The .hoop project file format

Public specification · Format version 1 · Revision 2026-07-27

.hoop is an open container format for machine-embroidery *projects*: the editable design objects, thread colours, trace images and a derived, machine-neutral stitch list, in a single file. Anyone may implement readers and writers. No licence, registration or royalty applies to the format.

This document is the normative reference. It is generated from the living specification at **hoop.stitchpencil.app**, which is the version to check if the two ever disagree.

The key words MUST, MUST NOT, SHOULD and MAY carry their usual normative meaning.

Why a project format

The established machine formats (DST, EXP, PES, JEF) describe a needle path. That is all a machine needs and all they were ever meant to carry. What they cannot carry is the design: the shape a fill was generated from, the width a satin column was given, which thread from which catalogue was intended, or the photo someone traced. Reopening a DST gives you stitches, and a stitch soup cannot be re-digitised — only re-drawn.

A .hoop file keeps both layers, honestly separated:

Layer	Entry	Role
Editable project	document.json	The single source of truth
Derived stitches	plan.json	Regenerable output, never imported back

A reader that only wants stitches never has to understand the editable model. A reader that wants to edit never has to trust the stitch list.

The minimum useful reader

Tools that only need to produce machine files have a short job:

- Open the archive as a ZIP.
- Read `plan.json`.
- Encode DST, PES, EXP, JEF or anything else from its stitch list.

Coordinates are millimetres, the origin is the hoop centre, y points up. Thread colours carry real catalogue identities rather than a palette index guess. The editable model exists for tools that want to go further, and can be ignored entirely.

1 Container

A .hoop file is a standard ZIP archive (PKZIP). No custom header, no wrapper, no magic bytes beyond ZIP's own. Renaming a .hoop to .zip is a supported way to look inside.

Permitted ZIP features

- Writers MUST use only compression method 0 (stored) or 8 (deflate).
- Writers MUST NOT use encryption, spanning or ZIP64.
- Writers SHOULD write manifest.json as the first entry.
- Writers SHOULD deflate JSON and store already-compressed images.

Readers MUST reject archives using any other method or feature rather than attempting a partial read.

Canonical layout

```
Design.hoop
■■■ manifest.json     REQUIRED  format version, checksums, entry roles
■■■ document.json     REQUIRED  the editable project (source of truth)
■■■ plan.json         optional  derived stitch list
■■■ preview.png       optional  512 px render of the design
```

```
■■■ references/       optional  original trace-image bytes
    ■■■ <uuid>.<jpg|png|heic|img>
```

Only manifest.json and document.json are required. A reader MUST cope with any other entry being absent rather than assuming a default. preview.png is a rendered image for galleries and file pickers; references/ holds the original bytes of trace images, matched to their metadata in document.json by the lowercased UUID prefix of the entry name.

Entry names

Entry names are UTF-8, use / as the separator, and are relative. Readers MUST reject any archive containing an entry whose name has a `..` component, begins with / or a drive letter, or contains a NUL byte — including files the reader's own application wrote.

2 manifest.json

Required. A reader SHOULD be able to answer three questions from it alone, before touching anything else: can I read this version, what is in the file, and is any of it damaged.

```
{
  "formatVersion": 1,
  "generatorApp": "StitchPencil",
  "generatorVersion": "0.1",
  "created": "2026-07-11T12:00:00Z",
  "title": "Koi Pond",
  "entries": [
    { "path": "document.json", "sha256": "<hex>",
      "role": "document", "derived": false },
    { "path": "plan.json", "sha256": "<hex>",
      "role": "plan", "derived": true }
  ]
}
```

Fields

Field	Type	Req.	Meaning
formatVersion	integer	yes	Container version. Increases only on breaking changes.
generatorApp	string	no	Writing application. Informational.
generatorVersion	string	no	Its version. Informational.
created	ISO-8601	no	Doubles as last-saved: the manifest is rewritten on every save.
title	string	no	Mirrors the document title.
entries	array	yes	One record per archive entry.

Writers may add summary fields alongside these — StitchPencil writes hoopWidthMM, hoopHeightMM, paletteHex and stitchCount so a gallery can list projects from the manifest and preview alone, without inflating any document. Readers MUST ignore fields they do not know.

Entry records

Field	Type	Meaning
path	string	Entry name inside the archive.
sha256	hex string	Lowercase hex SHA-256 of the entry's uncompressed bytes.

Field	Type	Meaning
role	string	document | plan | preview | reference. Open set — tolerate unknowns.
derived	boolean	True if regenerable from document.json.

