

VeloxFactory Print Service

Our Windows print service: installation, configuration, printing, the window, the command line and monitoring.

- [The Print Service: VeloxFactory on your printers](#)
- [Installation and updates](#)
- [Configuration reference](#)
- [Receiving and processing print tasks](#)
- [Printers and printing](#)
- [The Print Service window](#)
- [Command line reference](#)
- [Monitoring and troubleshooting](#)

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.

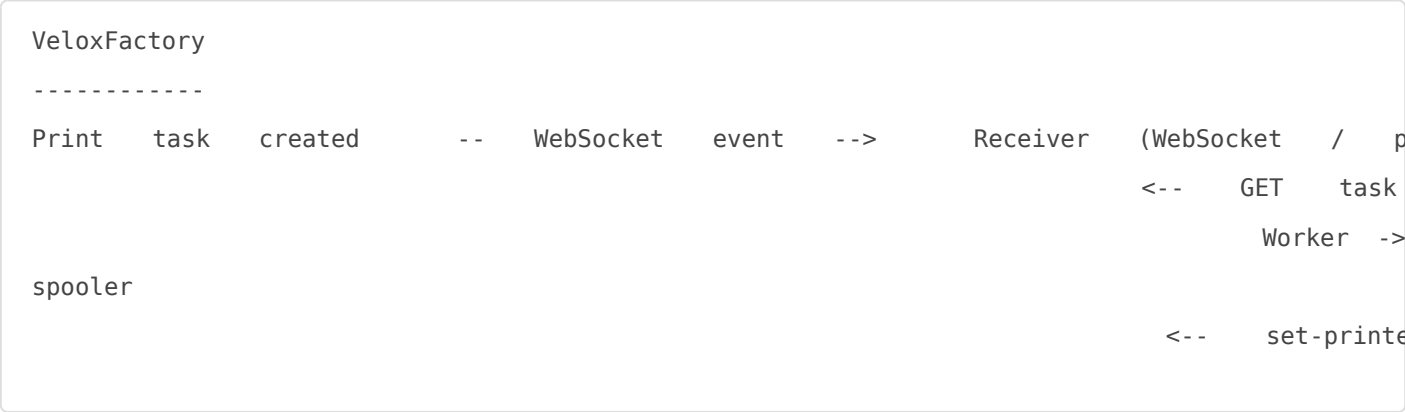
This chapter describes version **2.0**.

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

Installation and updates

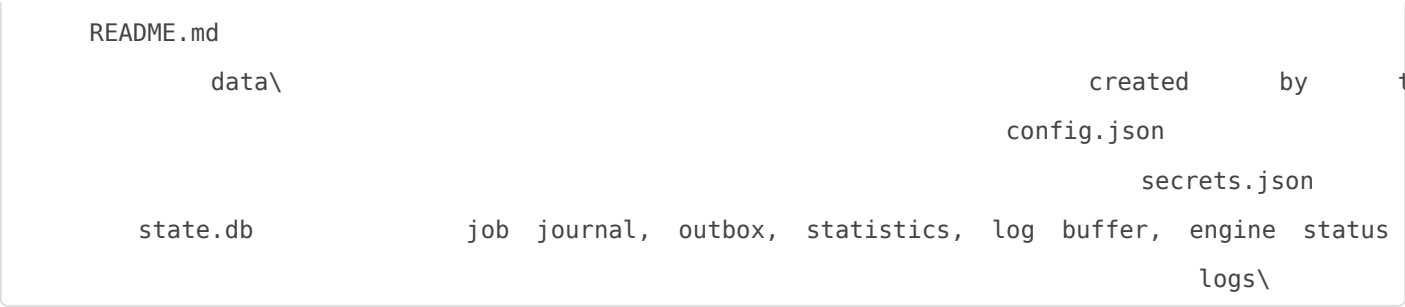
The Print Service has no installer. The program is delivered as a git repository that contains only the released build. Updates are a `git` checkout, driven by a script that stops the service gracefully first. Configuration, token, job journal and logs live in a separate data folder and survive every update.

Requirements

Operating system	Windows 10/11 or Windows Server 2019 or newer, x64
Runtime	.NET 10 Desktop Runtime (x64) . The build is framework-dependent, the runtime is not bundled.
Git	Git for Windows and read access to the release repository
Network	HTTPS to the VeloxFactory API and WSS to the Reverb server of your instance. With receive mode <code>Polling</code> the API alone is enough.
API token	Token of a user with <code>global:use-api</code> , <code>report-print-task:read</code> and <code>report-print-task:update</code> . <code>printer:read</code> is optional and enables the printer comparison. <code>:read</code> is also the right the private channel checks, a token without it never receives a WebSocket event.
Service account	A dedicated Windows account for the scheduled task, see below
Printers	Installed machine-wide, see Printers and printing

Folder layout

```
C:\kiwisoftware\VeloxPrint\
  app\                clone of the release repository, program only, updated by
                      kiwisoftware.VeloxFactory.PrintService
  bin\
  update.ps1
```



Where the data folder is:

- 1. `--config <path>` sets it explicitly: all files of the instance live next to that configuration file.
- 2. If the program runs from a folder named `app` or below it, the data folder is `data` next to that `app` folder.
- 3. Otherwise it is `%ProgramData%\kiwisoftware\VeloxPrint\`.

File	Permissions
<code>config.json</code>	Administrators and service account: modify. Users: read.
<code>secrets.json</code>	Administrators and service account: modify. Users: no access.
<code>state.db</code> , <code>logs\</code>	Service account: modify. Users: read.

The first start with administrator rights creates the data folder and sets these permissions. Setting up the scheduled task adds the service account.

 **Do not copy `secrets.json` to another machine.** The token is encrypted with DPAPI for this machine (scope LocalMachine), so the window of an administrator and the headless service account can both read it. On another machine it cannot be decrypted, enter the token there again.

First installation

1. Create the service account

The scheduled task runs under its own account. It must exist before the task is set up, the Print Service does not create it. PowerShell as administrator:

```
$pw = Read-Host -AsSecureString "Password"
New-LocalUser -Name "svc-veloxprint" -Password $pw -FullName "VeloxFactory Print Service" `
    -Description "Service account VeloxFactory Print Service" -PasswordNeverExpires
UserMayNotChangePassword
```

1. Do not add the account to the Administrators group, the standard Users group is enough.
2. Use `-PasswordNeverExpires`. An expired password stops the task until it is set up again.
3. For printers on a print server in a domain, use a domain account (`DOMAIN\svc-veloxprint`) instead. Only a domain account can authenticate at the print server.
4. Do not use `SYSTEM`. It accesses print servers with the computer account, shares often refuse that.

