VTX Printers 1: printing on the device's printers

Nov 15, 2026 · VTX 1 · extension · draft d01 · MIT licence · vtx.foundation/spec/v1/printers-1/

§1 Status and scope

Printers 1 lets a program in the session print on the printers of the user's device. The client tells the session which printers it has and what each can do; a job arrives as a PDF, and the device prints it with its own drivers, as RDP's Easy Print does. No printer driver is ever installed on the host. It is an extension of VTX 1, draft d01, in conformance profile 3, and the second redirection.

Adds	What it is
`redirect.printers`	A capability: the client can print, and says which document formats it accepts
`printers`	A permission: the server asks the user to let the session print
`print` channel	One channel per connection, opened by the client: the printers, the jobs and their states
`printer`	An application event: a printer was added, changed or removed

Why PDF. Every host can make it, every device can print it, and it keeps the layout the program chose. A client may also accept PWG Raster for devices that print without a PDF interpreter, such as some kiosks.

Out of scope. Printing from the device to a printer of the host or the farm, which is the farm's own print service. Scanners and fax. Stapling, punching and other finishing, and choosing paper trays, which stay in the device's own print dialog when the user asks to see it (Section 6). Printer accounting and quotas.

§2 Capabilities, the permission and sharing printers

The client offers `redirect.printers` with the document formats it prints; the server grants it with `{}`. The client shares its printers only when the user has allowed this server to print, and then opens the print channel after READY.

```
cap-printers = {
  ? "formats": [1*4 tstr],    ; client: document formats it prints; must include
"application/pdf"
  ? "max_jobs": 1..16,        ; client: most jobs open at once (default 4)
}
```

Allowing a server to print. The user turns printing on for a server in the client's own settings, or answers the server's PERMISSION with `printers`. The scope works as in v1: `once` for this connection, `session` until the session ends, `always` for this server from now on. Without it, the client opens no print channel and the session sees no printers of the device.

The channel. One channel of type `print` per connection, opened by the client. It carries the printers (client to server) and the jobs (server to client) of every printer, with credits pacing the documents like any other bulk data. Closing it removes every printer from the session; jobs not yet printed are cancelled.

Frame	Direction	What it carries
0x10 PRINTERS	Client to server	Every shared printer, what it can do and its state; again after every change
0x11 JOB	Server to client	A new job: printer, title, format, size and options
0x12 DATA	Server to client	The next bytes of a job's document
0x13 END	Server to client	The whole document has been sent, with its SHA-256
0x14 CANCEL	Server to client	The program cancelled the job
0x15 STATE	Client to server	A job's state on the device

§3 Printers

PRINTERS lists every printer the user shares with the session, with what a print dialog on the host needs to offer: paper sizes, colour, two-sided printing, copies and the printer's state. The client sends it when the channel opens and again whenever something changes: a printer is added or removed, runs out of paper, or goes offline.

```
printer = {
  "id": tstr .size (1..32),   ; chosen by the client for this connection
  "name": tstr .size (1..128),
  ? "default": true,
  "media": [1*32 media-name],       ; PWG names, e.g. iso_a4_210x297mm
  ? "media_default": media-name,
  "color": bool,
  "sides": [1*3 ("one-sided" / "two-sided-long-edge" / "two-sided-short-edge")],
  ? "copies": 1..999,         ; most copies of one job (default 1)
  "state": "idle" / "printing" / "stopped",
  ? "reasons": [1*8 tstr],    ; PWG reason keywords, e.g. "media-empty", "offline"
}
```

A desk with a duplex laser printer and a small photo printer:

id	name	media	colour	sides	copies
`office` (default)	Office printer	A4 (default), Letter, A5	no	one-sided, both two-sided	99
`photo`	Photo printer	A4, 4 × 6 in	yes	one-sided	10

Names. Paper sizes use PWG's self-describing names, `iso_a4_210x297mm` and `na_index-4x6_4x6in`, which CUPS and IPP use too, so a host maps them without a table. States and reasons are IPP's.

Rules. Printer ids are unique, at most one printer is the default, and a printer's default paper size is one of its paper sizes. A printer that can print two-sided can always print one-sided as well.

