Run Dify chat and workflows

Follow this guide to invoke one configured Dify app through its discovery-derived alias. The procedure applies to chat, completion, workflow, advanced-chat, agent-chat, and agent modes.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/guides/run-chat-and-workflows.md

The task is complete only after AIP records a terminal result or a documented human-input state. A provider request acknowledgement, task ID, or accepted cancel command is not equivalent to completed work.

Before you begin

Obtain these values from the authorized tenant context:

- one discovered `cap:dify:<app_id>` alias;
- its configured mode and admitted connector release;
- one stable Dify `user` mapping for the end user;
- the exact app input contract and any approved files;
- a unique Action ID and non-blank idempotency key;
- approval evidence when connector or deployment policy requires it.

Do not construct the capability from a provider URL. Discovery binds the app ID to a specific credential without exposing its Bearer value.

1. Select the app mode

The alias maps the configured mode to one provider route:

Configured mode	Provider route	Required invocation input	Connector approval
`workflow`	`POST /v1/workflows/run`	`user`, `inputs`	Required
`completion`	`POST /v1/completion-messages`	`user`, `query` or `prompt`	Not required by alias contract
`chat`	`POST /v1/chat-messages`	`user`, `query` or `prompt`	Not required by alias contract
`advanced-chat`	`POST /v1/chat-messages`	`user`, `query` or `prompt`	Required
`agent-chat`	`POST /v1/chat-messages`	`user`, `query` or `prompt`	Required
`agent`	`POST /v1/chat-messages`	`user`, `query` or `prompt`	Required

Deployment policy can require approval for plain chat or completion even when the alias contract does not. Never infer authorization from the mode alone.

All aliases are medium-risk, require an idempotency key, publish no safe retry, and declare rollback unsupported.

2. Bind app and end-user identity

Use one stable, non-secret `user` value for every related invocation, history read, stop, event resume, and form submission. It contains 1 to 256 bytes without control characters.

Do not reuse a shared placeholder across customers. Dify uses this value to isolate conversations, messages, files, and task control within the selected app.

Record the tenant, capability, app ID, mode, user mapping, Action ID, and input digest before dispatch. The Action cannot select a different provider origin or credential.

3. Freeze approval and idempotency

Create one immutable Action input and one tenant-scoped idempotency key. AIP retains the key reservation for 24 hours and revalidates the input hash on collision.

For approval-required modes, evidence includes the reason, input snapshot, and policy decision. The connector approval window is 900,000 milliseconds.

The AIP reservation prevents conflicting reuse inside its scope. It does not prove that Dify deduplicates a repeated model, workflow, tool, or agent effect.

4. Build a chat or completion input

Use `query` or `prompt` for the user message. The connector trims the selected value and rejects an empty message.

```
JSON
{
  "query": "Summarize the approved refund policy.",
  "inputs": {
    "region": "emea"
  },
  "user": "tenant-user-0190",
  "files": [],
  "auto_generate_name": false
}
```

Chat-like modes can also send `conversation_id`, `files`, and `auto_generate_name`. Completion accepts files but does not send conversation or automatic-name fields.

`query` and `prompt` each allow at most 1 MiB. `files` allows at most 100 objects. The complete encoded provider JSON cannot exceed 8 MiB.

Use only file references or remote sources already authorized for the same app and end user. Uploading bytes and invoking the app are separate effects.

5. Build a workflow input

Workflow invocation sends `inputs`, connector-selected `response_mode`, and the stable `user`. It does not require a query.

```
JSON
{
  "inputs": {
    "topic": "quarterly forecast",
    "region": "emea"
  },
  "user": "tenant-user-0190"
}
```

Validate every input name and value against the published workflow contract. The alias invokes the app's current published workflow; it does not select a specific workflow version ID.

Use the dedicated run-by-ID capability only when policy and discovery require a pinned published version.

6. Choose blocking or streaming execution

The alias contract supports synchronous, asynchronous, streaming, and cancellation surfaces. The connector, not Action input, controls Dify `response_mode`.

AIP execution path	Provider mode	Observable result
Blocking invocation	`blocking`	One completed or `requires_human` result
Typed streaming execution	`streaming`	Bounded AIP chunks plus one terminal result
Durable asynchronous caller flow	Runtime-managed	Poll the original Action; do not create a replay

Caller-supplied `response_mode` is not part of the alias input schema. Do not add it to the Action or assume it overrides the execution path.

Streaming limits each SSE frame to 1 MiB, parsed events to 10,000, and total bytes to the configured response limit. The default total limit is 8 MiB.

7. Interpret streaming and terminal state

The connector maps provider events to AIP stream chunks:

Provider event	AIP meaning
`workflow_started`, `node_started`	Progress
`tool_call`	Tool activity
`agent_thought`	Agent reasoning metadata
`message`	Data
`human_input_required`, `workflow_paused`	Pending approval or input
`error`	Failed result
`message_end`, `workflow_finished`	Terminal result

The connector assembles text from answer-bearing events. A terminal output can contain `answer` plus `final_event`, or only the final event data.

An upstream close without a terminal event is an error. A non-succeeded workflow finish is failed even when the HTTP stream began successfully.

8. Continue chat or answer human input

For a later chat turn, use the provider `conversation_id` returned by the first turn and preserve the same user:

```
JSON
{
  "query": "Apply that policy to order 1042.",
  "inputs": {
    "region": "emea"
  },
  "user": "tenant-user-0190",
  "conversation_id": "0190c42f-2d5a-7000-8000-000000000901"
}
```

Do not attach a conversation from another app or user. Confirm continuity through a conversation read when identity or outcome is uncertain.

A `human_input_required` or `workflow_paused` event produces an AIP `requires_human` result. Preserve the form token and read the form through its dedicated capability.

Submit one approved form response through a new Action. Forms are one-shot, so never replay the original app invocation to answer a pause.

9. Cancel with evidence

During streaming, the connector records a remote task only after Dify emits a task ID. Cancellation after that point uses the mode-specific stop route and the same app and user.

Cancellation before a task ID arrives can close the local request but cannot confirm a remote stop. Record `remote_stop_confirmed: false` and inspect the provider's authoritative run or conversation state.

A stop response means the provider accepted task control. Retain terminal event, workflow-run, or conversation evidence before reporting the business operation as stopped.

10. Settle uncertain outcomes

Observation	Decision
Input is rejected before dispatch	Correct the same reviewed intent without claiming a provider effect
Action remains pending approval	Decide the immutable Action; do not submit a replacement
Blocking response is lost	Read authoritative conversation or workflow state before another invocation
Stream closes before terminal evidence	Treat completion as unknown and inspect the provider run
Human-input state is returned	Preserve the form token and submit one separately approved response
Cancellation lacks a task ID	Record unconfirmed remote stop and inspect provider state
Terminal provider event reports failure	Keep the failed Action and repair the cause before a new intent
Result crosses an app or user boundary	Stop traffic and treat the event as an isolation incident

Never automatically retry an alias invocation. A new Action is appropriate only after authoritative evidence shows that a new intent is required.

Completion checklist

- The capability came from authenticated tenant discovery.
- App mode, user mapping, input, approval, and idempotency key were frozen.
- Files and remote sources belonged to the same authorized task.
- Blocking or streaming behavior matched the selected AIP execution path.
- A terminal, failed, or `requires_human` state was retained.
- Cancellation and replay claims match available provider evidence.
- Sensitive inputs, outputs, events, and errors were redacted from general logs.

Related documentation

- Dify capability index
- Dify app metadata and files
- Dify chat messages, conversations, and feedback
- Dify workflows, streaming, and human input
- Errors and retry decisions