Dify authentication and credential scopes

The Dify connector crosses several identity boundaries in one Action. Keep each identity in its source-owned layer so a caller cannot select a provider credential, another workspace, or another end user's conversation context.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/getting-started/authentication-and-credential-scopes.md

This page explains the boundaries and rotation decisions. It does not grant a provider scope or replace the deployment's tenant and secret policy.

Separate the four identity layers

Layer	Trusted source	Purpose	Must not select it
AIP caller	Verified signed envelope and membership	Authorizes the tenant Action	Untrusted request metadata
Connector host	Admitted instance, replica, artifact, manifest, and route	Fixes the executable and tenant boundary	Action input
Dify credential	Owner-controlled host configuration and key file	Authenticates one app or Knowledge domain	Action body, query, or file data
Dify end user	Validated Action field mapped by the chosen capability	Isolates user-bound provider data	Connector-wide fallback or bearer token

The signed AIP route selects an admitted host. The capability ID then selects one descriptor already loaded by that host. Only after both checks does the connector choose a Bearer key from its protected map.

Scope application keys per app

A Dify Service API key is application-scoped. The standalone host supports two configuration forms:

Form	Intended use	Constraint
Per-app `api_key_file`	Normal production isolation	Every app names the file for its own key
Global `AIP_DIFY_API_KEY_FILE`	Intentional shared-key compatibility	Every configured app omits `api_key_file` and truly shares that key

The host rejects an app that combines a per-app file with the global key. It also rejects an app with neither source. A global key without at least one app is invalid.

Prefer one file and one provider key per app. Shared material enlarges the blast radius of rotation, provider compromise, and descriptor misbinding. Never copy an app key into an Action, descriptor description, log, image, manifest, or public configuration file.

Scope Knowledge keys per workspace boundary

Every Knowledge descriptor requires its own `api_key_file`. The connector uses that key only for capabilities under:

```
TEXT
cap:dify:knowledge:<credential_id>:<operation>
```

The local `credential_id` is a public routing label, not a provider secret. It must map to the intended Dify workspace and remain stable for the lifetime of an admitted connector version.

An application key and a Knowledge key are not interchangeable. A successful app health probe does not prove Knowledge access, and a dataset read does not prove app access.

Keep provider selection outside the Action

The host accepts one Dify base origin and static descriptor sets at startup. Action input cannot override:

- provider scheme, host, port, or path;
- app ID or Knowledge credential ID outside the capability;
- key file or Bearer value;
- connector instance, tenant, artifact, manifest, or credential revision.

The base URL must be an origin without credentials, path prefix, query, or fragment. HTTPS is required except for explicit loopback. Redirects are disabled, so a provider cannot move a signed request and its Bearer key to a different origin through HTTP redirection.

Preserve Dify end-user identity

App invocation aliases require `user`. Catalogue operations require it only when their frozen route declares a query, JSON, or multipart user location. The value must contain 1 to 256 bytes without control characters.

Use a stable, opaque, non-secret value derived under tenant policy. It should not reveal an email address, phone number, credential, or internal database key.

The connector rejects a `user` that conflicts with another input partition. It cannot determine whether two valid values represent different people. Reusing one value across tenants or users can merge Dify conversations, messages, feedback, files, or cancellation authority.

Knowledge operations do not inject an end-user value. Their provider boundary is the selected workspace key, while AIP tenant policy still governs who may invoke the capability.

Understand credential metadata

Discovery exposes only the credential class and public descriptor ID:

Surface	Auth metadata
App invocation alias	`dify_app_api_key`
App catalogue operation	`dify_app_api_key` plus app descriptor ID
Knowledge operation	`dify_knowledge_api_key` plus Knowledge descriptor ID

The connector's debug formatter replaces both key maps with `[REDACTED]`. That reduces accidental diagnostics exposure, but it does not sanitize every external logger, crash dump, process inspection surface, or provider response.

The connector publishes `credentials: false` in implementation support. It does not accept AIP credential handles or dynamically resolve caller-selected secrets at this revision.

Provision a credential safely

Before admitting a host:

1. create a least-authority provider key for the exact app or Knowledge

workspace;

2. write it to a dedicated owner-only file of at most 16 KiB;
3. keep descriptor JSON public only if it contains no secret material;
4. bind each descriptor ID to the reviewed key file and provider account;
5. bind the host to one AIP tenant and immutable connector identity;
6. admit the exact artifact and manifest before starting Action traffic;
7. verify `/ready` and one safe read through the normal gateway route;
8. record the credential revision without recording the key.

Do not test a key by printing it or placing it on a command line. Use the configured host and a non-mutating capability whose provider scope is known.

Rotate without widening authority

Keys are loaded when the host process is constructed. Rotation therefore uses a controlled replacement:

1. pause new unsafe Actions for the affected descriptor;
2. identify in-flight Actions and remote Dify tasks;
3. create a new provider key with the same reviewed scope;
4. write a new owner-only file and update the intended credential revision;
5. admit the replacement version, then start its host with immutable metadata;
6. require readiness and one gateway-routed read for the affected credential;
7. drain the old replica while preserving task correlation and Action state;
8. revoke the old provider key only after traffic and recovery ownership move;
9. retain redacted evidence and remove the old file under secret policy.

Do not rotate an app key, Knowledge key, tenant binding, descriptor ID, provider origin, and connector version in one opaque change. Separate evidence makes a failed boundary and rollback target identifiable.

Respond to credential incidents

Treat unexpected account data, cross-user conversations, provider `401` or `403` after a known-good state, descriptor drift, or key exposure as an authentication or isolation incident.

Contain the affected descriptor without disabling unrelated credentials. Keep Action IDs, capability IDs, tenant, instance, credential revision, remote status, and redacted provider account evidence. Do not retain Bearer values, raw personal content, or secret-file bytes in the incident bundle.

After correction, require AIP signature and route verification, host readiness, the exact credential's safe health path, and provider-state reconciliation for every uncertain mutation. A green process `/health` response is insufficient.

Related documentation

- Dify connector overview
- Run the Dify quickstart
- Identity and trust

- Connector credentials and rotation
- Errors and retry decisions