2. Clone the release

PowerShell as administrator:

```
New-Item -ItemType Directory C:\kiwisoftware\VeloxPrint
git clone https://git.fischernetz.xyz/bfischer/veloxfactory-printservice-release.git
C:\kiwisoftware\VeloxPrint\app
```

3. Configure

1. Start `C:\kiwisoftware\VeloxPrint\app\bin\kiwisoftware.VeloxFactory.PrintService.App.exe` **as administrator**. The first elevated start creates `data\` and restricts its permissions.
2. **Settings → Connection:** base URL of VeloxFactory and API token. **Test connection** checks reachability, token, channel authorization, WebSocket handshake and rate limit headers.
3. **Settings → WebSocket (Reverb):** host, port, scheme and app key of your Reverb server.
4. **Settings → Receiving:** the broadcast ids of this site.
5. Save. See the [Configuration reference](#) for every setting.

4. Set up the scheduled task

Settings → Operation → Set up scheduled task, enter the service account (`.\svc-veloxprint` for a local account) and its password. Alternatively on the command line (as administrator):

```
& "C:\kiwisoftware\VeloxPrint\app\bin\kiwisoftware.VeloxFactory.PrintService.App.exe" --
install-task --user .\svc-veloxprint
```

The setup:

1. grants the account the right *Log on as a batch job*,
2. sets the permissions of the data folder and `secrets.json` for the account,
3. registers the Windows event log source *VeloxFactory Print Service*,
4. creates or updates the task with the settings below.

Task setting	Value	Reason
Name	<code>VeloxFactory Print Service</code> , for an alternative configuration <code>VeloxFactory Print Service (<config name>)</code>	Root folder of the Task Scheduler library
Run	Whether the user is logged on or not, password stored	No desktop needed; without stored password network printers are unreachable
Trigger	At system start, delayed 1 minute	Network and spooler are ready
Action	<code>--headless</code> (plus <code>--config</code> for an alternative instance), start in the program folder	The Task Scheduler would otherwise start in <code>System32</code>
Time limit	None	The Windows default of 3 days would end the engine
On failure	Restart every minute, up to 999 times	Crash recovery
Already running	Do not start a new instance	In addition to the instance mutex
Privileges	Not elevated	Not needed
Battery	Runs on battery, starts when available	Missed starts are caught up

⚠ **Do not edit the task by hand.** Setting it up again from the window or with `--install-task` overwrites it with the values above. After changing the password of the service account, set the task up again, the password is stored in the task.

The window shows the task under **Settings → Operation**: state, account, last and next run, last result and the XML definition. It can start and remove the task there, too.

5. Verify

1. **Printers:** print a test page on every printer the site uses.
2. Start the task (**Settings → Operation → Start** or reboot). The window switches to monitor mode.
3. Create a print task in VeloxFactory for one of the broadcast ids and watch it on the **Dashboard**.

Updates and rollback

PowerShell as administrator:

```
#                                newest                                release
powershell -ExecutionPolicy Bypass -File C:\kiwisoftware\VeloxPrint\app\update.ps1

# a specific release, also the way back to an older one
powershell -ExecutionPolicy Bypass -File C:\kiwisoftware\VeloxPrint\app\update.ps1 -Version 2.0.1
```

Step	What <code>update.ps1</code> does
1. Check	Refuses to run without administrator rights or if <code>app\</code> contains local changes
2. Fetch	<code>git fetch</code> before anything is stopped. If the server is unreachable, the service keeps running. Already on the target: nothing to do.
3. Stop	Calls <code>--stop</code> for every scheduled task that runs from this folder (with its <code>--config</code>). Running print jobs finish, the outbox is flushed. Waits up to 90 s until no process runs from <code>app\bin</code> .
4. Switch	Checks out the newest release (<code>origin/main</code>) or the tag <code>v<Version></code>
5. Start	Starts the scheduled tasks again that were running before

- 1. Releases are git tags (`v2.0.1`). `git -C C:\kiwisoftware\VeloxPrint\app log --oneline` shows the source commit of every release.
- 2. New settings are added to `config.json` with their defaults on the next start.
- 3. Never change files inside `app\`. The script refuses to update a modified folder. Settings belong into `data\`.

Several instances on one machine

Each instance needs its own configuration file. Its name becomes the prefix of all files of the instance, so several instances can share one data folder:

```
data\config.json
data\lager-nord.json
data\lager-nord.secrets.json
data\lager-nord.state.db
data\logs\lager-nord\
```

```
$exe = "C:\kiwisoftware\VeloxPrint\app\bin\kiwisoftware.VeloxFactory.PrintService.App.exe"
& $exe --config C:\kiwisoftware\VeloxPrint\data\lager-nord.json
& $exe --config C:\kiwisoftware\VeloxPrint\data\lager-nord.json --install-task --user .\svc-veloxprint
```

The task of that instance is called `VeloxFactory Print Service (lager-nord)`. Give every instance its own broadcast ids.

Configuration reference

All settings of an instance live in `config.json` in the data folder (see [Installation and updates](#)). The API token is not part of it, it is stored encrypted in `secrets.json`. Normally you edit everything in the window under **Settings**. The file can also be edited in a text editor, the running engine picks up the change by itself.

How the configuration is handled

Hot reload	The engine watches the file and applies changes without restart. A new URL, token or WebSocket setting triggers a reconnect, running print jobs finish. Exception: of the <code>logging</code> settings only <code>minimumLevel</code> applies immediately; retention, file size, event log and JSON file take effect after a restart.
Validation	On load and before saving. The window shows errors per field. A headless start with an invalid file ends with exit code 1. An invalid change while the engine runs is not applied, the engine keeps the last valid configuration and writes event id 5000.
New keys	After an update the service adds keys of the new version with their defaults and writes the file (if it may). Unknown keys are reported in the log and removed on the next save.
Change log	Every save is recorded with Windows account, time and changed keys in <code>state.db</code> . The diagnostic package contains the last 500 changes.
Export and import	Settings → Import and export. The export never contains the token. The import keeps name and id of the receiving instance, check and save afterwards. Useful to roll out one configuration to several sites.
Write access	Saving needs write access to the data folder. Without it the window offers Restart as administrator .

