-template totals are available in `adoptedPassCount`.

SmartPass API does not send webhook notifications. To react to adoption, query the passes periodically.

9. UPDATE A PASS

To change a pass that has already been issued, use:

```
PATCH /secure/passes/{id}
```

The change is applied to the pass on both platforms, including passes already saved in the holder's wallet. Send only the fields you want to change.

Request body example:

```
{
  "fieldValues": {
    "holderName": "John A. Smith"
  },
  "barcodeMessage": "TICKET-000123-B",
  "expiresAt": null,
  "notify": false
}
```

Request body fields:

Field	Type	Description
fieldValues	object	New per-pass field values.
barcodeMessage	string, nullable	New barcode content.
barcodeAltText	string, nullable	New text under the barcode.

Field	Type	Description
expiresAt	string (date-time), nullable	New expiry time. Sets the native expiration in Apple Wallet and the validity period and expiry notification in Google Wallet. Send `null` to remove the expiry.
notify	boolean	Default `true`. Set `false` to skip the wallet notification for fields with `notifyOnChange`, for example for a typo fix that should not alert the holder.

The API returns `200 OK` with the updated pass. A voided pass cannot be updated.

10. VOID A PASS

To deactivate a pass, use:

```
POST /secure/passes/{id}/void
```

The pass is marked as void on both platforms and is shown as voided in the holder's wallet. The record stays available in the API with `status: "voided"` and `voidedAt` set.

Voiding is final and cannot be undone. Further updates of a voided pass are rejected. To give the holder a new pass, issue a new one.

Passes with `expiresAt` are voided automatically in the same way when the expiry time passes.

11. LIST AND GET PASSES

LIST PASSES

```
GET /secure/passes/
```

Query parameters

Field	Type	Description
id	string (uuid)	Filter by pass ID.
templateId	string (uuid)	Filter by template.
userEmail	string	Filter by holder's email, substring match.
status	string	`active` or `voided`.
adopted	string	`true`: saved to a wallet. `false`: issued but not saved. Active passes only.

Field	Type	Description
createdAtFrom / createdAtTo	string (date-time)	Filter by creation date.
updatedAtFrom / updatedAtTo	string (date-time)	Filter by last update date.
expiresAtFrom / expiresAtTo	string (date-time)	Filter by expiry date.
voidedAtFrom / voidedAtTo	string (date-time)	Filter by void date.
orderBy	string	`createdAt`, `updatedAt`, `userEmail`, `expiresAt`, `voidedAt`, `id`. Default `createdAt`.
orderDirection, cursor, limit		See Pagination and sorting.

The response contains `data` (list of passes, same fields as in section 6 without `links`) and `nextCursor`.

GET PASS DETAILS

```
GET /secure/passes/{id}
```

Returns the pass together with its full `template` object and the current `links`. Use it to get the add-to-wallet links again, for example to resend them to the holder.

12. BULK ISSUING

To issue many passes from one template, use:

```
POST /secure/passes/batches/
```

Sparados validates every row first. If any row is invalid, nothing is issued and the API returns `422 Unprocessable Entity` listing the invalid rows. If all rows are valid, the API returns `202 Accepted` and issues the passes in the background. The request body can be up to 15 MB.

Request body example:

```
{
  "templateId": "6f1c2a9e-4b7d-4c1e-9a3f-2d5e8b7c1a40",
  "sendEmail": true,
  "language": "en",
  "rows": [
    {
      "userEmail": "anna.nowak@example.com",
      "fieldValues": { "holderName": "Anna Nowak" },
      "barcodeMessage": "TICKET-000124",
      "expiresAt": "2026-12-31T23:59:00Z"
```

```
    },
    {
      "userEmail": "piotr.kowalski@example.com",
      "fieldValues": { "holderName": "Piotr Kowalski" },
      "barcodeMessage": "TICKET-000125"
    }
  ]
}
```

Request body fields:

Field	Type	Description
templateId*	string, uuid	ID of the template all passes are issued from.
rows*	array	One object per pass, at least one. See row fields below.
sendEmail	boolean	When `true`, every successfully issued pass is emailed to its `userEmail` once the batch completes. Default `false`.
language	string	Language of the issued-pass emails.