Checksums and recovery

Verification is not optional, but the response depends on what failed:

Failing entry	Required behaviour
document.json	The file is corrupt. Fail the load with a clear error.
A derived entry	Drop it and continue. It can be regenerated.
A reference entry	Drop the image, keep the project, continue.

This is why checksums are per entry rather than one over the archive: a truncated preview should cost a thumbnail, not a design. ZIP's own CRC-32 is a separate, weaker check on a different layer; readers MUST verify both.

> *An archive may contain entries the manifest does not list. Readers MUST ignore them — that is how future revisions add content without breaking existing readers.*

3 plan.json

The derived stitch list in an open, machine-neutral form, so that a third party can produce DST, PES, EXP, JEF or anything else without understanding the editable model. If you are writing an exporter, this section is the only one you need.

It is optional. Exported files carry it; an application's own library saves commonly do not. Readers MUST handle its absence.

Coordinate system

- Unit: millimetres, floating point.
- Origin: the hoop centre.
- **y points up.** Most machine formats have y pointing down; converting is a sign flip, and forgetting it is the single most common porting bug.

Shape

```
{
  "stitches": [
    { "position": { "x": 11.844811, "y": 1.330744 }, "function": "jump" },
    { "position": { "x": 12.100000, "y": 1.500000 }, "function": "stitch" }
  ],
  "commands": [
    { "kind": { "colorChange": { "index": 3 } }, "beforeStitchIndex": 0 },
    { "kind": { "trim": {} },                    "beforeStitchIndex": 505 }
  ],
  "colorTable": [
    { "red": 47, "green": 48, "blue": 50,
      "catalogId": "madeira-polyneon", "code": "1800", "name": "Black" }
  ],
  "objectRanges": [
    { "objectID": "63C5BD58-…", "start": 0, "end": 502 }
  ],
  "stats": {
    "stitchCount": 2214, "jumpCount": 9, "trimCount": 8,
    "colorChangeCount": 4,
    "bounds": { "minX": -17.697304, "minY": -20.741376,
                "maxX": 17.492767, "maxY": 21.952183 }
  }
}
```

stitches

An ordered array of **absolute** needle positions — there are no deltas anywhere in this format, which is what keeps it immune to the drift that plagues delta-encoded machine formats.

function	Meaning
stitch	Needle down. A normal stitch.
jump	Pen-up movement. The machine moves without stitching.
travel	A needle-down connecting run between parts of a design. Sewn exactly like a normal stitch.

Readers that do not distinguish travel MUST treat it as stitch. Treating it as a jump produces a file that skips stitches the design needs.

commands

Machine commands anchored *before* the stitch at beforeStitchIndex. Several may share an index; they apply in array order. The JSON shape is a single-key object naming the kind, with the payload as its value — including for payload-less kinds, which encode as an empty object.

Kind	Payload	Meaning
colorChange	{ "index": Int }	Change to that thread in colorTable.
trim	{ }	Cut the thread.
stop	{ }	Pause the machine without a thread change.

colorChange indexes colorTable, not the document palette.

That table is generated for the plan and is not the same array as the palette in document.json; it may be longer, ordered differently, contain entries the document does not. Indexing the document palette with a plan index will silently give you the wrong colour.

colorTable

Field	Type	Meaning
red, green, blue	0–255	Display colour. Always present.
catalogId	string, optional	Thread catalogue, e.g. madeira-polyneon.
code	string, optional	Catalogue number, e.g. 1800.
name	string, optional	Human-readable name, e.g. Black.

The three optional fields are what a machine format cannot carry. DST has no colours at all; PES and JEF map to a manufacturer's fixed palette. Here the identity survives, so a worksheet can say *Madeira Polyneon 1800 Black* rather than *the dark one*. Any of them may be null for a freely picked colour; only the RGB triple is guaranteed.

objectRanges

Maps regions of the stitch list back to the design objects that produced them. start is inclusive, end exclusive, both indexing stitches. Ranges are in stitch order and together cover the whole array. objectID matches an object id in document.json when that entry is present.

stats

Field	Meaning
stitchCount	Number of **sewn** stitches.
jumpCount	Number of jumps.
trimCount	Number of trim commands.
colorChangeCount	Number of colour-change commands.
bounds	Design extent in millimetres.

> *stitches.count and stats.stitchCount are different numbers on purpose. In the sample Cherry.hoop the array holds 2223 entries, stitchCount is 2214, and the difference is exactly the nine jumps. Quote stitchCount where a user expects “how many stitches”; use stitches.count for indexing.*

Derived and write-only

- Editors MUST NOT import plan.json back into editable state. Doing so would replace a design with its own shadow.
- Re-writers MAY drop it, and SHOULD regenerate rather than copy it if the document changed.
- If plan.json and document.json disagree, document.json is right and the plan is stale.