Example

```
{

  "instance": { "id": "5d0c2f4e-...", "name": "Warehouse North", "serviceAccount":
"SRV01\\svc-veloxprint"
},
  "api": { "baseUrl": "https://vf.example.com", "timeoutSeconds": 30, "allowUnencrypted":
false,
          "serverTimeZone": "Europe/Berlin"
},

},
  "websocket": { "scheme": "wss", "host": "vf.example.com", "port": 443, "appKey": "your-
reverb-app-key",
                "channel": "private-report-print-tasks"
},
                "paused": false, "pauseReason": null, "
                "retryBackoffSeconds": 30
},
  "printerDefaults": { "scaling": "ActualSize", "autoRotate": true, "dpi": null, "copiesMode":
"Driver", "paperSource": null, "duplex": "Default", "color": "Default" },

  [ { "queueName": "WH-LABEL-01", "localName": "\\printsrv01\\WH-LABEL-01", "dpi": 300 },
    { "queueName": "PDF24", "copies": 1 }
],
  "logging": { "minimumLevel": "Information", "retentionDays": 30, "maxFileSizeMb": 250,
"eventLog": true, "jsonFile": false
},
    "statistics": { "retentionDays": 365, "resetAt": null
    "health": { "enabled": false, "port": 8080
    "ui": { "errorNotifications": true, "windowWidth": null, "windowHeight": null
```

```
"windowMaximized": false
}
```

instance

Key	Default	Description
<code>id</code>	new GUID	Identifies the instance (mutex, stop signal). Never copy it to another instance, an import keeps the local id.
<code>name</code>	machine name	Display name in the window, on the test page and in the diagnostic package
<code>serviceAccount</code>	null	Account of the scheduled task, written when the task is set up. Used for the file permissions.

api

Key	Default	Description
<code>baseUrl</code>	empty, required	Base URL of VeloxFactory without <code>/api/v1</code> , e.g. <code>https://vf.example.com</code> . All endpoints are derived from it.
<code>timeoutSeconds</code>	30 (5 to 300)	Timeout of a single API call
<code>allowUnencrypted</code>	false	Allows <code>http://</code> and <code>ws://</code> . Without it both are rejected, because token and PDFs would travel unencrypted. The window shows a warning banner while it is on.
<code>serverTimeZone</code>	Europe/Berlin	Time zone of the VeloxFactory instance (<code>APP_TIMEZONE</code>). Audit timestamps of the API carry no offset, the lead time statistics need it.

The **API token** is entered in **Settings** → **Connection** and stored in `secrets.json`. Required rights: `global:use-api`, `report-print-task:read`, `report-print-task:update`, optional `printer:read`. The token never appears in logs, exports or diagnostic packages.

receive

Key	Default	Description
<code>mode</code>	Hybrid	<code>Hybrid</code> , <code>WebSocket</code> or <code>Polling</code> , see Receiving and processing print tasks
<code>broadcastIds</code>	<code>[]</code>	Broadcast ids this instance prints, exact match. At most 20, unique, no commas, no surrounding spaces, max. 255 characters. At least one is required unless <code>includeWithoutBroadcastId</code> is on.
<code>includeWithoutBroadcastId</code>	false	Also take tasks and printers without broadcast id. VeloxFactory sends no WebSocket event for such tasks, they arrive only by polling (in Hybrid after at most <code>safetyPollIntervalSeconds</code>). Turn it on in one instance at most.
<code>pollingIntervalSeconds</code>	30 (min. 1)	Poll interval in mode <code>Polling</code> , and in <code>Hybrid</code> while the WebSocket is down. Below 3 s the window warns: constant API calls, the rate limit is used up quickly.
<code>safetyPollIntervalSeconds</code>	120 (min. 30)	Safety poll in <code>Hybrid</code> while the WebSocket is subscribed
<code>pageSize</code>	10 (1 to 100)	Tasks per poll page. Every page carries the PDFs, a small value keeps responses small. Further pages follow immediately.

websocket

Ignored in mode `Polling`. The URL is built as `{scheme}://{host}:{port}/app/{appKey}?protocol=7&client=veloxprint&version={x}`.

Key	Default	Description
<code>scheme</code>	wss	<code>wss</code> or <code>ws</code> (<code>ws</code> only with <code>api.allowUnencrypted</code>)
<code>host</code>	empty, required	Host of the Reverb server as reachable from the printer machine (<code>REVERB_HOST</code>)
<code>port</code>	443	Port of the Reverb server (<code>REVERB_PORT</code> , behind a reverse proxy usually 443)
<code>appKey</code>	jasper-who	<code>REVERB_APP_KEY</code> of your VeloxFactory instance
<code>channel</code>	private-report-print-tasks	Private channel of the print task events. Must start with <code>private-</code> . Keep the default.

processing

Key	Default	Description
<code>paused</code> , <code>pauseReason</code> , <code>pausedUntil</code>	false, null, null	Maintenance pause. With <code>pausedUntil</code> (ISO 8601 with offset) the pause ends by itself. Survives a restart.
<code>maxParallelJobs</code>	4 (1 to 32)	Tasks processed at the same time
<code>serializePerPrinter</code>	true	One job per printer at a time, in the order of the task ids
<code>retryCount</code>	3 (0 to 10)	Local retries on transient errors (printer offline, paper out, spooler refused the job)
<code>retryBackoffSeconds</code>	[5, 30, 120]	Wait before retry 1, 2, 3 and so on (1 to 3600 each). With more retries than values the last value repeats.
<code>printTimeoutSeconds</code>	120 (10 to 3600)	Limit for a print call and for the spooler wait of <code>Completed</code> . A driver that waits for a dialog ends here.
<code>confirmation</code>	Spooled	<code>Spooled</code> : confirm after handing the job to the Windows spooler. <code>Completed</code> : wait until the spooler job has finished. Can be overridden per printer.

Key	Default	Description
<code>dryRunFolder</code>	null	Absolute local or UNC path. When set, PDFs are written there instead of printed and not confirmed. Empty = print normally.

printerDefaults and printers

`printerDefaults` applies to every printer. An entry in `printers` overrides single values for one queue name; a value missing there (or `null`) falls back to the default. Details and advice in [Printers and printing](#).

Key	Default	Description
<code>queueName</code>	only in <code>printers</code> , required	Printer name as the print task carries it (<code>printerName</code> of the VeloxFactory printer). Unique, case-insensitive.
<code>localName</code>	only in <code>printers</code>	Alias: Windows printer that receives the jobs of this queue name. Empty = same name.
<code>scaling</code>	ActualSize	<code>ActualSize</code> (1:1 from the paper corner) or <code>FitToPage</code> (scaled into the printable area, centred)
<code>autoRotate</code>	true	Rotates a page when its orientation differs from the paper
<code>dpi</code>	null (72 to 1200)	Output resolution of the driver. Takes effect only if the driver offers exactly this resolution, otherwise the driver default stays. Empty = driver default.
<code>copiesMode</code>	Driver	<code>Driver</code> : one spooler job with n copies. <code>Loop</code> : one job per copy, for drivers that ignore the copy count (e.g. PDF24, Microsoft Print to PDF).
<code>paperSource</code>	null	Tray by name as the driver lists it. Unknown names keep the default tray.
<code>duplex</code>	Default	<code>Default</code> , <code>Simplex</code> , <code>Vertical</code> , <code>Horizontal</code>
<code>color</code>	Default	<code>Default</code> , <code>Color</code> , <code>Monochrome</code> (only on printers that support colour)

