Manage Dify knowledge

Use this procedure to create or select one Dify knowledge base, ingest an approved source, wait for indexing, enrich its records, and verify retrieval. An optional final branch updates or removes provider state.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/dify/guides/manage-knowledge.md

Each mutation uses its own Action ID, immutable input, approval, and idempotency key. A lost response never authorizes a replacement Action.

Before you begin

Obtain:

- one authenticated tenant and admitted Dify connector release;
- one discovered Knowledge capability prefix for the intended workspace;
- authority to read model, dataset, document, and metadata inventory;
- approved source content and its retention classification;
- an approver for every non-GET operation;
- a protected location for Action, provider, batch, and readback evidence.

The Knowledge prefix is:

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

Do not use an app alias or app API key. The configured Knowledge credential selects the workspace and never belongs in Action input.

1. Pin capability and workspace identity

Read authenticated tenant discovery and record the exact Knowledge prefix, connector revision, credential ID, and required operations. Stop if any capability is absent.

Record the credential revision without copying secret bytes. Dataset, document, segment, metadata, upload, and model IDs remain valid only inside that workspace boundary.

For every mutation, create one Action ID and one tenant-scoped idempotency key before approval. Keep the same Action, key, and canonical input until its outcome is settled.

2. Discover models and existing datasets

Read available embedding or reranking models with `model.list`. Then call `dataset.list` with a narrow keyword or tag filter.

```
JSON
{
  "query": {
    "page": 1,
    "limit": 20,
    "keyword": "customer support"
  }
}
```

Use an existing dataset only after `dataset.get` confirms its workspace, provider, permission, indexing technique, models, and retrieval settings. Name similarity alone is insufficient.

Model discovery proves current visibility, not quota or future availability. Retain the exact provider and model identifiers selected for indexing.

3. Create a dataset only when needed

Skip this step when an approved dataset already exists.

Prepare the smallest creation body that fixes ownership and indexing intent:

```
JSON
{
  "body": {
    "name": "Customer support knowledge",
    "description": "Reviewed support policy and product facts.",
    "provider": "vendor",
    "permission": "partial_members",
    "indexing_technique": "high_quality",
    "embedding_model_provider": "provider-name",
    "embedding_model": "embedding-model-name"
  }
}
```

Approve the exact name, permission, provider class, models, retrieval settings, and summary configuration. Successful creation returns an empty dataset, not indexed knowledge.

If the response is lost, list by the exact name and compare full settings. Do not create a second dataset merely because the first Action is uncertain.

4. Choose one ingestion branch

Choose exactly one branch for each approved source:

Source	Initial capability	Completion evidence
Authorized raw text	`document.text.create`	Returned document and batch, then terminal indexing status
Authorized local file	`document.file.create`	Returned document and batch, then terminal indexing status
Pipeline local file	`pipeline.file.upload`, then `pipeline.run`	Upload ID, run result, then authoritative document reads
Pipeline remote source	`pipeline.run`	Run result, then authoritative document reads

Direct file ingestion is unavailable for an external dataset. Pipeline upload creates a workspace file first; it does not attach that file to a dataset by itself.

Do not combine branches to recover a lost response. Inspect provider state and the retained IDs from the original branch.

5. Ingest text or a file directly

For text, supply name, content, document form, language, and the intended indexing technique:

```
JSON
{
  "dataset_id": "0190c42f-2d5a-7000-8000-000000001001",
  "body": {
    "name": "Refund policy",
    "text": "Approved customers may request a refund within thirty days.",
    "doc_form": "text_model",
    "doc_language": "English",
    "indexing_technique": "high_quality"
  }
}
```

For a file, provide one safe filename, MIME type, base64 content, and serialized processing settings. The connector caps decoded input at 32 MiB; provider type, quota, and upload limits can be stricter.

The first document needs an indexing technique when the dataset does not have one. Later documents inherit the dataset setting.

Both operations return `document` and `batch`. Persist both immediately. Their HTTP completion acknowledges ingestion but does not complete indexing.

6. Run a knowledge pipeline when selected

For a local pipeline source, upload the file and retain its returned ID. Use that ID as `reference` in `datasource_info_list`.

Run the published pipeline in blocking mode for a structured connector result:

```
JSON
{
  "dataset_id": "0190c42f-2d5a-7000-8000-000000001001",
  "body": {
    "inputs": {},
    "datasource_type": "local_file",
    "datasource_info_list": [
      {
        "reference": "upload-0190",
        "name": "refund-policy.pdf"
      }
    ],
    "start_node_id": "datasource-node-1",
    "is_published": true,
    "response_mode": "blocking"
  }
}
```

Datasource discovery and execution must use the same published or draft selection. Verify every remote URL, workspace, page, drive, bucket, plugin, and credential before approval.

