Release artifacts and verification

Use this page to distinguish artifacts that identify the recorded v1.0.0 source release from artifacts that the later, unreleased workflow is designed to produce. Never infer an image, signature, SBOM, attestation, or private package from t

SECTION	Project and Releases
TYPE	Reference
PROTOCOL	AIP 1.0
SOURCE	docs/reference/release-artifacts.md

Evidence state	What exists in the reviewed source
Recorded release identity	Annotated `v1.0.0` tag, exact commit and tree, source-snapshot metadata, changelog, 52 native schemas, and legal files
Defined for a future tag	Thirteen multi-platform image jobs, source-boundary inventories, image digest files, Cosign verification records, SBOMs, and provenance attestations
Not established by this review	A run of that workflow, published image digests, registry contents, signed source tag, permanent evidence catalog, source-archive checksum, or private SDK packages

The repository and workflows were inspected at source revision `97be86e9efedf07ecf1783b03800f683f107fb04`. No release job, registry, service, or remote publication was executed or queried.

Inventory the actual 1.0.0 source release

The source release is identified by Git objects, not by a container or package manifest.

Artifact	Exact identity or state
Annotated tag	`v1.0.0`; tag object `270ffb55d6f239c42d3487a6842a011d4a7f8d2a`
Commit	`034a520608eea44e6b14e61c34734bd402d989c0`
Tree	`0784fbb30a9795e3098eb2498126d9ddccde243d`
Source snapshot	`SOURCE_SNAPSHOT.json`; SHA-256 `45e905fbc83e710e8a7261ccb8f940f92ad083197430a10cabf44e922b74bcd2`
Changelog	`CHANGELOG`; release date 21 July 2026
Native schemas	`52` files under `schemas/aip`
Legal set	`LICENSE`, `NOTICE`, and `THIRD_PARTY_NOTICES`
Publication shape	Private clean-root source snapshot with one Markdown file

The tag object is annotated but contains no embedded cryptographic signature. Git object IDs detect content changes inside a trusted object database; they do not authenticate the publisher by themselves.

The tag contains neither `.github/workflows/release.yml` nor `private-sdk-release.json`. It records no canonical source-archive checksum, image digest manifest, SBOM, provenance statement, Cosign verification record, or registry package identity. Do not attach artifacts from the post-tag workflow to `v1.0.0` retrospectively.

Verify the local source identity

Run these read-only checks in a trusted clone that contains the tag:

```
SH
test "$(git rev-parse v1.0.0^{tag})" = \
  "270ffb55d6f239c42d3487a6842a011d4a7f8d2a"
test "$(git rev-parse v1.0.0^{commit})" = \
  "034a520608eea44e6b14e61c34734bd402d989c0"
test "$(git rev-parse v1.0.0^{tree})" = \
  "0784fbb30a9795e3098eb2498126d9ddccde243d"
git show v1.0.0:SOURCE_SNAPSHOT.json | sha256sum
```

The final digest must be `45e905fbc83e710e8a7261ccb8f940f92ad083197430a10cabf44e922b74bcd2`. Inspect the JSON and confirm its source revision, source tree, file count, export method, normalization, license, and publication boundary before using the source.

If an organization creates a source archive from the tag, record the archive format, command, tool version, and SHA-256 as a new packaging artifact. The repository does not provide a canonical archive digest to compare against.

Inventory the prospective image set

The pinned `immutable-release` workflow defines thirteen image jobs. They are source intent for a future tag, not evidence that the images exist.

Group	Images defined by the workflow	Count
Platform and operator	Core daemon, migration bundle, connector control plane, and `aipctl`	4
Public connector hosts	Cal.diy, Hermes Agent, Chatwoot, Dify, CrewAI, and Twenty hosts	6
Product runtime	CrewAI Python sidecar	1
Controlled qualification	Two validation-only host images, not public product connectors	2
Total	All image matrix entries	13

Every job targets `linux/amd64` and `linux/arm64`. The registry coordinate is derived as `ghcr.io//`. The workflow creates a version tag and a long source-SHA tag; it does not create `latest`.

The six public host images correspond exactly to the maintained connector set. The controlled qualification images must not appear in the public connector catalog or be mistaken for supported products.

Understand the release gate

Images depend on a source release gate that requires all of the following:

1. The ref name is `v<workspace-version>`.
2. The checked-out commit equals the event SHA and the worktree is clean.
3. Publication hygiene and the complete source release check pass.
4. Generated native schemas leave no diff.
5. Private-SDK and connector-repository boundaries are valid.
6. Deterministic inventories for both repository boundaries are generated.