Key	Default	Description
<code>confirmation</code>	only in <code>printers</code>	Overrides <code>processing.confirmation</code> for this printer

logging, statistics, health, ui

Key	Default	Description
<code>logging.minimumLevel</code>	Information	<code>Verbose</code> , <code>Debug</code> , <code>Information</code> , <code>Warning</code> , <code>Error</code> , <code>Fatal</code> . <code>Debug</code> shows WebSocket protocol details (ping, pong, auth). Applies immediately.
<code>logging.retentionDays</code>	30 (1 to 3650)	Log files older than this are deleted. Takes effect after a restart.
<code>logging.maxFileSizeMb</code>	250 (1 to 10240)	Size limit per log file; a day beyond it continues in <code>_001</code> , <code>_002</code> . Takes effect after a restart.
<code>logging.eventLog</code>	true	Warnings and errors to the Windows event log (needs the event source, created with the scheduled task). Takes effect after a restart.
<code>logging.jsonFile</code>	false	Additional log in compact JSON (<code>veloxprint-*.json</code>) for log collectors. Takes effect after a restart.
<code>statistics.retentionDays</code>	365 (30 to 3650)	How long statistics are kept
<code>statistics.resetAt</code>	null	Set by Reset statistics . Jobs before this time no longer count.
<code>health.enabled</code> , <code>health.port</code>	false, 8765 (1024 to 65535)	Local health endpoint <code>http://localhost:{port}/health</code> , see Monitoring and troubleshooting
<code>ui.errorNotifications</code>	true	Windows notification of the tray icon on print errors
<code>ui.windowWidth</code> , <code>windowHeight</code> , <code>windowMaximized</code>	null, null, false	Last window size, written by the window itself

Receiving and processing print tasks

This page follows a print task through the Print Service: how it arrives, which instance takes it, how it is printed, retried and confirmed, and what protects it against being printed twice.

Receive modes

Mode	Behaviour	Use it when
Hybrid (default)	WebSocket events, plus a safety poll every <code>safetyPollIntervalSeconds</code> (120 s). While the WebSocket is down it polls every <code>pollingIntervalSeconds</code> (30 s) until the reconnect succeeds.	Always, unless there is a reason for another mode
WebSocket	Only events. Polls once at start and after every successful (re)subscribe, never periodically.	Minimal API load
Polling	Polls every <code>pollingIntervalSeconds</code> , no WebSocket	Reverb is not reachable from the printer machine (firewall, proxy)

Why the safety poll

Reverb delivers an event once and does not keep it. An event sent during a short network interruption is lost, and a connection can look alive without delivering events. The safety poll finds such tasks at the latest after its interval. Tasks the journal already knows are skipped, so polling never prints twice.

The poll request

```
GET /api/v1/report-print-task?status=pending&perPage=10&page=1&withMedia=true
&withRelations=false&withAudit=true&broadcastId=Standard,Warehouse&withoutBroadcastId=false
```

1. The filter runs on the server: only tasks of the own broadcast ids are returned.
2. `withMedia=true` delivers the PDFs directly, no second request per task.
3. `withAudit=true` delivers the creation time for the lead time statistics.
4. If more pages exist, the next page is requested immediately instead of after the interval.

The WebSocket connection

1. Pusher protocol 7 against Laravel Reverb, private channel `private-report-print-tasks`, authorized with `POST /api/v1/broadcasting/auth` and the API token.
 2. The event `report-print-task.created` carries `{ model, id, broadcastId }`. If the broadcast id matches, the service loads the task with `GET /api/v1/report-print-task/{id}?withMedia=true`.
 3. Keep-alive: every `pusher:ping` of the server is answered with `pusher:pong`. After 25 s without traffic the service sends its own ping and drops the connection if no pong arrives within 30 s.
 4. Reconnect with exponential backoff from 1 s to 60 s, $\pm 20\%$ jitter. After every successful subscribe a catch-up poll fetches what was created in the meantime.
 5. A first connection failure is logged as warning. An outage longer than 5 minutes writes event id 3000.
-

Broadcast ids: which instance prints what

Every print task and every printer in VeloxFactory can carry a broadcast id, usually one per site or printer group. An instance only takes tasks whose broadcast id is in its `receive.broadcastIds`, both from events and from polls.

Tasks without broadcast id are taken only with `includeWithoutBroadcastId`. VeloxFactory sends no WebSocket event for them, so they arrive only by polling: in Hybrid within `safetyPollIntervalSeconds`, in mode WebSocket only at start and after a reconnect. Give print tasks a broadcast id whenever they should print immediately.

⚠ **One instance per broadcast id.** The job journal protects against double printing within one instance only. Two instances listening to the same broadcast id, or two instances with `includeWithoutBroadcastId`, print the same task twice.

Rate limit

The VeloxFactory API limits requests per IP address, by default 10 per minute (`API_RATE_LIMIT_PER_MINUTE`). The service reads the limit from `X-RateLimit-Limit` and the rest from `X-RateLimit-Remaining` in every response and shows both on the dashboard.

Action	API calls
Task received by WebSocket	2 (GET task, set-printed)
Task received by poll	1 (set-printed) plus its share of the poll
Poll	1 per page
Error report	1 (set-status)
WebSocket login	1 (broadcasting/auth)

1. On HTTP 429 every API call of the process waits until `Retry-After` (60 s if the header is missing), then the same request is sent again. Print tasks wait, they do not fail. The API state shows *Rate limited*, event id 2002 is written.
2. The service does not slow down in advance. A low remaining count is only displayed.
3. **Settings → Receiving** shows the base load of the chosen mode in calls per minute (Hybrid with 120 s: 0.5) and warns when it exceeds half of the limit.
4. The limit counts per IP. All print services, scripts and MCP clients behind the same NAT share it. For print environments raise `API_RATE_LIMIT_PER_MINUTE` in the `.env` of your VeloxFactory instance.

Processing

```
Receiver -> queue -> journal check -> worker (max. maxParallelJobs) -> printer lock -> queue  
check -> print -> outbox
```

1. **Journal check.** A task the journal knows in state *Printing* or later is skipped. A task that was confirmed or reported before and is pending again (reprint) is loaded fresh; only if VeloxFactory really shows it as pending is it printed again.
2. **Fetch.** Tasks from a WebSocket event are loaded now, tasks from a poll already carry their PDF. A task that no longer exists or is no longer pending is dropped.
3. **Printer.** The queue name of the task is resolved to the local printer (alias, see [Printers and printing](#)). The Windows default printer is never used; a task without printer name fails with `PrinterNotFound`.
4. **Printer lock.** With `serializePerPrinter` only one job runs per printer, ordered by task id. Jobs for different printers run in parallel up to `maxParallelJobs`.
5. **Queue check.** Before printing, the Windows queue is checked. *Not installed* fails immediately. *Offline*, *paper problem* or *error* goes into retry without printing.

