Dify connector changelog

Use this page to decide whether a Dify connector change requires a new artifact, admission record, migration, caller update, or qualification result. It records only changes established from the reviewed Git range.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/changelog.md

> **Release status:** the reviewed snapshot is not a tagged release. The repository tags `034a5206` as `v1.0.0`; the documentation revision is five commits later and still declares workspace version `1.0.0`.

Use full source, artifact, manifest, and provider identities to distinguish these states.

Identify the exact version

Identity	Published `v1.0.0` baseline	Reviewed source snapshot
Source commit	`034a520608eea44e6b14e61c34734bd402d989c0`	`97be86e9efedf07ecf1783b03800f683f107fb04`
Source position	Tagged	`v1.0.0-5-g97be86e`
Publication status	Published tag in reviewed repository	Unreleased source in reviewed ancestry
Workspace version	`1.0.0`	`1.0.0`
Native AIP version	`1.0`	`1.0`
Dify surface	One alias per configured app	Alias plus mode-projected app operations and Knowledge operations
Credential model	One connector-wide app key	Per-app keys and independent Knowledge keys, with legacy app-key option
Target placement	Bundled library connector	Separately built and admitted standalone host

An operator record for the reviewed snapshot needs the full source commit, tree, immutable image digest, manifest digest, connector version ID, release ID, descriptor hashes, and provider identity.

The string `1.0.0` cannot replace that identity tuple.

Reviewed source snapshot

The entries below describe Dify-relevant differences between tagged `v1.0.0` and `97be86e9`. They are source changes, not evidence that an artifact was built, published, deployed, or qualified.

Added

- A frozen catalogue of 79 Dify operations pinned to upstream commit `f8d47616c15d0959f604f6a5e2e1d32d3108991b`.

- Thirty-three app operation templates with mode-dependent projection.
- Forty-six workspace-scoped Knowledge operation templates.
- Per-app and per-Knowledge credential bindings.
- Bounded query, JSON, multipart, binary, and response handling.
- Typed Dify failure codes with retry and uncertainty metadata.
- Composite health across every configured credential.
- Cluster-safe task correlation through runtime profile state.
- A dedicated `aip-host-dify` binary and common standalone image path.
- Registry admission, tenant routing, leases, durable runtime storage, and

recovery integration.

- Standalone image and controlled product-fleet qualification procedures.

Changed

App aliases remain under `cap:dify:`. Each app now also publishes the operation suffixes permitted by its configured mode.

Mode	App operations	Alias	Total per app
`workflow`	23	1	24
`completion`	19	1	20
`chat`	27	1	28
`advanced-chat`	29	1	30
`agent-chat`	27	1	28
`agent`	27	1	28

Every configured Knowledge credential publishes 46 capabilities under `cap:dify:knowledge::`.

The manifest is now descriptor-dependent. Callers must discover capability IDs instead of assuming the tagged baseline's one-capability-per-app surface.

Credential isolation

The tagged connector stored one shared API key. The reviewed connector stores separate app keys and separate Knowledge keys.

The standalone host accepts a legacy global app key only when every app intentionally shares it. It cannot be mixed with per-app files and cannot authorize Knowledge operations.

This is a deployment and secret-ownership migration. It does not change the public identity of an existing app alias by itself.

Transport and response bounds

The reviewed provider client requires an HTTPS origin outside loopback, rejects credentials and path material, disables authenticated redirects, and uses bounded connection and request behavior.

It enforces query, JSON, upload, response, SSE frame, and event limits. Catalogue response kinds select JSON, normalized SSE, or bounded base64 binary output.

All non-GET catalogue operations require approval and an AIP idempotency key. Only GET operations publish safe retry support.

Streaming and cancellation durability

App aliases retain blocking and streaming execution. The reviewed connector adds explicit SSE operations and normalizes event kinds, terminal results, human-input pauses, and provider failures.

Remote task identity is stored without destructive read. Cancellation can survive restart through instance-scoped profile state in the standalone host.

Missing task identity remains an unconfirmed remote stop. Cancelling an event subscription closes only the local stream and never infers workflow stop.

