PrusaLink Local HTTP API Reference (Summary)

Base URL

Use the printer's local IP or hostname:
    http://<printer-ip>/

All endpoints below are relative to this base URL.

Authentication

Firmware and setup determine which auth method is active:

1) API key (older behavior – still present on many printers)
   - Header: X-Api-Key: <your_api_key>

2) HTTP Digest authentication (username/password)
   - Used by newer firmwares and many examples:
       curl --digest -u <user>:<password> http://<printer-ip>/api/v1/status

In your Discord bot / script you typically:
- Use HTTP digest auth, OR
- Add X-Api-Key header, depending on how your printer is configured.

Core Data Structures (Conceptual)

The API exposes several main objects (JSON):
- Version – versions of firmware, PrusaLink, API, capabilities
- Info   – static printer info (model, serial, features)
- Status   – live telemetry; includes printer + job + transfer + storage + camera
- Job   – current print job info
- Transfer – file transfer status
- Storage – available file storages and their usage
- FileInfo – metadata about files and directories
- Camera   – camera list, individual camera configs
- Update   – information about available PrusaLink updates

If you just want "current print progress + time info", the primary endpoint is:
- GET /api/v1/status (job.time_printing, job.time_remaining, job.progress, etc.)

---

1. Version & Basic Info

GET /api/version

Description:
    Returns API + firmware + PrusaLink version information and capabilities.

Typical usage:
    - Detect available features (e.g. upload-by-put in capabilities.upload-by-put).
    - Show firmware/PrusaLink version in your integration.

Relevant fields (response JSON):
    api   – API version string
    version   – PrusaLink version
    firmware   – printer firmware version
    printer   – printer firmware API version / model info
    text   – human-readable version string
    sdk   – SDK version
    capabilities – object with booleans like upload-by-put

GET /api/v1/info

Description:
Returns static printer information.

Typical content (logical, not exhaustive):
- printer model / name
- serial number
- hardware / feature flags (MMU, etc.)

Use this when:
- You need to list connected printers and their models.
- You want serial numbers or static metadata for logs/UI.

---
2. Live Status & Job Info

GET /api/v1/status

Description:
Main telemetry endpoint. Returns printer, job, transfer, storage and camera data in one JSON object. All sections except "printer" are optional.

Shape (conceptual JSON):

```
{
  "job": {
    "id": <int>,                // current job ID
    "progress": <float>,        // 0.0–100.0
    "time_printing": <int>,     // seconds already printed
    "time_remaining": <int>     // seconds remaining (estimate)
    ... other job fields ...
  },
  "printer": {
    "state": "PRINTING" | ...,
    "temp_bed": <float>,
    "target_bed": <float>,
    "temp_nozzle": <float>,
    "target_nozzle": <float>,
    "axis_z": <float>,
    "flow": <int>,
    "speed": <int>,
    "fan_hotend": <int>,
    "fan_print": <int>
    ... other telemetry fields ...
  },
  "transfer": { ... },
  "storage": { ... },
  "camera": { ... }
}
```

Key fields for your bot:
- job.progress – % complete
- job.time_printing – seconds already printed
- job.time_remaining – seconds remaining
- printer.state – IDLE / PRINTING / FINISHED / ERROR, etc.

Behavior:
- Always returns printer.* data when authenticated.
- job.* only appears if there is an active job.
- Great for periodic polling in automations or Discord bots.

GET /api/v1/job

Description:
Returns detailed info about the current job, if one exists.

Responses:
200 OK – Job object (similar to status.job, but can be more detailed)
204 No Content – No job currently running

Use this when:
- You need more detailed per-job info (file name, storage, etc.).
- You already know there is a running job (e.g. from /api/v1/status).

---
3. Job Control

These endpoints operate on a specific job ID. You usually obtain the ID
from /api/v1/status or /api/v1/job.

DELETE /api/v1/job/{id}

Description:
Stop (cancel) a specific job.
Responses:
204 No Content – success
404 Not Found – job ID not known
409 Conflict – job in wrong state

PUT /api/v1/job/{id}/pause

Description:
Pause the given job.
Responses:
204 No Content – success
404 Not Found
409 Conflict – cannot pause in current state

PUT /api/v1/job/{id}/resume

Description:
Resume a paused job.
Responses:
204 No Content – success
404 Not Found
409 Conflict – cannot resume in current state

PUT /api/v1/job/{id}/continue

Description:
Continue a job after a timelapse capture or other intermediate stop.
Responses:
204 No Content – success
404 Not Found
409 Conflict

Typical workflow for your bot:
1. Poll /api/v1/status to get job.id + job.state.
2. Use pause/resume/continue on that ID to control the print.

---
4. File Storage & Transfers

GET /api/v1/storage

Description:
Lists available storage backends (local, SD card, USB, etc.) and their capacity/usage.

Response (conceptual):

```
{
  "storage_list": [
    {
      "name": "PrusaLink gcodes",
      "type": "LOCAL" | "SDCARD" | "USB",
      "path": "/local",
      "print_files": <bytes>,
      "system_files": <bytes>,
      "free_space": <bytes>,
      "total_space": <bytes>,
      "available": <bool>,
      "read_only": <bool>
    },
    ...
  ]
}
```

Use this when:
- You want to know where you can upload files.
- You want to show storage usage in your UI.