Row fields:

Field	Type	Description
userEmail*	string, email	Pass holder's email.
fieldValues*	object	Every value key the template references.
barcodeMessage	string	Data encoded in the barcode.
barcodeAltText	string	Text under the barcode.
expiresAt	string (date-time)	Expiry time, must be in the future.

Example response (`202 Accepted`):

```
{
  "batchId": "952ec638-20ee-47d5-a132-408807101e26",
  "totalRows": 2
}
```

CHECKING BATCH PROGRESS

```
GET /secure/passes/batches/{id}
```

Returns the batch status and counters. When `pendingCount` reaches 0 and `status` is `completed`, every row has been processed.

Example response:


```
{
  "id": "952ec638-20ee-47d5-a132-408807101e26",
  "corporationId": "1b2c3d4e-5f60-4718-8293-a4b5c6d7e8f9",
  "templateId": "6f1c2a9e-4b7d-4c1e-9a3f-2d5e8b7c1a40",
  "templateName": "Developer Day 2026",
  "authorEmail": null,
  "status": "completed",
  "totalRows": 2,
  "successCount": 2,
  "failureCount": 0,
  "pendingCount": 0,
  "sendEmail": true,
  "language": "en",
  "createdAt": "2026-10-09T14:31:00Z",
  "updatedAt": "2026-10-09T14:31:05Z",
  "completedAt": "2026-10-09T14:31:05Z"
}
```


Field	Type	Description
status	string	`pending`, `processing`, `completed` or `failed`.
totalRows	integer	Number of rows in the batch.
successCount / failureCount / pendingCount	integer	Rows issued, rows failed, rows not processed yet.
authorEmail	string, nullable	Administrator who uploaded the batch in the Corporate Panel. `null` for batches created through the API.
completedAt	string (date-time), nullable	When processing finished.

BATCH ROW RESULTS

```
GET /secure/passes/batches/{id}/rows
```

Returns the rows in submission order with their outcome. Use `status=failed` to list only the failures. Supports `cursor` and `limit` (default 100, up to 1000).

Example response:

```
{
  "data": [
    {
      "rowNumber": 1,
      "userEmail": "anna.nowak@example.com",
      "status": "success",
      "error": null,
      "passId": "0d1e2f3a-4b5c-4d6e-8f70-819a2b3c4d5e"
    },
    {
      "rowNumber": 2,
      "userEmail": "piotr.kowalski@example.com",
      "status": "success",
      "error": null,
      "passId": "5e4d3c2b-1a09-4f8e-a7d6-c5b4a3928170"
    }
  ],
  "nextCursor": null
}
```


Store `passId` for each row so you can update or void the passes later.

LIST BATCHES

```
GET /secure/passes/batches/
```

Lists batches created through the API and in the Corporate Panel. Filters: `templateId`, `status`, `createdAtFrom` / `createdAtTo`, `updatedAtFrom` / `updatedAtTo`. Sorting by `createdAt`, `updatedAt` or `id`.

Bulk issuing in the Corporate Panel

The same batch can be created without the API in the Corporate Panel (SmartPass > Campaign > Bulk issue). The CSV file needs the column `userEmail` and one column per value key (for example `holderName`), with optional columns `barcodeMessage`, `barcodeAltText` and `expiresAt`. UTF-8, comma, semicolon or tab separated, up to 5 MB. Batches created in the panel are also visible through the API.

13. API SWAGGER DOCUMENTATION

Detailed API documentation is available here:

API Documentation	link

SmartPass API Swagger (Passes)	https://sp-api.verestro.dev/docs?urls.primaryName=External#/Passes
Connecting to server-to-server APIs	https://developer.sparados.com/books/sparados-api-documentation/page/connecting-to-server-to-server-apis
SmartPass product page	https://www.sparados.com/smartpass

Use the Swagger documentation as the source of detailed endpoint schemas, request parameters, response formats, and available enum values.

Revision #2
Created 2026-10-09 14:47:18 UTC by Michał Stachera
Updated 2026-10-12 06:13:06 UTC by Michał Stachera