6. **Print.** See [Printers and printing](#). The spooler document name is `VF #{id} {file name}`, so the job can be found in the Windows print queue.
7. **Confirm.** With `confirmation = Spooled` the task is confirmed as soon as the spooler accepted it. With `Completed` the service waits until the spooler job has finished (at most `printTimeoutSeconds`).

Retries

Retried (transient)	Fails immediately (permanent)
Printer reports offline, paper problem or error (<code>PrinterOffline</code>)	Printer not installed for the service account (<code>PrinterNotFound</code>)
Spooler refused to accept the job (<code>SpoolerError</code>)	PDF missing, unreadable or not renderable (<code>RenderFailed</code>)
	Print call or spooler job ran into the timeout, spooler job ended with an error (<code>Timeout</code> , <code>SpoolerError</code>)

With the defaults (`retryCount` 3, `retryBackoffSeconds` [5, 30, 120]) a task waits about 2.5 minutes before it is reported as failed. For label printers that are offline longer, for example during a roll change, use more retries or a longer last value, e.g. [5, 30, 120, 300, 600] with `retryCount` 5.

i Once handed to the spooler, a job is never retried. A failure after that point (timeout, spooler error) is only reported. Retrying could print the document twice.

Error reports

Permanent failures are reported with `PATCH /api/v1/report-print-task/{id}/set-status`, status `error` and an `errorMessage` of the form `[Category] message`, shortened to 1,000 characters. Categories: `PrinterNotFound`, `PrinterOffline`, `RenderFailed`, `Timeout`, `SpoolerError`, `Unknown`. The task shows the message in VeloxFactory.

Job journal and outbox

VeloxFactory has no endpoint to claim a task. The local journal in `state.db` takes that role and records every task with its state:

State	Shown as	Meaning
<code>Received</code>	Received	Taken, waiting or in retry
<code>Printing</code>	Printing	Print call running

State	Shown as	Meaning
Spooled	Spooled, not confirmed	In the spooler, confirmation pending. Never printed again.
Confirmed	Printed	VeloxFactory shows the task as printed
Failed	Failed	Failed, error report pending
ErrorReported	Error reported	VeloxFactory shows the task as error
Unknown	Unknown	The service ended while printing. Not printed again automatically, reported as error with <i>Print state unknown after service restart, check printer and reset task if needed.</i>
DryRun	Dry run	Written to the dry-run folder, not confirmed

Outbox. Confirmations and error reports are written to the outbox first and then sent. If the API is unreachable, the outbox retries with backoff (5 s, doubling, at most every 5 minutes) until it succeeds. If VeloxFactory answers 404 (task deleted) or 422 (rejected), the entry is dropped. After a crash, spooled tasks without outbox entry get their confirmation again. The dashboard tile *Queue* shows open outbox entries as *to confirm*.

Reprint. To print a task again, use **Jobs → Print again** in the window. It resets the task to pending in VeloxFactory (`PATCH set-printed` with `false`), the next poll prints it. Resetting the task in VeloxFactory itself has the same effect.

Pause processing

A pause stops printing without stopping the service. Use it for printer maintenance, a roll change or while wrong tasks are cleaned up in VeloxFactory.

1. Set it on the dashboard (**Pause / Resume**) or under **Settings → Processing** with reason and optional end time. The pause is stored in `config.json`, so it survives a restart and a pause set in the window also pauses the headless engine.
2. While paused: the WebSocket stays connected, events are noted but not processed, periodic polls are skipped. Tasks already in the pipeline wait before their next print attempt; the waiting time does not count as a retry. A print call already in progress is not interrupted. The outbox keeps delivering confirmations.
3. New tasks stay pending in VeloxFactory.
4. On resume the service polls immediately and prints everything that arrived meanwhile.
5. The engine state is *Paused* with reason and end time. The health endpoint still answers 200.

Dry run

For tests and demonstrations without paper. With `processing.dryRunFolder` set (or `--dry-run <folder>` on the command line, which applies to that run only) every task is received, fetched and resolved as usual, but instead of printing the PDF is written to `<folder>\<task id>_<file name>`. Nothing is confirmed, the task stays pending in VeloxFactory.

⚠ **Dry-run tasks are not printed later.** The journal marks them as *Dry run*, and this instance skips them after the dry run is switched off, although they are still pending in VeloxFactory. Use a test broadcast id for dry runs, or create new print tasks afterwards.

Printers and printing

A print task names its printer by the `printerName` of the VeloxFactory printer master data (see [Printers: master data for your print targets](#)). The Print Service calls this the **queue name** and maps it to a Windows printer on its machine. This page covers that mapping, the per-printer settings, how pages are printed and which printers are not suitable.

Installing printers for the service

The headless service runs under its own account without a desktop. What that account cannot see, it cannot print on.

Installation	Shown as	Suitable
Local printer (USB, TCP/IP port)	Local	Yes
Print server connection installed for all users	Machine-wide	Yes
Print server connection added by a user (<code>\\server\printer</code>)	User profile only	No. Only that user sees it, the service account does not.

Install a print server connection machine-wide (as administrator, takes effect after the spooler restarts or the next logon):

<code>rundll32</code>	<code>printui.dll,PrintUIEntry</code>	<code>/ga</code>	<code>/n\\printsrv01\WH-LABEL-01</code>
<code>Restart-Service</code>			Spooler

i The default printer is never used. The default printer is a per-user setting. The service always prints on the printer named in the task (or its alias). A task without printer name fails with `PrinterNotFound`.

The printer list

The **Printers** page of the window compares four sources:

- 1. the VeloxFactory printer master data (GET /api/v1/printer, needs printer:read), filtered on the own broadcast ids. Printers without broadcast id only appear with includeWithoutBroadcastId,
- 2. the printers installed on this machine,
- 3. printers with own settings in config.json,
- 4. printers that received jobs in the last 7 days but are in none of the sources above (Only in recent print jobs).

Local printers that no row points to are listed at the end as **Not in VeloxFactory**, useful as alias targets and for test pages.

Column	Content
Printer	Display name from VeloxFactory and queue name; an alias shows as → local name
Type, Location	From VeloxFactory
Local	Local, Machine-wide, User profile only, Installed (scope unknown) or Missing
Status	Windows queue: Ready, Offline, Paper problem, Error, Paused, Unknown, plus the number of jobs in the queue
Hint	Not installed on this machine, Only connected for this user profile, Prints with a dialog, not suitable headless, Inactive in VeloxFactory, tasks are printed nevertheless

Actions per row: copy the printer name, print a test page, edit the printer settings, open the Windows print queue.

⚠ **A printer shows as "Not in VeloxFactory" although it exists there?** Its broadcast id is not one of this instance's broadcast ids, or it has none and includeWithoutBroadcastId is off. Set the broadcast id of the printer in VeloxFactory, or turn on *Also take tasks without broadcast id* under **Settings → Receiving**, which also takes all print tasks without broadcast id.