§4 At a glance

> **Diagram: a print job from a program on the host to the printer on the desk.** Drawn in the PDF version, `14-printers-1.pdf`.

The program prints as it would to any printer; the host's print system makes a PDF, and the server sends it to the device. There the client checks the document, the trusted zone shows the job and holds it while the device is locked, and the device's own driver prints it. The states travel back to the program's print queue.

§5 Jobs

A job is JOB, the document in DATA frames, then END. The client reports what happens to it with STATE until it reaches one of the three final states.

```
JOB   {"job": 1, "printer": "office", "title": "Invoice 0412", "format": "application/
pdf",
       "size": 2048, "pages": 2, "media": "iso_a4_210x297mm", "sides": "two-sided-long-
edge", "color": false}
DATA  {"job": 1, "offset": 0, "bytes": h'255044462d312e370a…'}
DATA  {"job": 1, "offset": 1024, "bytes": h'…'}
END   {"job": 1, "sha256": h'…'}
STATE {"job": 1, "state": "printing"}
STATE {"job": 1, "state": "completed", "pages": 2}
```

Options. A job asks only for what its printer offers: one of its paper sizes, two-sided only where `sides` allows it, colour only on a colour printer, and no more than its `copies`. Options the job leaves out take the printer's defaults. The client applies them through the device's own print system, which also knows what the protocol does not carry, such as trays and finishing.

The document. DATA frames come in order, each starting where the last one ended, until the job's `size`; END follows only then. With `sha256`, the client checks the whole document before it prints. A PDF starts with `%PDF-`; a document that is not what its format says is aborted, not printed. The client may start printing before END only when the device's print system can stream, and never shows a partial document as complete.

States.

State	Means	Next
`pending`	Received, waiting for the printer	`held`, `printing`, or a final state
`held`	Waiting for the user to allow it (Section 6)	`printing`, `canceled`
`printing`	On the printer; `pages` counts what has come out	`completed`, `canceled`, `aborted`
`completed`	Printed, after END	Final
`canceled`	By the user, or after the server's CANCEL	Final
`aborted`	The device could not print it; `reason` says why	Final

A job never moves back, and once it is final the server sends nothing more for it. At most `max_jobs` jobs are open at once; a program that prints more waits in the host's own queue.

§6 The user in control

Paper comes out on the user's desk, so the user decides what prints. The trusted zone shows every job as it arrives, with the server, the title, the page count and the printer, and a button to cancel it.

How a server's jobs are handled. The user picks one of three for each server:

Setting	What the client does
Print at once	Prints each job as soon as it is complete, and shows it in the trusted zone. The usual choice for a trusted desktop
Ask each time	Holds each job (`held`) and shows a preview; the user prints it or cancels it
Show the print dialog	Opens the device's own print dialog with the job's options filled in, so the user can pick a tray, finishing or another printer

While the device is locked, the client holds every job, whatever the setting, so a document never comes out of a shared printer with nobody there to take it.

Cancelling. The user cancels in the trusted zone, or in the device's own print queue; the program cancels through the server's CANCEL. Either way the client stops the job and reports `canceled`, with the pages already printed.

§7 With other extensions, and without the capability

Extension	With printing
Brokering 1	Every host connection has its own print channel. The client keeps the user's printing setting per broker, so a farm whose hosts the broker vouches for needs one decision, not one per host
Viewers 1	Only the owner's connections share printers, as with folders: jobs come out on the owner's desk
Windows 1	A program's print dialog on the host lists the device's printers. With "Show the print dialog", the device's own dialog opens as a local window
Drives 1	Unrelated: printing to a PDF file saves it like any file, on a shared folder if the user picks one

Without the capability. The session sees no printers of the device. Programs print to a PDF file, which the user downloads with Transfer 1, or to the farm's own printers.

§8 Application interface

The server offers each shared printer to the host's print system as a driverless IPP printer on the host itself, with the paper sizes, colour, sides and copies from PRINTERS. Every current desktop prints to such a printer without a driver: CUPS on Linux through IPP Everywhere, Windows through its IPP class driver, and macOS as it would to an AirPrint printer. The print system converts the job to PDF, and the server sends it as JOB, DATA and END. No document ever crosses the application interface.

