Twenty record query and aggregation

Use these four read operations to inspect records, fetch one stable ID, detect duplicate candidates, or request provider grouping. They accept standard and custom object names through fixed Twenty REST routes.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/twenty/capabilities/record-query-and-aggregation.md

The connector validates the outer input and query allowlist. Twenty owns field names, filter grammar, ordering syntax, aggregation semantics, and the returned record shape for the selected workspace revision.

Compare the operations

Operation	Provider request	Required input	Primary result
`record.list`	`GET /rest/{object}`	`object`	Provider collection and pagination data
`record.get`	`GET /rest/{object}/{id}`	`object`, `id`	One provider record
`record.find_duplicates`	`POST /rest/{object}/duplicates`	`object`, `body`	Provider duplicate candidates
`record.group_by`	`GET /rest/{object}/groupBy`	`object`	Provider groups and aggregates

All four are low-risk reads. They require no approval and accept an optional idempotency key. They advertise synchronous execution and safe connector retry, with no asynchronous, streaming, or cancellation support.

The POST method used by duplicate detection does not make it a mutation in the reviewed AIP contract.

Use a bounded object name

Every input requires `object` matching:

```
TEXT
^[A-Za-z][A-Za-z0-9_]{0,127}$
```

This admits standard and custom identifiers without allowing slashes, dots, hyphens, percent escapes, or caller-selected paths. It does not prove that the object exists in the configured workspace.

Single-record IDs and duplicate-ID candidates must be canonical RFC 4122 UUIDs with version 1 through 5 and an RFC variant nibble.

List records

`cap:twenty:record.list` accepts this input shape:

```
JSON
{
  "object": "companies",
  "query": {
    "limit": 25,
    "order_by": "createdAt[DESC]",
```

```
    "depth": 0,
    "filter": "name[like]:AIP%"
  }
}
```

Allowed query keys are:

Key	Connector schema
`limit`	Integer from `0` through `200`
`order_by`	String, array, or object
`depth`	Integer `0` or `1`
`filter`	String, array, or object
`starting_after`	RFC 4122 UUID
`ending_before`	RFC 4122 UUID

The connector does not require a limit or cursor. Applications should choose a bounded limit and retain provider pagination state rather than assuming one response is complete.

Get one record

`cap:twenty:record.get` requires an object and ID:

```
JSON
{
  "object": "companies",
  "id": "0190c42f-2d5a-7000-8000-000000000010",
  "query": {
    "depth": 1
  }
}
```

`depth` is the only accepted query key. Omit it when nested relationships are not needed. The connector sends the configured workspace credential and does not add a workspace selector to the Action.

Find duplicate candidates

`cap:twenty:record.find_duplicates` accepts exactly one of two body forms.

Compare one through 200 candidate objects:

```
JSON
{
  "object": "companies",
  "body": {
    "data": [
      {
        "name": "AIP Quickstart Company"
      }
    ]
  },
  "query": {
    "depth": 0
  }
}
```

Or compare one through 200 existing IDs:

```
JSON
{
  "object": "companies",
  "body": {
    "ids": [
      "0190c42f-2d5a-7000-8000-000000000010"
    ]
  }
```

```
}
```

The body cannot contain both alternatives, neither alternative, or additional top-level fields. Each `data` member must be an object, and each `ids` member must pass the connector's UUID validation.

The connector treats this provider POST as read-only. Provider behavior still belongs to the pinned mapping and selected workspace deployment.

Group records

`cap:twenty:record.group_by` accepts a bounded provider grouping request:

```
JSON
{
  "object": "companies",
  "query": {
    "limit": 20,
    "filter": "employees[gte]:10",
    "group_by": "address.city",
    "aggregate": "count",
    "include_records_sample": false
  }
}
```

Allowed query keys are:

Key	Connector schema
`limit`	Integer from `0` through `200`
`order_by`	String, array, or object
`filter`	String, array, or object
`group_by`	String, array, or object
`include_records_sample`	Boolean
`order_by_for_records`	String, array, or object
`aggregate`	String, array, or object
`viewId`	RFC 4122 UUID

The connector forwards allowed values; it does not interpret provider field paths or prove that a named view belongs to the selected object.

Understand query encoding and bounds

The connector appends every allowed query member as one URL query pair.

- strings are forwarded as strings;
- booleans and numbers use their JSON scalar text;
- arrays and objects use compact JSON text;
- `null` is rejected;
- one encoded value cannot exceed 32 KiB or contain NUL;
- the complete encoded URL cannot exceed 128 KiB;
- no query object can exceed the operation's fixed property set.

The common append layer has a 64-entry ceiling, while these four schemas admit at most eight named keys. Unknown keys fail before provider dispatch.

Structured query values preserve their JSON shape as encoded text. The connector does not transform them into provider filter or order expressions.

Interpret results

A completed result has this connector-owned wrapper:

```
JSON
{
  "http_status": 200,
  "provider_request_id": "provider-request-id-or-null",
  "body": {}
}
```

`body` is the decoded provider JSON without record-shape normalization. An empty response becomes `null`. A non-empty response that is invalid JSON or exceeds the configured response bound fails as an invalid provider response.

`provider_request_id` comes from `x-request-id`, then `traceparent`, when the header is valid, contains no control character, and is at most 512 bytes.

Handle failures and retry

Condition	Connector code	Retry decision
Invalid object, UUID, body, query key, value, or bound	`connector.twenty.invalid_request`	Correct input
Provider `401` or `403`	`connector.twenty.authentication`	Repair fixed host credential or workspace binding
Provider `429` or `5xx`	`connector.twenty.remote_temporary`	Retry the same read according to policy
Other provider non-success	`connector.twenty.remote_rejected`	Correct the provider-facing request or deployment
Read transport failure	`connector.twenty.transport`	Retryable for these operations
Oversized or invalid JSON response	`connector.twenty.invalid_response`	Not automatically retryable

Provider error projection retains only bounded `statusCode`, `error`, `message`, `messages`, and `code` fields. Do not expect the complete provider error body in AIP evidence.

Retry the same read intent with a stable Action identity. A retryable flag does not prove that provider state, schema, or pagination remained unchanged between attempts.

Security and data boundaries

All four contracts classify data as confidential, possibly containing personal information, with redaction required. Inputs and outputs can expose customer, relationship, view, or custom-field data even though the operations are reads.

The connector fixes origin, workspace, token, and route templates. Query values cannot select another host, add a path segment, or replace the provider credential.

Retain only the fields needed for the application task. Apply tenant access, log redaction, pagination bounds, and evidence retention controls outside the connector.

Related documentation

- Twenty capability index
- Twenty connector quickstart
- Twenty connector configuration

- Errors and retry decisions
- Capabilities and contracts