Standalone security boundary

The reviewed topology fixes provider origin, descriptors, key files, connector identity, artifact, manifest, tenant, lifecycle route, and credential revision outside Action input.

These controls describe reviewed source. They do not qualify an arbitrary Dify deployment or make direct provider calls part of AIP.

Deprecated, removed, or renamed

No Dify capability is declared deprecated, removed, or renamed in the reviewed range. The operation suffixes and Knowledge domain are additions.

Future provider aliases or route removals must not be inferred from upstream names. Record them only when reviewed connector source or release policy defines the change and migration.

Apply the required migration

Change encountered	Required action	Do not do
Bundled connector to standalone host	Preserve tenant, app, key, Action, task, approval, and idempotency evidence through the fleet migration	Run both mutation paths without a fence
Same version string on a new snapshot	Admit a new immutable connector version with exact source, image, manifest, and release identities	Overwrite an existing record or identify it only as `1.0.0`
Alias-only manifest expands	Refresh discovery, policy, generated clients, expected counts, and catalogue digest	Construct suffixes from provider route names
Shared app key becomes scoped	Create owner-only per-app files and bind each descriptor explicitly	Copy one key into unrelated app or Knowledge scopes
Knowledge is enabled	Add independent workspace descriptors, keys, policy, and 46-capability admission per credential	Reuse an application key
Durable cancellation is enabled	Provision coherent host storage and preserve instance-scoped profile state	Use process memory as restart-safe evidence
Response treatment changes	Revalidate bounds, SSE terminal behavior, binary consumers, and uncertainty handling	Infer treatment only from `Content-Type`
Upstream revision changes	Re-audit methods, paths, schemas, mode support, and exclusions, then requalify	Promote new provider bytes under the old compatibility claim

Stabilize standalone identity and credential isolation before adopting new Knowledge mutations or a provider upgrade.

Retain compatibility pins

Surface	Reviewed pin
Connector ID	`dify`
Connector profile	`aip.connector.dify.v1`
Manifest format	`aip-manifest/v1`
Workspace and host version	`1.0.0`
Native AIP version	`1.0`
Dify upstream commit	`f8d47616c15d0959f604f6a5e2e1d32d3108991b`
Frozen operation templates	33 app and 46 Knowledge
Default provider-response limit	8 MiB
Maximum configurable response limit	64 MiB
Standalone host	`aip-host-dify`
Standalone version-ID shape	`cver_dify_1_0_0_<release-id>`

The upstream commit identifies reviewed source, not a built Dify image or deployment. Changing provider bytes does not establish compatibility with the new artifact.

Decide when a new connector version is required

Create and admit a new immutable connector version when a change affects:

- capability IDs, projection, schemas, contracts, kind, risk, approval, retry,

idempotency, or compensation;

- provider method, path, query, body, multipart, response, event, or status

mapping;

- descriptor semantics, app mode, credential scope, secret handling, URL

policy, health probe, or request bounds;

- task-store namespace, task identity, cancellation certainty, stream

terminal behavior, or cleanup;

- artifact bytes, dependencies, manifest digest, upstream baseline, host

lifecycle, or storage behavior.

A documentation clarification alone does not require a connector version. It still needs the source revision that proves the corrected statement.

Record future entries completely

Each future entry should state:

1. release or explicit unreleased status and UTC review date;
2. full source commit, tree, and reconstructable dirty state when applicable;
3. connector, image, manifest, catalogue, release, descriptor, and upstream

identities;

4. added, changed, fixed, security-relevant, deprecated, and removed surfaces;

5. caller, operator, storage, credential, and provider migration actions;
6. rollback and compatibility boundary;
7. exact connector, image, fleet, and live qualification result, or `not_run`;
8. replacement and removal date only when source defines them.

Do not reconstruct release history from filenames, mutable tags, or source presence. Mark unknown history as unknown.

Related documentation

- AIP release notes
- Migrate bundled connectors to the fleet
- Dify connector configuration
- Dify capability index
- Qualify the Dify connector