Meet the API

VeloxFactory ships with a fully capable REST API, and it is not an afterthought. Every action you can perform in the frontend can also be performed via the API, because **the frontend and the API share the same controller methods**. There is no separate implementation, no feature gap, no second-class citizen.

This makes the API a genuine alternative to the UI, not just an integration bolt-on. Automate report rendering in your CI pipeline, trigger prints from your ERP, manage report configurations programmatically, have VeloxFactory run the whole thing on a schedule, all with the same logic that powers the frontend you already know.

> ℹ For the full endpoint reference including all parameters and response schemas, explore the live demo API documentation at demo.veloxfactory.dev.
>
> **Login:** `demo@veloxfactory.dev` · **Password:** `demo`

Authentication

The API uses **token-based authentication** via Laravel Sanctum. Every request must include a Bearer token in the `Authorization` header.

```
Authorization:                              Bearer                                  your-api-token
```

Tokens are created and managed per user, either from the user management section in the frontend, or via the API itself (`POST /api/v1/user/{id}/create-token`). Each token can be revoked at any time. A revoked token is rejected immediately, there is no grace period.

```
{

                "errors":       ["This       API       token       has       been

}
```

The audit trail records which token was used for every create and update operation across all models, so every API action is fully traceable.

Base URL

All API v1 endpoints are prefixed with:

```
/api/v1/
```

Referencing Records

Wherever a request points at another record, it accepts either the numeric ID or the unique name of that record. These two are the same call:

```
POST                                                    /api/v1/report-config/12/render
POST                          /api/v1/report-config/Shift                       Report/render
```

The same holds inside request bodies, for report contexts, connection configs, mailers, mail templates and schedules alike, so a payload stays readable without a lookup table of IDs. A Report History Record is addressed by its ID or by its trace id, which is what makes a trace id worth carrying through your own systems.

> **i A value of digits only is always read as an ID.** That is why a name cannot consist of digits only. VeloxFactory refuses such a name when a record is created or renamed, so `4711` is never ambiguous.

Response Structure

All API responses follow a consistent envelope format.

Success:

```
{
                                                    "data":                         {

}
```

Error:

```
{
                                    "errors":                 ["Descriptive

}
```

The `count` field reflects the number of records in `data`, `1` for single-record responses, the actual number of returned records for collections. The `meta` field carries contextual information such as the `traceId` of a render run.

Query Parameters

Most `GET` endpoints and the render endpoint share a common set of query parameters that control what is included in the response. They are additive, combine them freely.

Parameter	Type	Default	Description
`limit`	integer	`25`	Maximum records to return. Set to `0` for all.
`withRelations`	boolean	`false`	Include related models (e.g. parameters, fields, resources on a ReportConfig).
`recursiveRelations`	boolean	`false`	Include nested relations within relations.
`withMedia`	boolean	`false`	Include Base64-encoded file content, previews, and thumbnails.

Parameter	Type	Default	Description
`withAudit`	boolean	`false`	Include the audit segment on each record (timestamps, users, token, method).

Example, retrieve a single report config with all relations, media, and audit data:

```
GET /api/v1/report-config/1?withRelations=true&withMedia=true&withAudit=true
```

Audit Segment

When `withAudit=true` is set, every record in the response carries an `audit` block:

```
"audit": {
  "createdAt":
  "updatedAt":
}
```

`creationMethod` and `updateMethod` are derived automatically, `"API"` when a token was used, `"Frontend"` when the action came through the UI. A run of a scheduled job fills these exactly like a direct call does, with the owner of the schedule and the token it runs with.

Render-Specific Request Body

The render endpoint (`POST /api/v1/report-config/{id}/render`) accepts the following fields in the **request body** (JSON).