Tools that only consume — exporters, viewers, estimators, stitch players — are free to use it as their only input. The restriction is on writing back, not on reading.

4 document.json

The editable project and the single source of truth. This section describes the structure a reader can rely on. It deliberately does not enumerate every parameter — the parameter set grows additively with the application, and enumerating it here would produce a document that is wrong by the next release.

```
{
  "formatVersion": 1,
  "id": "<uuid>",
  "meta":     { "title": "Cherry", "createdAt": …, "modifiedAt": …,
               "appVersion": "0.1" },
  "hoop":     { "name": "130 × 180 mm", "widthMM": 130, "heightMM": 180 },
  "palette":  [ { "red": 47, "green": 48, "blue": 50,
                  "catalogId": "madeira-polyneon", "code": "1800",
                  "name": "Black" } ],
  "projectColors": [ … ],
  "groups":   [ { "id": "<uuid>", "name": "Cherry 1 blend",
                  "joined": false } ],
  "references": [ … ],
  "objects":  [ … ],
  "settings": { "maxTravelMM": 6, "fabric": "knit", … }
}
```

All coordinates are millimetres, hoop-centred, y up — the same system as plan.json.

palette

Ordered thread colours. Objects reference them **by index**, which makes the array structural: an entry an object refers to MUST NOT be removed, and reordering it changes the design. projectColors is a separate, non-structural convenience list; nothing references it by index.

objects

The design itself. **Array order is stitch sequence** — the first object is sewn first, and that is also the drawing order, so later objects sit on top of earlier ones.

Field	Meaning
id	Stable identity. Matches objectID in the plan's objectRanges.
kind	run | satin | fill | manualStitch.
name	User-facing label.
isVisible	Hidden objects are not stitched.
isLocked	Editing guard. No effect on output.
colorIndex	Index into palette.
displayOpacity	Canvas display only. The stitch file always sews solid thread.
geometry	One of four cases, below.
params	Generator parameters for this object's kind.

Optional per-object fields (absent means the default):

Field	Values	Meaning
groupID	uuid	Membership in a groups entry.

Field	Values	Meaning
postCommand	none | stop | colorChange	Manual machine command after this object. stop pauses without a thread change; colorChange forces a stop even for the same thread.
trimAfter	auto | always | never	Thread handling toward the next object. never connects with an uncut jump regardless of distance.
directionLocked	boolean	The stitch angle is governed by the object's group rather than itself.

geometry

Exactly one of four cases, encoded as a single-key object:

Case	Payload	Used by
centerline	{ "points": […] }	Runs, and satins defined by a centre line plus a width
rails	{ "railA": […], "railB": […] }	Satins digitised as two rails
region	{ "outer": […], "holes": [[…], …] }	Fills, with optional holes
points	{ "points": […] }	Manual stitches

Points are { "x": Double, "y": Double } in millimetres. Rails in the rails case are **index-paired**: railA[i] faces railB[i], and the pairing is what defines each stitch, not arc length. A reader that re-samples one rail without the other destroys the column.

params

Which keys are meaningful depends on kind — a fill ignores satinWidthMM, a run ignores angleDeg. Common ones include stitchLengthMM, densityMM, angleDeg, pullCompMM, passes, underlay, tieIn/tieOff, capStart/capEnd, plus feature-specific sub-objects such as fillPattern, fillGuides, sashiko, satinEdge and foam. This set grows: new keys appear in minor revisions without a formatVersion bump, and a reader MUST ignore the ones it does not know.

> *document.json describes intent, not output. How many stitches an object produces, where its seam lands, whether a connection becomes a trim — all of that is the result of running a generator over these parameters, and none of it is recorded here.*

5 Reader requirements

Compatibility

These four rules are what make the format survivable. They are normative.

- Readers MUST ignore unknown JSON keys and unknown container entries.
- formatVersion increases only on breaking changes. A reader encountering a version greater than it supports MUST refuse with a clear “created with a newer version” error, and MUST NOT guess-read.
- Entries marked "derived": true are regenerable; readers and re-writers MAY drop them.
- document.json is the single source of truth; plan.json MUST NOT be imported back into editable state.

Security

.hoop files are untrusted input — including files your own application wrote, because you do not know what happened to them in between. A conforming reader enforces all of the following.

Archive limits. The values below are the reference implementation's. A reader MUST enforce caps of this kind; the exact numbers are an implementation choice.