The source-boundary inventories are uploaded as one Actions artifact retained for 90 days. A separate release-check workflow also runs the complete source gate and history secret scan, but the image workflow repeats the release gate before building images.

Manual dispatch does not bypass tag identity: the same ref-name and workspace version checks still apply.

Understand each image result

Before building, the job rejects an existing remote version tag. A successful job then:

1. builds one multi-platform image from the exact tagged checkout;
2. writes OCI source, revision, and version labels;
3. requests a BuildKit SBOM and maximum-mode provenance;
4. pushes the version and long-SHA tags and captures the multi-platform digest;
5. signs that digest keylessly with GitHub OIDC through Cosign;
6. verifies the signature against the exact workflow identity and GitHub token

issuer;

7. publishes a GitHub build-provenance attestation to the registry;
8. writes `<image>.digest` and `<image>.cosign-verification.json`;
9. uploads both files as an Actions artifact retained for 90 days.

The workflow has no final job that combines all thirteen digests into one signed release manifest. It also does not create standalone binary archives, a source archive, a package-registry release, or a release-page attachment.

Verify a published image result

Obtain the digest from the retained release evidence, not from a mutable image tag. Set the exact image, tag, and digest before verification:

```
SH
TAG="v1.1.0"
IMAGE="ghcr.io/getaip/aipd-core"
DIGEST="sha256:<recorded-multi-platform-digest>"
IDENTITY="https://github.com/getaip/core/.github/workflows/release.yml@refs/tags/${TAG}"

docker buildx imagetools inspect "${IMAGE}@${DIGEST}"
cosign verify \
  --certificate-identity "${IDENTITY}" \
  --certificate-oidc-issuer "https://token.actions.githubusercontent.com" \
  --output json \
  "${IMAGE}@${DIGEST}"
```

Do not accept a successful tag lookup as digest verification. Compare the resolved index digest with the retained `.digest` file, then verify:

- both required platform manifests;
- the OCI source, revision, and version labels;
- the keyless signature identity and issuer;
- the BuildKit SBOM and provenance linked to the exact subject digest;
- the GitHub build-provenance attestation subject and source revision;
- the corresponding source-boundary inventories and clean tagged tree.

Store the verification output with the release record. A later registry lookup does not replace the evidence captured when the release was accepted.

Promote ephemeral evidence to a durable catalog

The workflow's inventory, digest, and Cosign verification artifacts expire after 90 days. General Gitea CI retains boundary and qualification artifacts for 30 days; those CI runs are not substitutes for a tagged release record.

Before accepting a future release, assemble a durable, access-controlled catalog containing:

- tag object, commit, tree, workspace version, and source-snapshot identity;
- source-boundary inventory JSON and its checksum;
- one row for every expected image with name, platforms, digest, and labels;
- Cosign verification JSON, SBOM, provenance, and build attestation for each

image digest;

- release-gate and secret-scan run identities and terminal states;
- license and third-party notice checksums;
- applicable conformance and qualification evidence identities;
- missing, failed, skipped, superseded, or revoked artifact records.

Sign or otherwise integrity-protect the catalog itself. A directory of unsigned digest files cannot prove that the expected set is complete.

Keep private SDK publication separate

The pinned source defines an inventory order for 24 private SDK packages and a 12-package connector repository boundary. It does not enable publication:

```
TEXT
publication_mode = disabled_until_registry_bootstrap
minimum_verified_private_sdk_releases = 2
```

All reviewed workspace packages remain `publish = false`. Generated inventory JSON proves a selected source boundary; it is not a package archive, registry entry, signature, or successful private SDK release.

Accept or reject an artifact set

Accept a future immutable-image release only when:

- the semantic version tag, commit, tree, and workspace version agree;
- the source gate and boundary inventories are complete;
- all thirteen expected workflow jobs have exact digests;
- every accepted image has both required platforms, correct labels, verified

signature, SBOM, provenance, and build attestation;

- the durable catalog covers the complete set and survives CI retention;
- conformance and product qualification claims cite their own exact artifacts;
- no evidence is borrowed from `v1.0.0` or another source revision.

Otherwise report the artifact set as incomplete or unverified. Do not replace a missing signature, digest, platform, inventory, or qualification result with the statement that its workflow is present.

Related documentation

- Release notes
- Licensing
- Implementation status

- Testing and qualification
- Production deployment