Dify workflows, streaming, and human input

Use this family to inspect workflow runs and logs, execute a published workflow version, stop active generation, resume workflow events, and answer a paused human-input form. The nine operations use one configured Dify app credential.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/capabilities/workflows-streaming-and-human-input.md

Task control is not proof of terminal provider state. Preserve the app, end user, task, workflow run, and form identities until a read or terminal event establishes the outcome.

Choose an operation

Operation suffix	Method and route	Connector modes	Result	Policy
`completion.stop`	`POST /v1/completion-messages/{task_id}/stop`	`completion`	JSON	Medium mutation; approval and key required
`chat.stop`	`POST /v1/chat-messages/{task_id}/stop`	Four chat-like modes	JSON	Medium mutation; approval and key required
`workflow.run.get`	`GET /v1/workflows/run/{workflow_run_id}`	`workflow`, `advanced-chat`	JSON	Low-risk read; retry supported
`workflow.run_by_id`	`POST /v1/workflows/{workflow_id}/run`	`workflow`	SSE or blocking JSON	High mutation; approval and key required
`workflow.stop`	`POST /v1/workflows/tasks/{task_id}/stop`	`workflow`	JSON	Medium mutation; approval and key required
`workflow.log.list`	`GET /v1/workflows/logs`	`workflow`, `advanced-chat`	JSON	Low-risk read; retry supported
`workflow.event.stream`	`GET /v1/workflow/{task_id}/events`	`workflow`	SSE	Low-risk read; retry supported
`human_input_form.get`	`GET /v1/form/human_input/{form_token}`	All six app modes	JSON	Low-risk read; retry supported
`human_input_form.submit`	`POST /v1/form/human_input/{form_token}`	All six app modes	JSON	Medium mutation; approval and key required

The four chat-like modes are `chat`, `advanced-chat`, `agent-chat`, and `agent`. Connector discovery exposes `workflow.event.stream` only for `workflow`, although the pinned provider route also accepts `advanced-chat`.

The capability template is:

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

Use tenant discovery to obtain the exact capability ID. A task ID, workflow run ID, workflow version ID, or form token does not select a different app.

Partition the Action input

Field	Use in this family
`task_id`	Renders a stop or event-stream route segment
`workflow_run_id`	Renders the workflow-run detail route segment
`workflow_id`	Renders the pinned workflow-version route segment
`form_token`	Renders a human-input form route segment
`user`	Enters JSON for stop, run, and form submission, or query for event resume
`query`	Supplies workflow-log filters or event-resume controls
`body`	Supplies workflow inputs or human-input form values

Each rendered path value contains 1 to 512 bytes without control characters. The connector percent-encodes each value as one path segment.

`user` contains 1 to 256 bytes without control characters. The connector inserts it into the required location and rejects a conflicting query or body value.

`query` has at most 128 keys and at most 256 scalar values per array. `body` has at most 512 properties, and its encoded JSON cannot exceed 8 MiB.

Inspect workflow runs and logs

Use `workflow.run.get` with the provider workflow-run ID:

```
JSON
{
  "workflow_run_id": "0190c42f-2d5a-7000-8000-000000000201"
}
```

The result can include `id`, `workflow_id`, `status`, `inputs`, `outputs`, `error`, step and token totals, timestamps, and elapsed time. A paused run returns an empty `outputs` object until execution continues.

This read is app-scoped, not end-user scoped. The pinned route checks the app and tenant but has no `user` input. Treat its inputs, outputs, and errors as confidential PII-bearing data.

`workflow.log.list` accepts these optional query fields:

Field	Constraint
`keyword`	Provider log search term
`status`	`succeeded`, `failed`, or `stopped`
`created_at__before`	ISO 8601 timestamp
`created_at__after`	ISO 8601 timestamp
`created_by_end_user_session_id`	End-user session filter
`created_by_account`	Account ID filter
`page`	1 to 99,999; default 1
`limit`	1 to 100; default 20

Log records can include workflow-run status, errors, creator details, and execution details. This operation is also app-scoped and can expose data from multiple end users under that app.

Run one published workflow version

Select `workflow.run_by_id` when the task requires a specific published workflow version rather than the app's default invocation alias:

```
JSON
{
  "workflow_id": "0190c42f-2d5a-7000-8000-000000000202",
  "user": "tenant-user-0190",
  "body": {
    "inputs": {
      "topic": "quarterly forecast"
    },
    "files": []
  }
}
```

The pinned provider requires `inputs`; `files` is optional. A file input uses either a provider upload ID or an authorized remote URL according to the workflow's published parameter definition.