Limit	Reference value	Why
Entry count	≤ 256	A project has a handful of entries; thousands is an attack or a bug.
Per-entry uncompressed size	≤ 64 MB	
Total uncompressed size	≤ 256 MB	
Compression ratio per entry	≤ 100:1	Zip bombs. A 1 MB entry inflating to 1 GB is not a design.

Entry names. Reject any name containing a `..` component, beginning with /, or containing a NUL byte. Never construct a filesystem path from an archive entry name without this check — this is the classic Zip Slip.

Integrity. CRC-32 per entry as the archive is read, SHA-256 per manifest record afterwards, with the recovery behaviour from section 2.

JSON sanity. Parsing succeeding does not make the content safe.

Check	Reference value
Objects per document	≤ 10 000
Points per path	≤ 50 000
Palette entries	≤ 512
Reference images	≤ 64
Coordinates	Reject non-finite values (NaN, ±∞)
Hoop size	Reject absurd dimensions

Non-finite coordinates deserve particular attention: JSON permits a parser to produce them, geometry code rarely checks for them, and a single NaN propagates through a bounding box into a layout that renders nothing and reports no error. Numeric parameters SHOULD additionally be clamped into their sensible ranges while decoding rather than validated afterwards.

Images. Decode reference images through a hardened decoder with a pixel-count limit. An image decoder is the largest attack surface in the whole format, and the one part of a .hoop reader that is usually somebody else's code.

Writing

Writers have far fewer obligations than readers, and the ones they have are in section 1. The asymmetry is deliberate: a strict writer and a lenient reader is how formats rot. This one asks for a strict reader and a conservative writer.

6 Versioning

The formatVersion field in manifest.json is the only version a reader has to branch on. It increases only when a conforming reader of the previous version would get the file wrong. There is no minor version number.

Does not bump the version

- New keys in any JSON object.
- New entries in the archive.
- New values in an open set — catalogue identifiers, entry roles, parameter sub-objects.
- New optional per-object fields whose absence has a defined default.

Does bump the version

- Removing or renaming an existing key.
- Changing the meaning, unit or coordinate system of an existing value.
- Changing the container in a way an existing reader cannot parse.

Version 1 has already grown additively: the document model gained groups, a directionLocked flag, and parameter sub-objects for features that did not exist when the format was published; the manifest gained the gallery summary fields. None of it moved the version, and a reader from the first day still opens today's files.

formatVersion	Date	Changes
1	2026-07-11	Initial public specification.

7 Sample files

Two real files written by StitchPencil, for implementers to test readers against. Both are downloadable from hoop.stitchpencil.app. They are not synthetic fixtures: both are actual projects, with the irregularities that come with that.

File	Size	Contents	Stitches	Colours
Cherry.hoop	72 KB	manifest, document, plan, preview	2 214	4
Japan.hoop	43 KB	manifest, document, preview	5 628	1

```
b936db0c879257f62447323c2e4ce2d6baf553cbd69ba457d89da38eddce6f0f  Cherry.hoop
95605b720ef730344771bd6e55bd0c2e1f12328b20973ffa38786dd98c93b8b3  Japan.hoop
```

Cherry.hoop is an exported file and carries plan.json. Start here if you are writing an exporter: fill and satin objects, blend groups, several thread colours from a real catalogue, colour-change and trim commands, and all three stitch functions including travel. Its plan is where the numbers in section 3 come from.

Japan.hoop has no plan.json, because it is a library save rather than an export. That is not a defect in the sample; it is the case the specification requires you to handle, and a reader that assumes the entry is present will fail on it. It also carries a sashiko-patterned fill and a lettering group, so its document.json exercises parameter sub-objects a reader must skip over gracefully.

Terms of use

The **format** is open: implement readers and writers freely, no permission needed, nothing to sign.

The **designs in these two files** are not. They are provided solely so that developers can test an implementation, and they remain the property of their author.

You may:

- download them and open them in your own software,
- use them while developing, testing and debugging a .hoop implementation,
- include them in an automated test suite that stays inside your organisation.

You may not:

- redistribute them, publish them, or bundle them with software, a library, a repository or a dataset,
- sell them or include them in anything sold,
- stitch them out for sale, or sell items made from them,
- use them in marketing, on a product page, or as sample content in a shipped application,
- create and distribute derivative designs based on them.

If you need a design you can ship, digitise your own — the format is open precisely so you can. For anything not covered here, ask before assuming: **hello@stitchpencil.app**.

> *These terms cover the two sample designs only. They say nothing about .hoop files you create, which are yours, or about the format itself, which is free to implement.*

Format version 1 · Revision 2026-07-27 · The living specification is at hoop.stitchpencil.app, and is what to check if this document and the site disagree.