

Sample stage receipts — preview

****PREVIEW ARTIFACT, 11 September 2026.**** These are illustrative samples of a record format whose first Internet-Draft is on the datatracker (``draft-saha-stage-receipts-00``, posted 8 September 2026, <https://datatracker.ietf.org/doc/draft-saha-stage-receipts/>) and of its ****forthcoming**** reference SDK. The draft is a draft, not a standard; the SDK does not exist yet. Nothing here is a released implementation, a stable schema, or a capability claim.

They are published now for one reason: the Validation Battery says what a conforming implementation must do, and a claim like that is worth more if you can hold something in your hand and check it.

Check it yourself, in three commands

```
```sh
sha256sum receipts/*.json MANIFEST.json # our digests, your tool
python3 canonicalize.py receipts/*.json # is each file its own canonical form?
python3 verify.py MANIFEST.json # the chain, offline
```
```

Or run everything, including the rejection vectors:

```
```sh
./run_checks.sh
```
```

****No network. No key. No account. No call home.**** A verifier that needs any of those has made you trust someone again, which is the problem this format exists to remove.

What is here

| Path | What it is |
|--|---|
| <code>`receipts/*.json`</code> | A four-stage chain for one question: retrieve → rerank → generate → verify |
| <code>`MANIFEST.json`</code> | The chain: ordered digests and the chain head |
| <code>`canonicalize.py`</code> | The canonical form, specified as runnable code |
| <code>`verify.py`</code> | A minimal reference verifier — ~140 lines, no dependencies |
| <code>`rejection/`</code> | Ten vectors a conforming verifier **must refuse** , each with its reason |
| <code>`_build.py`</code> , <code>`_build_rejections.py`</code> | How the samples were generated, so nothing is hand-waved |

Read the third receipt first

``03-generate.json`` records a ****refusal****. The generator produced no claim that could be bound to a retrieved passage, so the run has no answer — and the receipt says so, in its own outcome class, with a reason. The fourth receipt then records zero claims examined and states how that zero was derived.

That is the sample we would most like you to look at. A format that only records successes is a marketing surface. **Nothing to verify is a result, not an absence.**

What these samples do NOT show

- ****Non-linear topology.**** This chain is linear. Fan-out, fan-in, retries as distinct attempts, branches and loops are an open requirement of the format and are not represented here. It is the largest hole and we would rather name

it than crop the picture.

- ****Anchoring.**** Every receipt declares ``anchor.state: unanchored``, and the verifier reports originality as NOT-RUN with that reason. These records are consistent with themselves. They are not evidence that they are the records that were made — that requires anchoring to a commitment the operator cannot rewrite, which is an open requirement.
- ****Signatures.**** Sequenced behind actor identity; not present.
- **Payload contents.** The digests here reference bytes that are not published with the samples; the structure is the point.