What is VeloxFactory?

VeloxFactory takes the JasperReports template format, JRXML, and puts a lightweight render engine of its own underneath it. The templates you already know, designed in Jaspersoft Studio as before, run on a slim, modern stack, render fast, and are one HTTP call away from a finished report. Around that engine sits a web application to manage and render them - via a clean frontend, a powerful REST API, or both.

> **Built for the desk, usable on the move.** VeloxFactory's views are optimised for desktop browsers, a resolution of **1600×900** or more gives the most comfortable working experience. The pages work on tablets and phones as well, which is what Scan2Print is built for: a scan station is a mobile device in the warehouse, not a workstation.

Screenshot placeholder: `report-config.index.png`
Report Config overview, the landing page after login, with context badges and thumbnails.

Where VeloxFactory Fits

JasperReports is a mature, battle-tested report engine. It produces high-quality PDFs, supports complex layouts, barcodes, images, and dynamic data - and it has been doing so reliably for decades. Integrating it directly requires a JVM on your server and Java code to drive it - which is a natural fit for Java stacks, but an overhead most PHP, Python, or C# teams prefer to avoid.

VeloxFactory keeps the template format and replaces what runs beneath it. Its render engine is lightweight and runs in pure PHP - no JVM process required on your application server - and its capabilities are exposed via a clean REST API and a management frontend. You design your report templates in Jaspersoft Studio as usual, upload them once, and from that point on, rendering is just a POST request.

```
POST                                                    /api/v1/report-config/A5_KanBan/render
Content-Type:                                                                  application/json
Authorization:                                  Bearer                                 <token>


{
```

```
                    "parameters":          {          "P_ARTICLE_NUMBER":          "456128
}
```

That is the entire integration.

Where It Came From

VeloxFactory was built by someone with a logistics and IT background who needed a simple, reliable way to generate article labels on demand - from whatever system was running at the time, without caring about the underlying report engine. The original idea was modest: a small API wrapper around JasperReports, nothing more.

It grew. A frontend to manage templates. Connection configs for live SQL data. History records for traceability. Print task dispatching, printer master data, a scan station. Mail delivery with its own mailers, templates and dispatch queue. A job scheduler, so the recurring work runs itself. A permission system. What started as "just get me a label PDF" is now a full suite, capable of enterprise integration on demand.

What VeloxFactory Does

At its core, VeloxFactory does six things:

Manages report templates. You upload `.jrxml` files, and VeloxFactory analyses them automatically, detecting parameters, data fields, and image resources. Everything is stored, versioned, and ready to render.

Connects to your data. VeloxFactory can execute SQL queries against live databases at render time. MySQL, MariaDB, PostgreSQL, and SQL Server are all supported. No live connection needed either - you can deliver data directly in the render request as a JSON array, which makes it equally useful for applications that already have the data in memory.

Renders on demand. A single API call produces a PDF. You choose the output format - Base64-encoded inline, a file URL, or silent output for print-only flows. Rendering is synchronous and fast.

Dispatches print jobs. Rendered PDFs can be forwarded directly to a print service via WebSocket, creating a traceable `ReportPrintTask` with status tracking. Labels off the printer, not just on the screen. Printers are maintained as master data and picked by name, and with **Scan2Print** a barcode scan on a phone or a handheld is enough to turn a scanned value into a printed label.

Delivers by mail. A rendering can be sent straight to whoever needs it. A render request carries an optional `mailing` segment, and VeloxFactory sends an HTML mail through a configured mailer, using a configured template, with the PDF attached and, on request, an xlsx export of the same data. Sending is synchronous or queued, every mail is logged as a traceable `ReportMailTask`.

Works on a schedule. Not every document has somebody waiting for it. The shift report that belongs in a mailbox at six in the morning, the weekly stock list, the cleanup that keeps the history from growing without bound: recurring work is configured as a schedule with a crontab expression, maintained in the frontend or through the API like everything else. A scheduled render is a complete render request, printing and mailing included, and every run is logged with its trace id.

Who It Is For

VeloxFactory is for anyone who needs to generate documents - labels, reports, delivery notes, production sheets, certificates - reliably and programmatically. The typical home is in **logistics, production, and warehousing**, where printing article labels, kanban cards, or shipment documents is a daily operational need and downtime is not an option.

More broadly: if your application needs to produce a PDF from structured data, and you do not want to build and maintain a report engine yourself, VeloxFactory is the answer.

Deployment

VeloxFactory is a standard Laravel application. It runs wherever PHP runs, which is nearly everywhere.

Scenario	Works?
Cloud VM or VPS	**Yes**
On-premise server	**Yes**

Scenario	Works?
Docker container	**Yes**
Raspberry Pi on the shop floor	**Yes**
Jaspersoft Server	Not needed

The setup is intentionally slim. PHP 8.2 or newer, a MySQL or MariaDB database for VeloxFactory itself, and Redis for the queue that carries thumbnails, background mails and scheduled jobs, that is the stack. Two small helpers complete it: `poppler-utils` draws the report thumbnails, and Supervisor keeps the background processes running. The render engine and every other library come with the application and are installed by Composer. No application server, no JVM process to manage, no separate Jaspersoft infrastructure. The complete list is on Installing VeloxFactory.

> ℹ **Want to try it first?** A live demo is available at demo.veloxfactory.dev.
>
> **Login:** `demo@veloxfactory.dev` · **Password:** `demo`

Revision #29
Created 2026-05-10 10:34:50 UTC by Benjamin Fischer
Updated 2026-10-02 05:07:35 UTC by Benjamin Fischer