Dify app metadata and files

Use this family to inspect a published application's contract, read one Dify end user, upload one bounded user file, or retrieve one bounded file preview. All seven operations are available for every supported app mode.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/capabilities/app-metadata-and-files.md

Choose the configured app through the capability ID. The host supplies its app-specific Bearer key and fixed provider origin.

Choose an operation

Operation suffix	Method and route	Required input	Result	Policy
`parameters.get`	`GET /v1/parameters`	None	JSON	Low-risk retry-supported read
`meta.get`	`GET /v1/meta`	None	JSON	Low-risk retry-supported read
`info.get`	`GET /v1/info`	None	JSON	Low-risk retry-supported read
`site.get`	`GET /v1/site`	None	JSON	Low-risk retry-supported read
`file.upload`	`POST /v1/files/upload`	`user` and `file`	JSON	Medium mutation; approval and key required
`file.preview`	`GET /v1/files/{file_id}/preview`	`file_id` and `user`	Binary	Low-risk retry-supported read
`end_user.get`	`GET /v1/end-users/{end_user_id}`	`end_user_id`	JSON	Low-risk retry-supported read

The capability template is:

```text
TEXT
cap:dify:<app_id>:<operation>
```

Use discovery to obtain the exact app ID. Do not insert a provider key, origin, or unrelated app ID into the Action.

Partition input correctly

Every operation accepts the generic frozen input object:

Field	Use in this family
Path fields	`file_id` or `end_user_id` where the route requires them
`user`	Required for upload and preview
`query`	Optional provider query; preview also receives the validated `user`
`body`	Not used by the upload multipart builder; optional on other routes only when provider-compatible
`file`	Required only for upload

Path values contain 1 to 512 bytes. `user` contains 1 to 256 bytes without control characters. Query objects have at most 128 properties and arrays at most 256 scalar values.

The connector appends the required preview `user` itself and rejects a conflicting query value.

Read app metadata

Use the four metadata reads to answer different questions:

Read	Use it to inspect
`parameters.get`	Published application input parameters
`meta.get`	Tool and model metadata
`info.get`	Published application information
`site.get`	Published site settings

All four send an authenticated empty-path read to the selected app. They do not require Dify end-user identity and do not prove that model execution, file access, or another app key works.

Each result has the generic JSON shape:

```
JSON
{
  "http_status": 200,
  "body": {},
  "content_type": "application/json",
  "provider_request_id": null
}
```

The provider owns the nested `body` schema. Treat unknown fields as compatible provider data unless the selected task requires a stronger reviewed contract.

Read one end user

Select `end_user.get` and supply only the provider's end-user identifier:

```
JSON
{
  "end_user_id": "end-user-identifier"
}
```

This route does not use the caller's `user` field. Authorization comes from the selected app key and AIP tenant policy. A returned record can contain personal data, so apply the connector's confidential and redaction-required contract.

An absent or wrong identifier is a read failure. It is not evidence that the same person has no conversations, messages, files, or state under another app.

Upload one file

Before upload:

1. select the intended app and stable tenant-scoped end-user value;
2. confirm the file is authorized for that user and Dify application;
3. calculate the decoded size and intended content type;
4. create one Action ID and one idempotency key;
5. obtain the required tenant-policy approval.

The Action input has this shape:

```
JSON
{
  "user": "tenant-user-0190",
  "file": {
    "filename": "context.txt",
    "content_type": "text/plain",
    "content_base64": "ZXhhbXBsZQ=="
  }
}
```

The connector validates a safe basename, base64, content type, and decoded size. It sends multipart fields named `file` and `user`. For this operation it does not serialize Action `body` as a multipart `data` field.

The decoded-file implementation limit is 32 MiB. The base64 schema and common native envelope can impose lower effective limits. Do not raise a response limit to work around an oversized request.

Upload is a provider mutation. It requires approval and a tenant-scoped key, publishes no automatic retry, and has no rollback capability. After an interrupted response, inspect provider and Action state before another upload.

Preview one message file

Supply the file and the same stable Dify user identity that owns access:

```
JSON
{
  "file_id": "file-identifier",
  "user": "tenant-user-0190"
}
```

The connector renders `file_id` in the path and `user` in the query. It returns bounded bytes as base64:

```
JSON
{
  "http_status": 200,
  "content_base64": "ZXhhbXBsZQ==",
  "content_type": "text/plain",
  "size_bytes": 7,
  "provider_request_id": null
}
```

Decode only after authorization, content-type policy, malware controls, and output-size checks. The connector does not write the result to a caller-selected path.

Recover safely

Observation	Decision
Metadata read is unauthorized	Verify the selected app descriptor and key without printing it
Preview is unauthorized	Verify both app scope and exact end-user identity
Path field is rejected	Supply one non-empty identifier within 512 bytes
File basename or base64 is rejected	Correct the input before provider dispatch
Upload remains pending approval	Review and decide the same immutable Action
Upload response is temporary, transport-failed, or oversized	Treat provider outcome as uncertain; inspect file state
Binary result exceeds the response limit	Review expected file size and policy; do not retry blindly
Returned data belongs to another user or app	Stop traffic and treat it as an isolation incident

For reads, retry support still requires the same Action policy, budget, and verified error. For upload, neither a key nor approval authorizes automatic replay.

Related documentation

- Dify capability index

- Dify connector configuration
- Authentication and credential scopes
- Dify connector overview
- Errors and retry decisions