GET /api/v1/transfer

Description:
Information about the current file transfer (if any).

Responses:
200 OK – transfer object
204 No Content – no transfer in progress

DELETE /api/v1/transfer/{id}

Description:
Cancel a running file transfer.

---
5. Files API

All paths here are relative to a storage path from /api/v1/storage.

Path parameters:
{storage} – e.g. "/local" or "/usb"
{path} – file or folder path on that storage

Headers:
Accept-Language – optional; language for human-readable names
Accept – "application/json" or "text/html" (others as text/plain)

GET /api/v1/files/{storage}/{path}

Description:
Get metadata for a file or folder.

Response:

- FileInfo / PrintFileInfo / FirmwareFileInfo / FolderInfo JSON

Use this to:
- List folders, inspect file metadata, show print time estimates from G-code, firmware file info, etc.

PUT /api/v1/files/{storage}/{path}

Description:
Upload a file or create a directory.

Headers:
Content-Length – size in bytes
Content-Type – usually "application/octet-stream"
Print-After-Upload – "?0" or "?1" (start printing after upload)
Overwrite – "?0" or "?1" (overwrite existing file)

Body:
Raw file content (binary).

Responses:
201 Created
404 Not Found (invalid path)
409 Conflict (e.g. overwrite not allowed)

POST /api/v1/files/{storage}/{path}

Description:
Start printing an existing file, if no job is currently running.
Body:
Ignored.

Responses:
204 No Content – print started
409 Conflict – job already running, or other conflict

HEAD /api/v1/files/{storage}/{path}

Description:
Check for file presence and state via headers only.

Important headers in response:
Read-Only – boolean
Currently-Printed – boolean (true if this file is currently being printed)

DELETE /api/v1/files/{storage}/{path}

Description:
Delete a file or folder.

Headers:
Force – "?0" or "?1" to allow deleting non-empty folders.

Responses:
204 No Content
409 Conflict – e.g. trying to delete a file that is currently being printed

---
6. Cameras

GET /api/v1/cameras

Description:
List all configured cameras and their properties.

Responses:
200 OK – array of Camera objects

PUT /api/v1/cameras

Description:
Update the list/order of cameras.
Body:
JSON array of strings (printer IDs / camera IDs in target order).

GET /api/v1/cameras/{id}

Description:
Get configuration and properties for a specific camera.

POST /api/v1/cameras/{id}

Description:
Create or repair camera configuration.

Body:
CameraConfigSet JSON (name, trigger scheme, resolution, etc.).

DELETE /api/v1/cameras/{id}

Description:
Delete a camera from the list.

GET /api/v1/cameras/snap

Description:
Return a PNG snapshot from the default camera.
Responses:
200 OK – image/png
204 No Content – no snapshot available
304 Not Modified – if you use caching headers

GET /api/v1/cameras/{id}/snap

Description:
Return a PNG snapshot from the camera with the given ID.

POST /api/v1/cameras/{id}/snap

Description:
Trigger a snapshot capture (mostly during initialization or in manual mode).
Response:
200 OK – returns image/png

PATCH /api/v1/cameras/{id}/config

Description:
Update camera settings (resolution, rotation, focus, etc.).
Body:
CameraConfigSet JSON.

DELETE /api/v1/cameras/{id}/config

Description:
Reset camera settings to defaults.

POST /api/v1/cameras/{id}/connection

Description:
Register camera to Prusa Connect.

DELETE /api/v1/cameras/{id}/connection

Description:
Unregister camera from Prusa Connect.

---
7. Updates

Path parameter:
{env} – currently "prusalink" for the PrusaLink package itself.

GET /api/v1/update/{env}

Description:
Check if an update is available for the given environment.

Responses:
200 OK – JSON with package info, header Update-Available: true
204 No Content – header Update-Available: false

POST /api/v1/update/{env}

Description:
Trigger an update of the specified environment (e.g. PrusaLink).

---
8. Practical Tips for Your Bot
---

1) Getting basic status and print time info
- Call GET /api/v1/status with proper authentication.
- Use:
  job.progress -> percentage
  job.time_printing -> seconds printed
  job.time_remaining -> seconds remaining
  printer.state -> state to display to user

2) Starting a print from an uploaded file
- Use /api/v1/storage to find desired storage path (e.g. "/local").
- Upload the file with PUT /api/v1/files/{storage}/{path}.
- Optionally set Print-After-Upload header to start immediately,
  or later call POST /api/v1/files/{storage}/{path}.

3) Pausing / resuming from a Discord command
- Poll /api/v1/status to get job.id and state.
- Use:
  PUT /api/v1/job/{id}/pause
  PUT /api/v1/job/{id}/resume
  DELETE /api/v1/job/{id} (cancel)

4) Camera snapshot for embedding in Discord
- Use GET /api/v1/cameras/snap or GET /api/v1/cameras/{id}/snap.
- Treat response as PNG image bytes and send as an attachment.

This document is a summary based on Prusa's OpenAPI spec and community
libraries. For absolute ground truth, see:

https://github.com/prusa3d/Prusa-Link-Web (spec/openapi.yaml)