Alias: local printer name

The alias (localName) sends the jobs of a queue name to a Windows printer with a different name. VeloxFactory does not need to know about it.

```
"printers":
[
  {
    "queueName": "WH-LABEL-01",
    "localName": "\\\\"printsrv01\\WH-LABEL-01"
  },
  {
    "queueName": "OFFICE-A4",
    "localName": "Brother HL-L3210CW series Printe
  ]
```

1. Without alias the Windows printer must have exactly the queue name (case-insensitive).
2. Typical uses: the same printer has a different name on every workstation, a temporary replacement printer, or a PDF printer for tests.
3. The alias is set in the printer editor, field *Local name*. The same name as the queue name, or empty, stores no alias.
4. The alias lives in the local `config.json`. Every machine needs its own entry.

Printer settings

`printerDefaults` in **Settings** → **Printer defaults** applies to all printers. The printer editor on the **Printers** page overrides single values for one printer (*Use default* keeps the default). The effective value is: printer override, else default, else the driver's setting.

Setting	Effect
Scaling	<i>Actual size</i> : 1:1, placed at the physical paper corner (labels, forms). <i>Fit to page</i> : scaled into the printable area and centred.
Auto rotate	Switches to landscape for a page only when page and paper orientation differ. Label papers are often defined landscape already.
DPI	Selects this output resolution of the driver if the driver offers exactly it; otherwise the driver default stays. For label printers use the resolution of the print head (203 or 300).
Copies	<i>One job, copies by the driver</i> or <i>One job per copy</i> . Use the second for drivers that ignore the copy count. PDF24 and Microsoft Print to PDF report copy support but produce only one copy.
Paper source	Tray by name, the editor lists the trays of the driver
Duplex	Driver default, simplex, long edge (vertical), short edge (horizontal)
Color	Driver default, colour, monochrome
Confirmation	Not a driver setting: when the task is confirmed to VeloxFactory, after spooling or after the spooler job completed

All settings apply to the single print job only. The service never changes the configuration of the Windows printer. Some drivers ignore job settings for duplex, colour or tray and use their own defaults; in that case change the setting in the printing preferences of the Windows printer.

How pages are printed

1. The PDF stays in memory, nothing is written to disk.
2. PDFium (the PDF engine of Chromium) draws every page directly onto the device context of the printer, in the device resolution and without an intermediate bitmap. Text, lines and barcodes are sent as vector graphics and print as sharp as the printer can.
3. The target rectangle is computed per page from the printer's resolution and printable area, so pages of different size or orientation in one document work.
4. Output uses the Windows spooler without any print dialog. The document name in the spooler is `VF #{task id} {file name}`.
5. PDFium is not thread-safe; its calls are serialised inside the service, jobs on different printers still run in parallel.

Limitations:

1. Pages with transparency (for example PNG images with alpha channel) are output by PDFium as an image in device resolution.
 2. Form fields (AcroForm) are not printed.
 3. Lines with width 0 in the PDF are drawn as the thinnest line the device can print, one printer dot. On high-resolution printers they become very thin. Give lines and borders in the JRXML an explicit width, e.g. `lineWidth="0.5"`.
-

Printers that open a dialog

Some printers ask for a file name or open an application window when printing. In the headless service there is no desktop: the dialog is invisible and waits forever. The job ends after `printTimeoutSeconds` and is reported as `Timeout`.

The service marks these printers with *Prints with a dialog, not suitable headless*. Detected by port (`PORTPROMPT:`, `XPSPort:`, `PDF24:`, `FAX:`, `SHRFAX:`) or driver name (Microsoft Print To PDF, Microsoft XPS Document Writer, OneNote, PDF24, Fax). In the window they work, the test page asks for confirmation first. For PDF tests without dialog use the [dry run](#) or configure PDF24 with automatic saving.

Test page

The test page is an A4 PDF in the VeloxFactory design, generated locally without API access. It contains instance name, printer, time and account, a line pair pattern that shows the effective resolution, grey scale steps, a Code 39 barcode and a frame 10 mm from the paper edge to check margins and scaling. It is printed with the settings of the printer, including alias.

Print it from the **Printers** page or on the command line:

```
& "C:\kiwisoftware\VeloxPrint\app\bin\kiwisoftware.VeloxFactory.PrintService.App.exe" --test-print WH-LABEL-01
```

i Test under the right account. A test page from the window runs under your account, the headless service under the service account. A printer connected only for your profile prints the test page but fails in production. Check the column *Local*: it must say *Local* or *Machine-wide*.

The Print Service window

Started without parameters, the Print Service opens its window in the VeloxFactory design (light, dark or system theme). The header shows the pages, the instance name and the mode badge: **Interactive** when the engine runs inside the window, **Monitor** when a headless engine runs and the window only shows its state. Screenshots follow.

Dashboard

Tile	Content
Engine	Running, paused (with reason and end), stopped; mode, account, uptime; Pause and Resume
API	OK, token rejected (401/403), rate limited (429), unreachable; time of the last successful request
WebSocket	Subscribed, reconnecting (attempt n), off; reconnects today
Rate limit	Remaining of limit from the response headers, as bar
Queue	Active, waiting, in retry, to confirm (outbox)
Printers	Number of printers OK and with problem, including their names; tooltip with the state per printer, click opens the Printers page. Checked are printers with own settings and printers of jobs from the last 7 days. A problem is any queue state other than Ready.

Below the tiles: the last 20 jobs and the last 5 errors. A click opens the job.

Console

The live log, written for users: *Task #4711 received (WebSocket, Standard) → WH-LABEL-01, 2 copies* instead of raw protocol events.

1. Columns: time, level (icon and colour), area (API, WS, Print, Engine), task, message.

2. Filters: levels as chips, area, text search, a single task, errors only. **Clear filters** resets them.
 3. Autoscroll can be paused. Copy a line, export the selection or all visible lines, click a task to open it.
 4. Protocol details (ping, pong, channel auth) are logged on level Debug and hidden by default.
 5. Buffer: 10,000 lines in the window. In monitor mode the lines come from `state.db`, which keeps the last 50,000 entries.
-

Jobs

The local history from the job journal: id, time, printer, file, copies, pages, channel (WebSocket or poll), duration, status, trace id. Filters for period, status and printer, search by `#id`, file name, trace id or error text.

The detail view shows the timeline of states, error text and attempts. Actions:

Open in VeloxFactory	Opens <code>{baseUrl}/report-print-task/{id}/show</code> in the browser
Print again	Sets the task back to pending in VeloxFactory (after confirmation); the next poll prints it
Send confirmation again	Sends an open outbox entry now. In monitor mode the headless engine repeats open confirmations automatically.
Copy	Printer name (with alias: the local printer) and trace id. The trace id links the job to the history record and print task in VeloxFactory.