The connector controls `response_mode`. Its streaming execution path requests SSE, while its synchronous path requests a blocking response. A caller-supplied `response_mode` does not override that decision.

This operation invokes models and changes provider execution state. It is high-risk, requires tenant-policy approval and an idempotency key, publishes no safe retry, and has no rollback capability.

Stop an active task

All three stop operations require the task ID and the same stable Dify user used to start generation:

```
JSON
{
  "task_id": "task-0190c42f",
  "user": "tenant-user-0190"
}
```

Choose the stop suffix that matches the configured app mode. The connector does not infer the task family from the task ID.

The pinned completion and chat routes pass the user identity to Dify task control. The pinned workflow route requires a user field but sends its stop commands without checking task ownership against that user. Approval must therefore confirm the exact app and workflow task.

A successful stop response means Dify accepted the command. Confirm the terminal state through the workflow-run read or a terminal stream event before releasing task correlation data.

Resume workflow events

Use `workflow.event.stream` after a dropped connection or a human-input pause:

```
JSON
{
  "task_id": "0190c42f-2d5a-7000-8000-000000000203",
  "user": "tenant-user-0190",
  "query": {
    "include_state_snapshot": true,
    "continue_on_pause": true
  }
}
```

The provider verifies the app, tenant, end-user creator role, and exact end user. A finished run emits one `workflow_finished` event and closes.

`include_state_snapshot` replays a persisted status summary before new events. `continue_on_pause` keeps the stream open across consecutive `workflow_paused` events. Both fields default to `false`.

The connector limits each SSE frame to 1 MiB, the stream to 10,000 parsed events, and the total response to the configured response bound. A stream that closes without a terminal event is a connector error, not successful completion.

Read and submit a human-input form

`human_input_form.get` requires only `form_token`. The pinned provider returns the form only when it belongs to the selected app and tenant and uses an allowed Service API recipient type.

The response contains `form_content`, input definitions, `resolved_default_values`, `user_actions`, and an optional expiration time. The form token normally comes from a `human_input_required` event.

Submit the selected action and values with `human_input_form.submit`:

```
JSON
{
  "form_token": "form-token-0190",
  "user": "tenant-user-0190",
  "body": {
    "inputs": {
      "decision": "approve"
    },
    "action": "continue"
  }
}
```

The `action` value must match an ID from `user_actions`. Each `inputs` key must match the form definition. File values use Dify's local-upload or remote-URL form mapping.

Forms are one-shot. The first accepted response wins, regardless of which end user submits it, and an expired form cannot resume the workflow. Submission therefore requires approval, one immutable input, and no automatic replay.

Interpret results and cancellation

Synchronous JSON operations complete with the generic connector envelope:

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

For streaming execution, the connector emits each parsed provider event as an AIP stream chunk. The terminal Action output contains the final event, or an assembled answer with that event.

`human_input_required` and `workflow_paused` produce an AIP `requires_human` result. An `error` event, or a non-succeeded `workflow_finished` event, produces a failed Action result.

During streaming, the connector records a provider task ID only after Dify emits it. Cancellation before that point drops the request but cannot confirm a remote stop. Later cancellation sends the mode-specific stop request.

Recover safely

Observation	Decision
Path, user, query, or body is rejected before dispatch	Correct the same bounded request without claiming provider state
Workflow version is missing or still a draft	Select a published version; do not retry the rejected ID
Run or log read is unauthorized	Re-establish the app credential and tenant route without printing secrets
Event resume cannot match the end user	Recover the original task and user mapping before another stream
SSE closes before a terminal event	Treat completion as unknown and inspect the workflow run
Cancellation occurs before any task ID arrives	Record that remote stop is unconfirmed and inspect provider state
Stop returns success but no terminal evidence exists	Poll the run or resume events; do not report completion
Form is expired or already submitted	Do not replay; inspect workflow state and preserve the first-response evidence
Run, stop, or form submission has an ambiguous transport outcome	Treat provider state as uncertain and reconcile through authoritative reads
Returned run or log data crosses the intended app or tenant	Stop traffic and treat the event as an isolation incident

All five POST operations require approval and a non-blank Action idempotency key of at most 512 bytes. The AIP reservation is tenant-scoped for 24 hours, but it does not prove provider deduplication.

Related documentation

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