Dify chat messages, conversations, and feedback

Use this family to read chat history, suggestions, conversations, feedback, and conversation variables. It also supports feedback, rename, variable-update, and conversation-deletion mutations under AIP approval and replay controls.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/capabilities/chat-messages-conversations-and-feedback.md

Most operations require a chat-like Dify application and one stable end-user identity. Application-wide feedback is the exception. It has no end-user input and is projected for every configured app mode.

Choose an operation

Operation suffix	Method and route	Identity input	Connector projection	Policy
`message.list`	`GET /v1/messages`	`user` in query	Four chat-like modes	Low-risk read; retry supported
`message.feedback.create`	`POST /v1/messages/{message_id}/feedbacks`	`user` in JSON	Four chat-like modes	Medium mutation; approval and key required
`feedback.list`	`GET /v1/app/feedbacks`	None	All six app modes	Low-risk read; retry supported
`message.suggested.get`	`GET /v1/messages/{message_id}/suggested`	`user` in query	Four chat-like modes	Low-risk read; retry supported
`conversation.list`	`GET /v1/conversations`	`user` in query	Four chat-like modes	Low-risk read; retry supported
`conversation.delete`	`DELETE /v1/conversations/{conversation_id}`	`user` in JSON	Four chat-like modes	High mutation; approval and key required
`conversation.rename`	`POST /v1/conversations/{conversation_id}/name`	`user` in JSON	Four chat-like modes	Medium mutation; approval and key required
`conversation.variable.list`	`GET /v1/conversations/{conversation_id}/variables`	`user` in query	See mode boundary below	Low-risk read; retry supported
`conversation.variable.update`	`PUT /v1/conversations/{conversation_id}/variables/{variable_id}`	`user` in JSON	See mode boundary below	Medium mutation; approval and key required

The four chat-like connector modes are `chat`, `advanced-chat`, `agent-chat`, and `agent`. At the pinned Dify revision, conversation-variable routes accept only the first three. Connector discovery still projects both variable operations for `agent`, so do not invoke them for that mode.

The capability template is:

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

Obtain the exact capability ID from tenant discovery. A capability exposed for one app does not authorize access to another app or another end user.

Partition the Action input

Input field	Connector behavior
`message_id`	Renders one message route segment
`conversation_id`	Renders one conversation route segment or enters a provider query
`variable_id`	Renders one conversation-variable route segment
`user`	Enters the query for user-scoped reads and JSON for mutations
`query`	Supplies bounded provider query fields
`body`	Supplies bounded provider JSON fields

Each rendered path value contains 1 to 512 bytes without control characters. The connector accepts a bounded string, while the pinned provider routes parse message, conversation, and variable path values as UUIDs.

`user` contains 1 to 256 bytes without control characters. The connector inserts it into the required query or body location. A conflicting provider `user` value is rejected before dispatch.

`query` has at most 128 keys. Each name contains 1 to 256 bytes, and each value is a scalar or an array of at most 256 scalars. `body` has at most 512 properties, and its encoded JSON cannot exceed 8 MiB.

Read messages and suggestions

List one user's messages by selecting `message.list` and passing the provider conversation ID in `query`:

```
JSON
{
  "user": "tenant-user-0190",
  "query": {
    "conversation_id": "0190c42f-2d5a-7000-8000-000000000101",
    "limit": 20
  }
}
```

The pinned provider requires `conversation_id`. `first_id` is an optional message cursor, and `limit` ranges from 1 to 100 with a default of 20. The first page returns the latest messages; the next request uses the first retained message ID to fetch older records.

Use `message.suggested.get` with a UUID `message_id` and the same `user` value. The provider can reject the read when the message is absent or suggested questions are disabled.

Read application feedback

`feedback.list` reads feedback across the selected application rather than one end user. Its provider query accepts `page` from 1 upward and `limit` from 1 to

101. Both default to 1 and 20 respectively.

The returned records can identify conversations, messages, end users, or accounts. Treat the result as confidential PII even though the request has no `user` field.

Read conversations and variables

`conversation.list` accepts optional `last_id`, `limit`, and `sort_by` query fields. `limit` ranges from 1 to 100 and defaults to 20. `sort_by` is one of `created_at`, `-created_at`, `updated_at`, or `-updated_at`.

`conversation.variable.list` requires a UUID `conversation_id`. Its query can carry `last_id`, a limit from 1 to 100, and `variable_name`. The name contains 1 to 255 safe alphanumeric, hyphen, underscore, or period characters.

Conversation and variable reads are end-user scoped. A valid app key does not permit one `user` value to read another user's conversation state.

Apply the four mutations

Every mutation requires tenant-policy approval and one non-blank Action idempotency key of at most 512 bytes. The connector sends the key to Dify and retains a tenant-scoped AIP reservation for 24 hours. This does not prove that the provider deduplicates a repeated external effect.

Submit or replace message feedback with `message.feedback.create`:

```
JSON
{
  "message_id": "0190c42f-2d5a-7000-8000-000000000102",
  "user": "tenant-user-0190",
  "body": {
    "rating": "like",
    "content": "The answer resolved the task."
  }
}
```

The pinned provider accepts `like`, `dislike`, or `null` for `rating`; `null` revokes earlier feedback. `content` is optional.

The other mutation bodies are:

Operation	Required provider body
`conversation.delete`	No field beyond the connector-inserted `user`
`conversation.rename`	Non-blank `name`, or `auto_generate: true`
`conversation.variable.update`	`value` matching the variable's declared type

Delete is irreversible through this connector. Rename and variable update also declare no rollback capability. Verify the exact conversation, user, and variable before approval.

Interpret the result

A successful JSON operation completes with the generic connector envelope:

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

The nested `body` follows the selected pinned Dify route. A non-success provider status becomes a typed connector error instead of a completed envelope.

All nine operations are synchronous and do not publish streaming or capability-specific cancellation. Results, query values, feedback, messages, and variables carry confidential data with PII and redaction requirements.

Recover safely

Observation	Decision
Read input is rejected before dispatch	Correct the path, user, query, or body under the same task decision
Variable operation on `agent` is rejected	Do not retry; use a supported app mode or remove the unsupported call
Message or conversation is absent	Re-establish the exact app, user, and UUID before another request
Read fails with a verified retryable error	Retry only within the declared Action budget
Mutation remains pending approval	Decide the same immutable Action and input
Mutation fails before provider dispatch	Correct the rejected input without claiming an external effect
Mutation response is lost or transport-ambiguous	Treat provider outcome as uncertain and inspect authoritative Dify state
Delete outcome cannot be established	Escalate; do not create a replacement delete Action
Returned data crosses an app or user boundary	Stop traffic and treat the event as an isolation incident

None of the four mutations publishes safe retry support. Preserve the original Action ID, input, key, approval, provider request identity, and final read used to settle the outcome.

Related documentation

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