

The Print Service:

VeloxFactory on your

printers

VeloxFactory does not talk to printers directly. A companion Windows application, the **VeloxFactory Print Service**, runs on any machine that has the target printers installed. It receives print tasks from VeloxFactory, prints the PDF on the right printer and reports the result back. Both sides communicate only over the VeloxFactory REST API and the WebSocket server (Laravel Reverb). There is no shared database and no shared file system.

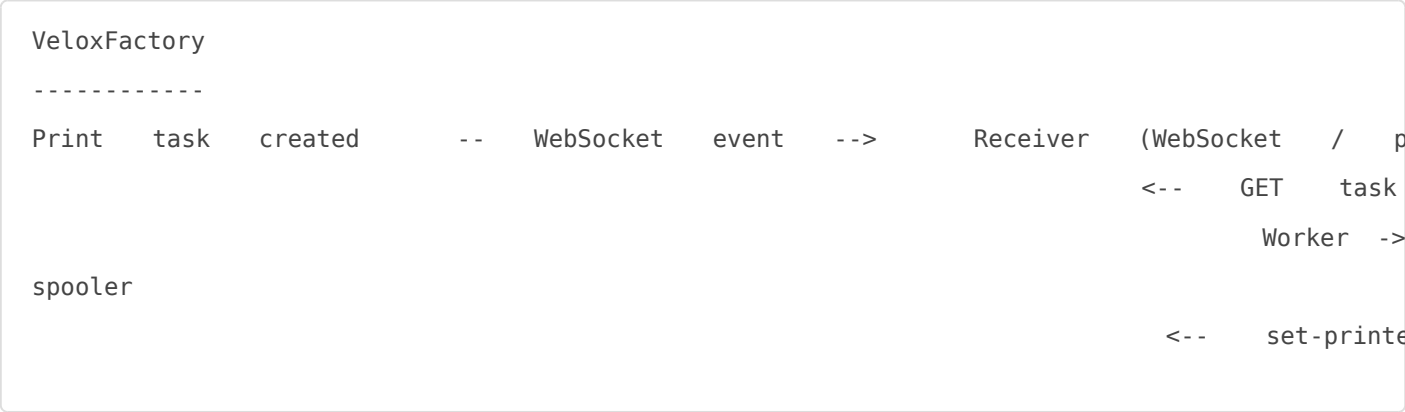
This chapter describes version **2.0**.

At a glance

One program, two ways to run	Started without parameters it opens a window with dashboard, live console, job history, printer overview, statistics and settings. Started with <code>--headless</code> the same program runs without any window from the Windows Task Scheduler, whether a user is logged on or not.
Reliable receiving	WebSocket events for speed, a safety poll against lost events, automatic polling while the WebSocket is down. Only tasks of the own broadcast ids are taken.
No double prints	A local job journal remembers every task. A task that reached the Windows spooler is never printed a second time, even after a crash or a lost confirmation.
Sharp output	PDFium draws every page directly onto the printer in its native resolution. Text, lines and barcodes stay vector graphics, nothing is rasterised in between.
Rate limit aware	Reads the rate limit headers of the API. On HTTP 429 every API call pauses until <code>Retry-After</code> , print tasks wait instead of failing.
Nothing on disk	PDFs stay in memory. The API token is stored encrypted (DPAPI) and separate from the configuration.

Operable	Rolling log files, Windows event log with fixed event ids, optional local health endpoint, diagnostic package, graceful stop for updates.
-----------------	---

How it works



1. **Receive.** A new print task triggers the WebSocket event `report-print-task.created` on the private channel `private-report-print-tasks`. The event only carries the task id and the broadcast id. In addition, the service polls the pending tasks of its broadcast ids (details in [Receiving and processing print tasks](#)).
2. **Fetch.** The service loads the task including the PDF (`withMedia=true`). The printer machine therefore needs HTTP access to VeloxFactory, not only WebSocket access.
3. **Deduplicate.** The job journal decides whether the task is new. Known tasks are skipped.
4. **Print.** A worker takes the task, waits for the printer if another job is running on it, checks the Windows queue (offline, paper out) and prints. Transient problems are retried.
5. **Confirm.** Success is written to the outbox first and then sent with `PATCH /api/v1/report-print-task/{id}/set-printed`. If VeloxFactory is unreachable, the outbox retries until it succeeds. Permanent failures are reported with `set-status` and a categorised error message.

Operating modes

Mode	Started by	Behaviour
Headless	<code>--headless</code> , normally from the scheduled task	The engine runs without any window. This is the production mode.

Mode	Started by	Behaviour
Monitor	Program started without parameters while a headless engine runs	The window shows state, console, jobs and statistics of the headless engine and edits its configuration. It does not print itself.
Interactive	Program started without parameters, no engine running	The engine runs inside the window. Closing the window keeps it running in the notification area. Useful for setup, tests and machines without scheduled task.

A named mutex per instance makes sure only one engine runs. The window detects a running headless engine automatically and switches to monitor mode.

i Several instances per machine. With `--config <path>` every instance gets its own configuration, token, journal, logs and scheduled task. Use one instance per broadcast id or site. Two instances listening to the same broadcast id would both print the same task, the journal only protects within one instance.

In this chapter

Installation and updates	Requirements, folder layout, service account, scheduled task, update and rollback
Configuration reference	Every key of <code>config.json</code> with default and range, token, hot reload, import and export
Receiving and processing print tasks	Receive modes, broadcast ids, rate limit, pipeline, retries, pause, journal, outbox, dry run
Printers and printing	Printer comparison, alias, per-printer settings, dialog printers, test page
The Print Service window	Dashboard, console, jobs, printers, statistics, settings, info, tray icon
Command line reference	All parameters, exit codes and examples
Monitoring and troubleshooting	Log files, event ids, health endpoint, diagnostic package, common problems
Open Source Attribution	Bundled packages and their licenses