After the run, list and read generated documents. Provider pipeline completion does not replace document and indexing evidence.

7. Settle indexing

For direct ingestion, poll `document.indexing_status` with the returned batch and same dataset:

```
JSON
{
  "dataset_id": "0190c42f-2d5a-7000-8000-000000001001",
  "batch": "batch-0190c42f"
}
```

Normal progress is `waiting`, `parsing`, `cleaning`, `splitting`, `indexing`, then `completed`. Preserve paused, stopped, or error evidence instead of reporting success.

Require a terminal state for every intended document. The connector exposes no provider indexing cancellation operation.

Read each document after completion. Confirm dataset, source type, name, display status, form, segment count, metadata, error, and summary state needed by the task.

8. Inspect chunks before enrichment

Use `segment.list` to inspect representative content, answers, keywords, attachments, summaries, enabled state, and errors. Use `child_chunk.list` when the document uses a parent-child structure.

Do not modify generated chunks only to make a failed ingestion appear complete. Repair the source or processing configuration under a new approved intent.

A segment update can return HTTP success with `data.status: error` after an indexing failure. Inspect returned status and perform a follow-up read.

9. Apply tags and metadata

Use tags for workspace organization and metadata for structured document attributes. Discover current identities before writing either.

For document metadata, use the exact field ID, name, and type returned by `metadata.list`. Set `partial_update: true` to preserve unspecified custom values.

```
JSON
{
  "dataset_id": "0190c42f-2d5a-7000-8000-000000001001",
  "body": {
    "operation_data": [
      {
        "document_id": "0190c42f-2d5a-7000-8000-000000001002",
        "metadata_list": [
          {
            "id": "0190c42f-2d5a-7000-8000-000000001003",
            "name": "approved_region",
            "value": "emea"
          }
        ],
        "partial_update": true
      }
    ]
  }
}
```

The default false replaces custom metadata and bindings. Multi-document input commits each document separately, so a later failure can leave earlier entries changed.

Read fields, tags, and every intended document after mutation. Provider success alone is insufficient for metadata helpers that can suppress internal errors.

10. Verify governed retrieval

`dataset.retrieve` performs a read-oriented provider search, but the connector classifies its POST route as a medium mutation. It requires approval and an idempotency key and publishes no safe retry.

Use one representative query of at most 250 characters. Bind approval to the dataset, query, model settings, attachments, expected result count, and data handling.

Verify that returned segments belong to the intended dataset and documents. Record scores and fields needed for acceptance without retaining unrelated content.

11. Maintain or remove state

Use partial dataset update for approved configuration changes. Re-read the complete dataset after changing permissions, models, retrieval, or external knowledge settings.

Document replacement returns a new batch and must repeat indexing settlement. Status actions require readback of every changed document.

Delete only after dependencies and retention duties are cleared. Document delete removes all chunks and is blocked during indexing or while archived. Dataset delete removes all documents and is blocked while an app uses it.

Both deletes are high risk, unsafe to retry, and have no rollback. Confirm absence through authoritative reads after settlement.

Recover uncertain work

Observation	Read first	Decision
Dataset create response lost	Dataset list and matching detail	Reuse the confirmed dataset or escalate
Direct ingestion response lost	Document list, source identity, and recent batches	Do not ingest the source again blindly
Pipeline response lost	Upload inventory, pipeline evidence, and document list	Reconcile generated state before another run
Indexing pauses or fails	Batch status and document detail	Repair the cause under a new approved intent
Metadata batch fails partway	Every intended document	Reconcile only the divergent entries
Retrieval response is lost	Original Action state and provider evidence	Do not repeat automatically
Delete response is lost	Exact dataset or document read	Accept confirmed absence or escalate
Data crosses a workspace boundary	Credential, route chain, and audit evidence	Stop traffic and treat it as an isolation incident

Completion checklist

- Capability and credential identity came from authenticated discovery.
- Model and dataset settings matched the reviewed intent.
- Exactly one ingestion branch owned each source.
- Document and batch identities were retained before polling.
- Every intended document reached a verified terminal indexing state.
- Tags, metadata, and retrieval results passed authoritative readback.
- Each mutation kept its original Action, input, approval, and key.
- Sensitive content and identifiers were redacted from general logs.
- Any destructive change passed dependency and retention checks.

Related documentation

- Dify knowledge bases, tags, and retrieval
- Dify knowledge documents and indexing
- Dify segments and child chunks
- Dify models and knowledge metadata
- Dify knowledge pipelines and uploads