Field	Type	Required	Description
`outputType`	string		How to return the PDF: `base64`, `url`, `preview`, or `none`

Field	Type	Required	Description
`createHistoryRecord`	boolean	□	Whether to persist a `ReportHistoryRecord` for this render
`createPrintTask`	boolean	□	Whether to create a `ReportPrintTask` after rendering
`printerName`	string	if `createPrintTask`	Name of the target printer
`useExampleValues`	boolean		Use stored example values instead of supplying parameters manually
`parameters`	object		Key-value map of parameter names to values
`resourceOverrides`	object		Per-render override for `P_RESOURCE_*` image parameters, keyed by parameter name. A path has to point at a readable file inside the resources folder, a URL has to be `https://`, or the image is supplied as a Base64 upload. See Rendering with our powerful API for the full reference.
`data`	array		Array of data rows, for reports without a live DB connection
`numberOfCopies`	integer		Number of print copies (default: 1)
`broadcastId`	string		WebSocket channel ID, triggers real-time status updates for the print task
`traceId`	string		Custom trace ID; auto-generated as UUID if omitted. Unique across all history records
`laconicResponse`	boolean		Strip the response to just the PDF output (see below)

Field	Type	Required	Description
`mailing`	object		Send the rendering by mail once it succeeded: mailer, mail template, recipients, attachments and sync or queued dispatch. See Report Mailing for the full reference.

> ⚠ **Output type `none` requires `createPrintTask: true`.** Rendering without any output and without a printer to send it to is rejected, and a `mailing` segment does not take the place of the print task. `preview` and `mailing` are refused together as well, a preview render deletes its own file immediately and has nothing to attach.

Laconic Responses

By default, a render response is fully populated, it includes the echoed input, the PDF output, the linked `ReportConfig`, the created `ReportHistoryRecord`, and the `ReportPrintTask` and `ReportMailTask` if applicable. In high-frequency or bandwidth-sensitive scenarios, add `laconicResponse=true` to strip everything down to just the PDF output.

The Same Body on a Schedule

This body is also what a render job of the Job Scheduler stores. A request you have tested here can be dropped into the `payload` of a schedule unchanged, minus `traceId`, which every run generates for itself. See Render Jobs.

Rate Limiting

The API enforces a configurable rate limit per token. The default is **10 requests per minute**. When the limit is exceeded:

```
{
    "message": "Too many requests! The current rate limiting is 10 requests per minute."
}
```

The limit is adjusted via the `API_RATE_LIMIT_PER_MINUTE` environment variable, see Configuration and Data Models.

Endpoints Overview

Resource	Available operations
User	List, show, create, update, change password, enable/disable, create/revoke tokens
ReportContext	List, show, create, update, delete
ReportConnectionConfig	List, show, create, update, delete, test connection
CommonReportResource	List, show, create, update, delete
ReportConfig	List, show, create, update, delete, update resources / parameters / fields, generate previews, **render**
ReportResource	Link / unlink CommonReportResource
ReportHistoryRecord	List, show, create, update, delete, print, **mail**
ReportPrintTask	List, show, create, update, delete, set printed, set status
ReportMailTask	List, show, create, delete, repeat
Printer	List, show, create, update, delete
Mailer	List, show, create, update, delete, test configuration, send test mail
MailTemplate	List, show, create, update, delete, placeholder catalog, preview
ScheduledJob	List, show, create, update, delete, **run now**, cron preview, placeholder catalog
ScheduledJobRun	List, show, delete

A Practical Example: The Full Render Response

A render call returns a `ReportRendering` object. By default it is fully populated, you see the input you sent, the PDF output, and the linked records that were created.


```
{
      },
    },
    },
                                                                              "repo
                                                                      "reportHistory
  },
  },
}
```


Add `laconicResponse=true` and the response collapses to just `output`, no echoed input, no embedded related records. Ideal for automated pipelines that only need the PDF.

Revision #18
Created 2026-05-10 10:43:08 UTC by Benjamin Fischer
Updated 2026-09-21 20:52:08 UTC by Benjamin Fischer