Run the Twenty connector quickstart

In this tutorial, you will read one Twenty collection and create one record through an already admitted connector. You will then read the original AIP Action until its governed mutation reaches a terminal state.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/twenty/getting-started/quickstart.md

The example uses a standard `companies` object. Confirm that this object and its `name` field exist in the admitted workspace before submitting the create.

This procedure does not deploy Twenty or the connector. The reviewed fleet Compose file has no Twenty service, so the tutorial starts from a deployment whose operator has already admitted and connected the host.

What you will verify

By the end, you will have observed:

- two admitted fixed capabilities rather than an arbitrary provider route;
- a read scoped to the host's configured workspace and credential;
- one create held for tenant-policy approval;
- a stable idempotency key bound to the create input;
- a connector-generated deterministic record ID when the input omits `id`;
- the same queued Action resuming after approval;
- a durable terminal result containing the provider HTTP status and body.

You will not test webhook delivery, metadata mutation, batch operations, deletion, restore, merge, failover, or a connector deployment.

Prerequisites

You need:

- a reachable native AIP endpoint;
- an admitted Twenty host bound to the intended workspace;
- an AIP caller credential authorized for discovery and invocation;
- a trusted tenant approver for mutations;
- permission to create the named fixture in that workspace;
- a durable place to retain Action, approval, and idempotency identities;
- `aipctl` built from the reviewed AIP revision.

Set only AIP client coordinates in the shell:

```
SH
export AIP_URL='https://aip.example.com'
export AIP_TOKEN_FILE='/run/secrets/aip-native-token'
```

The operator configures the Twenty origin, workspace UUID, and API token on the host. Do not put the provider token or workspace selector in Action input.

1. Confirm the admitted contracts

Discover the two capabilities used by this tutorial:

```
SH
aipctl \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  capability list "$AIP_URL" \
  --capability-id cap:twenty:record.list \
  --limit 1

aipctl \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  capability list "$AIP_URL" \
  --capability-id cap:twenty:record.create \
  --limit 1
```

Record the catalog revision plus both contract and schema digests. Confirm that `record.list` is a retry-safe read. Confirm that `record.create` requires approval and external-account idempotency with a seven-day contract TTL.

Catalog visibility proves admission at one revision. It does not prove current provider reachability or that the workspace still exposes the example object.

2. Read a bounded collection

Write `companies-list.json`:

```
JSON
{
  "object": "companies",
  "query": {
    "limit": 1
  }
}
```

Submit a synchronous read with a stable Action ID:

```
SH
export LIST_ACTION_ID='act_twenty_quickstart_list_001'

aipctl \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action call "$AIP_URL" \
  cap:twenty:record.list \
  --action-id "$LIST_ACTION_ID" \
  --mode sync \
  --input @companies-list.json
```

A completed result contains `http_status`, an optional `provider_request_id`, and the provider `body`. The connector does not reshape the collection into an AIP-specific record model.

Stop if this read reports authentication, object, or schema failure. Correct the deployment or select a workspace-owned object before attempting a write.

3. Freeze one create intent

Write `company-create.json` once:

```
JSON
{
  "object": "companies",
  "body": {
```

```
    "name": "AIP Quickstart Company"
  }
}
```

The connector accepts standard and custom object names that match its bounded identifier grammar. Twenty still decides whether the selected object and body match the current workspace schema.

Reserve stable identities for this exact input:

```
SH
export CREATE_ACTION_ID='act_twenty_quickstart_create_001'
export CREATE_KEY='twenty-quickstart-company-v1'
```

Keep the Action ID, key, object, and body unchanged. Reusing the key with different canonical input is an idempotency collision, not a new create.

4. Submit the governed create

Submit the mutation once:

```
SH
aipctl \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action call "$AIP_URL" \
  cap:twenty:record.create \
  --action-id "$CREATE_ACTION_ID" \
  --idempotency-key "$CREATE_KEY" \
  --mode sync \
  --input @company-create.json
```

Without an admitted approval decision, the Action enters `requires_human` and returns an approval identity. Store that identity and leave the original Action in place.

The connector forces provider `upsert=true`. When the body omits `id`, it derives one UUID from the configured workspace, idempotency key, and record index. This makes a repeated create target the same provider record.

Deterministic targeting limits duplicate risk. It does not authorize the mutation or prove that Twenty committed the upsert.

5. Complete trusted approval

Use the deployment's approver-facing process to inspect the frozen request and submit a decision from an authenticated tenant-policy authority. The decision must match the approval ID and policy hash and provide the required reason, input snapshot, and policy-decision evidence.

The Twenty approval validity window is 900,000 ms. Approval resumes the same queued Action. Do not submit a replacement create or change the idempotency key while the original Action remains pending.

Approval authorizes one attempt of the frozen intent. It does not establish provider success, schema compatibility, or safe replay after an uncertain mutation outcome.

6. Read the durable result

Read the original Action independently of its submission connection:

```
SH
aipctl \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action status "$AIP_URL" \
  "$CREATE_ACTION_ID" \
  --include-result \
  --include-receipts \
  --wait-ms 30000
```

Proceed only after a terminal `completed` result. Verify a successful `http_status`, retain `provider_request_id` when present, and inspect the returned body for the deterministic record ID and intended name.

Treat a nonterminal Action as still owned by the original workflow. Treat an uncertain mutation failure as a reconciliation case; changing the Action ID or key can create an additional provider effect.

7. Confirm idempotent observation

Repeat the exact create request with the same Action ID, key, and bytes:

```
SH
aipctl \
  --native-bearer-token-file "$AIP_TOKEN_FILE" \
  action call "$AIP_URL" \
  cap:twenty:record.create \
  --action-id "$CREATE_ACTION_ID" \
  --idempotency-key "$CREATE_KEY" \
  --mode sync \
  --input @company-create.json
```

Expected result: the runtime returns the durable outcome for the same AIP intent. This check is not permission to send a new Action with a new key.

Recover without duplicating the write

Observation	Response
Capability is absent	Stop; the operation is not admitted for this route
Read returns authentication failure	Repair host credential or workspace binding; do not add credentials to input
Create is `requires_human`	Complete the existing approval workflow
Create remains pending	Read the same Action; do not replace it
`connector.twenty.invalid_request`	Correct object, body, identifier, or key under a new governed intent
`connector.twenty.remote_rejected`	Inspect redacted provider details and current workspace schema
Temporary read failure	Retry the read according to its contract
Uncertain mutation failure	Preserve the original Action and key; reconcile before any new write

The connector has no cancellation, transaction, compensation, or automatic provider-reconciliation operation. A caller cannot turn an uncertain create into a safe retry by replacing its identity.

Verify the tutorial outcome

Retain and compare:

- the catalog revision plus both contract and schema digests;
- the list Action and its bounded input;
- the immutable create input, Action ID, and idempotency key;
- the approval ID, policy hash, decision, and required evidence;
- the terminal Action result and receipts;
- the provider request ID and returned record ID when present.

These artifacts demonstrate one governed application path. They do not by themselves establish live-provider qualification, production readiness, webhook delivery, external notification, or compatibility with another Twenty revision.

Related documentation

- Twenty connector overview
- AIP quickstart
- Command-line interface
- Approvals and policy
- Errors and retry decisions