Printers

Comparison of VeloxFactory printers, local printers, own settings and printers of recent jobs, with state, hints, test page, printer editor and a link to the Windows print queue. Described in [Printers and printing](#).

Statistics

Local statistics of this instance, nothing is sent to VeloxFactory.

1. Tiles for today and 7 days: jobs printed and failed, error rate, pages.
2. Jobs over time per hour (24 h, 7 days, 30 days), pages and copies per printer, top error categories.
3. Lead time from creation in VeloxFactory to confirmation (median, P95), share of WebSocket and poll, WebSocket availability and reconnects, API calls and 429 events.
4. Export as CSV. The mouse wheel over a chart zooms the chart only.
5. Retention `statistics.retentionDays` (365 days); after 30 days the data is kept as daily aggregates.

Settings

Group	Content
Connection	Base URL, API token, allow unencrypted, timeout, server time zone, Test connection
WebSocket (Reverb)	Scheme, host, port, app key, channel
Receiving	Mode, broadcast ids as chips, tasks without broadcast id, intervals, page size, API base load compared with the rate limit
Processing	Pause with reason and end time, parallel jobs, one job per printer, retries, timeout, confirmation, dry-run folder
Printer defaults	Defaults for all printers
Logging and statistics	Log level, retention, file size, event log, JSON file, statistics retention. Maintenance: Reset statistics and Clear logs (administrator rights)
Operation	Scheduled task: set up, start, remove, state and XML. Local health endpoint. Error notifications.
Import and export	Configuration as file, never with token

1. **Save** writes `config.json`, the engine applies it without restart. Unsaved changes are asked for when leaving the page.
2. **Test connection** checks: reachability, token (`GET /report-print-task?perPage=1`), printer master data (`printer:read`), channel authorization and WebSocket handshake (15 s), rate limit headers compared with the base load.
3. **Reset statistics** counts from now on. The job journal stays, it protects against double printing.

4. **Clear logs** empties the console history and deletes all log files of the instance except today's.
 5. Without write access to the data folder the window offers **Restart as administrator**.
-

Info

Version, .NET runtime, Windows account (with *Administrator* when elevated), mode, instance id, paths of configuration, data and log folder with buttons to open them, and **Create support package** (see [Monitoring and troubleshooting](#)).

Notification area

In interactive and monitor mode the Print Service has a tray icon with status colour (green OK, yellow warning, red error) and a menu: open, pause, exit. Closing the window hides it in the notification area; in interactive mode the engine keeps running. **Exit** stops the engine gracefully: running jobs finish, the outbox is flushed. Print errors show a Windows notification unless `ui.errorNotifications` is off.

Command line reference

The program file is `kiwisoftware.VeloxFactory.PrintService.App.exe` in `app\bin`. Without parameters it opens the window. All other functions are available as parameters, for the scheduled task, scripts and software distribution.

Parameters

Parameter	Effect	Exit codes
(none)	Window, interactive or monitor mode	0
<code>--headless</code>	Engine without window, for the scheduled task. Never creates a window or dialog.	0, 1, 2, 3, 10
<code>--config <path></code>	Alternative configuration file, for several instances per machine. Combines with every other parameter.	
<code>--console</code>	Attaches a console window and writes the log to it as well. For debugging an interactive start.	
<code>--validate-config</code>	Validates the configuration and runs the connection test (reachability, token, rate limit, printer master data, WebSocket). Prints one line per step.	0, 1
<code>--install-task --user <account></code>	Sets up the scheduled task (see Installation and updates). Asks for the password. Administrator rights required.	0, 4
<code>--uninstall-task</code>	Removes the scheduled task of the instance. Administrator rights required.	0, 4
<code>--test-print <printer></code>	Prints the test page on the printer, locally without API. Uses the printer settings and alias of the configuration.	0, 5

Parameter	Effect	Exit codes
<code>--dry-run <folder></code>	Starts the engine headless and writes PDFs into the folder instead of printing, nothing is confirmed. Applies to this run only, the configuration is not changed.	like <code>--headless</code>
<code>--support-package <target.zip></code>	Writes the diagnostic package (see Monitoring and troubleshooting). Works while the engine runs.	0
<code>--stop</code>	Ends engine and windows of the instance gracefully: running print jobs finish, the outbox is flushed. Waits until the engine has ended. Administrator rights required. Used by <code>update.ps1</code> .	0, 6
<code>--help</code> , <code>-h</code> , <code>/?</code>	Shows the usage	0

1. Paths (`--config`, `--dry-run`, `--support-package`) must be absolute. The Task Scheduler starts in `C:\Windows\System32`, a relative path would end up there.
2. Mapped network drives do not exist for the service account. Use local paths or UNC paths.
3. An unknown parameter or a missing value prints the usage and ends with exit code 1.

Exit codes

Code	Meaning
0	OK, or regular stop of the engine
1	Configuration invalid, connection test failed, or invalid parameters
2	The engine of this instance is already running
3	API token rejected at start
4	Scheduled task could not be set up or removed (rights, account, password)
5	Test print failed
6	Stop failed: no administrator rights or the engine did not end in time
10	Unexpected error

Exit codes 1 and 3 of a headless start are visible in the Task Scheduler as *Last run result*. The Task Scheduler restarts the task every minute; fix the configuration or token, the next start picks it up.

Examples

```
$exe = "C:\kiwisoftware\VeloxPrint\app\bin\kiwisoftware.VeloxFactory.PrintService.App.exe"

# check configuration and connection
& $exe --validate-config

# Configuration: C:\kiwisoftware\VeloxPrint\data\lager-nord.json
# [OK]
# [OK]
# [OK] Token: Token accepted, report-print-tasks
# [OK] WebSocket: Subscribed to private-report-print-tasks on vf.example.com:8080

# scheduled task for the default instance and for a second instance
& $exe --install-task --user .\svc-veloxprint
& $exe --config C:\kiwisoftware\VeloxPrint\data\lager-nord.json --install-task --user .\svc-veloxprint

# test page
& $exe --test-print WH-LABEL-01

# engine in a console window for debugging (stop the scheduled task first)
& $exe --headless --console

# dry run into a folder
& $exe --dry-run C:\Temp\veloxprint-dryrun --console

# diagnostic package
& $exe --support-package C:\Temp\veloxprint-support.zip

# graceful stop of the default instance
& $exe --stop
```

i Exit code in PowerShell. The program is a Windows application, PowerShell does not wait for it with the `&` operator. To read the exit code, use `$p = Start-Process -FilePath $exe -ArgumentList '--validate-config' -Wait -NoNewWindow -PassThru; $p.ExitCode`.