Host	The printer appears as
Linux	A CUPS queue, `Office printer (via VTX)`, that only the session's user may print to

Host	The printer appears as
Windows	A printer in the session, with the same name, on the IPP class driver
macOS	A printer in the session's print dialogs, with the same name

The desktop shell learns about printers through `printer` events, so its settings and print dialogs can mark the device's default and show a stopped printer:

```
{"kind":"printer","action":"added","id":"office","name":"Office
printer","queue":"Office printer (via VTX)","default":true,"state":"idle"}
{"kind":"printer","action":"changed","id":"photo","state":"stopped"}
{"kind":"printer","action":"removed","id":"photo"}
```

§9 Security

A server that may print can waste paper and put documents in front of other people. A printer list also says something about where the user is. The rules below keep printing to what the user expects.

- **Only when allowed.** No print channel without the user's say for that server (Section 2). The setting is per server identity, or per broker for a farm, and the user can withdraw it at any time.
- **Every job in sight.** The trusted zone shows each job with its server, title, pages and printer, and can cancel it. While the device is locked, every job waits.
- **No drivers, no code.** The host never installs a printer driver, and the device receives a document, not a program: a PDF or PWG Raster, checked against its format and its SHA-256 before it prints. A document that is not what it says is aborted.
- **Little about the place.** PRINTERS carries a printer's name and what it can do, never its network address, serial number or the device's print server. The user can share only some printers, or rename them before they are shared.
- **Limits.** At most `max_jobs` open jobs (16 at the most), 256 MiB per document, and credits pacing the data. A client may abort a job that prints far more pages than JOB announced.

§10 Definitions and checks

The definitions live in the bundle under `ext/printers-1/`:

File	What it holds
`vtx-printers-1.cddl`	The capability, the `printers` permission, and the printers, jobs, documents and states on the print channel
`vtx-app-events-printers-1.schema.json`	Server lines to an application, plus the `printer` event. Printers 1 adds no scene op and no application line
`examples/valid/`	The desk trace (an invoice two-sided, a photo in two copies through an empty paper tray, a draft cancelled by its program, a payslip held while the laptop is locked) and the printer events
`examples/invalid/`	Broken printer lists, printer events and traces, each with the check that must catch it

`tools/gen_ext_printers.py` writes the schema, built on the Drives 1 generator; `tools/make_printers.py` writes the examples, with small made-up PDFs of fixed bytes. Both are deterministic.

The validator's Section 22 runs 52 checks, and every one passes:

Part	Checks	What they cover
Setup	6	The schema is valid; the CDDL parses alone and with every earlier extension; no rule is defined twice; earlier schemas reject `printer`; the core alone rejects the `printers` permission
CDDL and printer lists	18	6 valid samples accepted and 8 malformed ones rejected; 4 printer lists pass the CDDL and fail a rule of Section 3
Traces	22	The valid trace replayed in full, every document checked against its size, SHA-256 and format; 21 broken ones, each caught by the right check, among them colour on a black-and-white printer, a gap in a document, a PostScript file sent as a PDF, a hash that does not match, a job completed before its END, and data for a job that was cancelled
Application interface	6	4 valid printer events, 2 rejected

The whole bundle runs 2,543 checks, all passing.

§11 Wire summary

Everything Printers 1 adds, for the registry and for implementers:

Kind	Addition
Capability	`redirect.printers`: from the client, `formats` (with `application/pdf`) and `max_jobs` (1 to 16; default 4); the server grants `{}`
Permission	`printers`: the server asks the user to let the session print
Channel type	`print`, opened by the client, at most one per connection: 0x10 PRINTERS, 0x11 JOB, 0x12 DATA, 0x13 END, 0x14 CANCEL, 0x15 STATE
Job states	`pending`, `held`, `printing`, `completed`, `canceled`, `aborted`
Names	Paper sizes as PWG 5101.1 self-describing names; printer states and reasons as IPP's
Application event	`{"kind":"printer","action":"added", "changed" or "removed","id":…,"name":…,"queue":…,"default":…,"state":…}`

No frame of the core or of an earlier extension changes meaning. Stopping printing is v1's CLOSE on the print channel.