Monitoring and troubleshooting

The headless service has no window, so it reports through log files, the Windows event log and an optional local health endpoint. For support cases it bundles everything into one diagnostic package.

Log files

```
data\logs\veloxprint-20260925.log
data\logs\veloxprint-20260925_001.log           same    day,    after
data\logs\veloxprint-20260925.json
data\logs\lager-nord\veloxprint-...           instance with -
```

```
2026-09-25 07:16:48.120 +02:00 [INF] [Print] [#35] Task #35 received (WebSocket, Standard) ->
WH-LABEL-01,                                     1                                     copy
2026-09-25 07:16:49.402 +02:00 [INF] [Print] [#35] Task #35 spooled to WH-LABEL-01, 1 page
2026-09-25 07:16:49.655 +02:00 [INF] [API] Task #35 confirmed as printed
```

Format: time with offset, level (`VRB`, `DBG`, `INF`, `WRN`, `ERR`, `FTL`), area (`API`, `WS`, `Print`, `Engine`), task id where known, message.

1. One file per day, a new file (`_001`, `_002`) when `logging.maxFileSizeMb` (250 MB) is reached.
 2. Files older than `logging.retentionDays` (30) are deleted, by age, not by count.
 3. Lines of a task carry its task id and trace id; the trace id links it to the history record and print task in VeloxFactory.
 4. The API token never appears in any log.
-

Windows event log

Warnings and errors go to the event log *Application*, source **VeloxFactory Print Service**, with fixed event ids for monitoring systems such as Checkmk, PRTG, Zabbix or SCOM. The source is registered when the scheduled task is set up; without it the event log output stays off. Switch it off with `logging.eventLog`.

Event id	Meaning	Action
1000	Engine started	
1001	Engine stopped	Unexpected outside updates and reboots
2000	API not reachable	Network, DNS, proxy, VeloxFactory up?
2001	API token rejected (401/403)	Token revoked or rights missing, enter a new token
2002	Rate limit reached (429)	Occasionally harmless. Often: raise <code>API_RATE_LIMIT_PER_MINUTE</code> or the poll intervals
3000	WebSocket down longer than 5 minutes	Reverb, firewall, host, port, app key. Hybrid keeps printing by polling.
4000	Print failed	Error text contains the category
4001	Printer missing	Install machine-wide or set an alias
5000	Configuration invalid	A changed configuration was not applied, the engine keeps the last valid one

Local health endpoint

Optional, switched on under **Settings → Operation** (`health.enabled`, port `health.port`, default 8765). It listens only on `127.0.0.1`, needs no URL reservation and no administrator rights, and is not reachable from other machines. A monitoring agent on the same machine polls it.

GET	<code>http://localhost:8765/health</code>
<pre>{</pre>	

```

        "rateLimit": { "limit": 60, "remaining": 58, "rateLimitedUntil":
    },
    "websocket": { "state": "Subscribed", "since": "2026-09-25T05:01:14+00
"reconnectAttempt": 0 },
    "queue": { "active": 0, "waiting": 0, "retrying": 0, "outbox": 0
        "printers": { "ok": 3, "withProblems":
    }

```

HTTP status	Condition
200, <code>"status": "ok"</code>	Engine running or paused, and API state OK, unknown or rate limited
503, <code>"status": "degraded"</code>	Everything else, e.g. token rejected or API unreachable
no answer	Engine not running

Worth alerting on besides the status: `queue.outbox` above 0 for a longer time (printed but not confirmed), `queue.retrying` above 0 (printer problem), `printers.withProblems` above 0, `websocket.state` not `Subscribed`.

Diagnostic package

Info → Create support package, or `--support-package C:\Temp\support.zip`. The ZIP contains:

<code>versions.txt</code>	Program version, machine, account, Windows, .NET, configuration path, whether a token is stored
<code>config.json</code>	Configuration, anything resembling a token removed
<code>engine-status.json</code>	Last engine status (as in the health endpoint)
<code>printers.txt</code>	Local printers with scope, queue state, jobs, driver and driver version, port, dialog flag
<code>journal-last-500.csv</code>	The last 500 jobs of the journal
<code>outbox.csv</code>	Open confirmations and error reports: <code>task id;kind;attempts;next attempt;last error</code> . Empty is the normal case.

<code>config-changes.csv</code>	Last 500 configuration changes: time, account, changed keys
<code>scheduled-task.txt</code>	XML definition of the scheduled task
<code>logs/</code>	The 7 newest log files

The API token is removed from every file.

Common problems

Symptom	Cause and fix
Tasks stay pending, nothing in the log	Broadcast id of the task is not in <code>receive.broadcastIds</code> , or the task has none and <code>includeWithoutBroadcastId</code> is off. Processing paused? Engine running (dashboard, Task Scheduler)?
Tasks are printed, but only after a minute or two	Tasks without broadcast id: VeloxFactory sends no event for them, only the poll finds them. Give them a broadcast id. Otherwise the WebSocket is not subscribed: check the WebSocket tile and Test connection (host, port, scheme, app key, firewall). A token without <code>report-print-task:read</code> fails at the channel authorization.
Prints from the window, fails from the scheduled task with <code>PrinterNotFound</code>	Printer only connected for your user profile. Install it machine-wide (Printers and printing).
Network printer fails from the scheduled task	Task set up without stored password, or a local account against a domain print server. Use a domain account and set up the task again.
Error <code>[Timeout]</code> after 120 s	Printer opens a dialog (PDF printers, OneNote, fax), or the driver hangs. Check the hint on the Printers page.
Only one copy instead of n	Driver ignores the copy count. Set <i>Copies</i> to <i>One job per copy</i> for this printer.
Output shifted, cut off or too small	Check scaling (<i>Actual size</i> for labels), paper size of the driver and auto rotate. The frame of the test page shows margins and scaling.
Very thin or missing lines	Lines with width 0 in the report. Set <code>lineWidth</code> in the JRXML.
Printed, but pending in VeloxFactory	Confirmation in the outbox (Queue tile <i>to confirm</i> , <code>outbox.csv</code>). The column <i>last error</i> names the reason. The task is not printed again, the confirmation follows as soon as the API answers.
Task shows error <i>Print state unknown after service restart</i>	The service ended while the task was printing. Check the printer; if nothing came out, use Print again .

Symptom	Cause and fix
Many 429 / event 2002	Several services or scripts behind one IP share the limit. Raise <code>API_RATE_LIMIT_PER_MINUTE</code> , use Hybrid instead of short polling intervals.
Scheduled task shows result 0x1 / 0x3	Exit code 1 configuration invalid, 3 token rejected. Start the window or <code>--validate-config</code> .
Window cannot save settings	No write access to the data folder. Use Restart as administrator .
<code>update.ps1</code> refuses to update	Local changes in <code>app\</code> , or a window of the Print Service is still open under another user. Close it and run the update again.