Twenty record delete, restore, and merge

Use these seven mutations to change record lifecycle or consolidate duplicate records. Choose the narrowest operation, freeze its exact target, and preserve the original Action and idempotency identity through verification.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/twenty/capabilities/record-delete-restore-and-merge.md

Soft delete, permanent destroy, restore, and merge have different provider consequences. The connector does not expose transactions, compensation, rollback, reconciliation, cancellation, or safe retry for any of them.

Compare the operations

Operation	Provider request	Risk	Target
`record.soft_delete`	`DELETE /rest/{object}/{id}?soft_delete=true`	High	One ID
`record.soft_delete_many`	`DELETE /rest/{object}?filter=...&soft_delete=true`	High	Explicit filter
`record.destroy`	`DELETE /rest/{object}/{id}?soft_delete=false`	Critical	One ID
`record.destroy_many`	`DELETE /rest/{object}?filter=...&soft_delete=false`	Critical	Explicit filter
`record.restore`	`PATCH /rest/restore/{object}?filter=id[eq]:{id}`	High	One ID through filter
`record.restore_many`	`PATCH /rest/restore/{object}?filter=...`	High	Explicit filter
`record.merge`	`PATCH /rest/{object}/merge`	High	Two through nine IDs

Every operation requires tenant-policy approval and an external-account-scoped idempotency key. Every operation advertises unsafe retry.

Apply shared target rules

Every operation requires `object` matching:

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

Single-record operations require a canonical RFC 4122 UUID with version 1 through 5 and an RFC variant nibble. Merge applies the same check to each member.

The connector fixes the provider origin, workspace, token, and path template. Object and filter values cannot select a different provider host.

Soft-delete one record

`cap:twenty:record.soft_delete` accepts only object and ID:

```
JSON
{
  "object": "companies",
  "id": "0190c42f-2d5a-7000-8000-000000000040"
}
```

The connector adds `soft_delete=true`. It accepts no query object for this single-target operation.

Soft deletion represents a reversible provider lifecycle state only when the record still exists and the provider permits restore. It is not an AIP undo token or retention guarantee.

Soft-delete a filtered set

`cap:twenty:record.soft_delete_many` requires a non-empty provider filter:

```
JSON
{
  "object": "companies",
  "query": {
    "filter": "id[in]:[0190c42f-2d5a-7000-8000-000000000040]"
  }
}
```

The connector rejects a missing filter, whitespace-only filter, literal `{}`, or literal `null`. It does not interpret the provider filter or count matches.

Preflight the same filter with an authorized read and retain the expected IDs and count in approval evidence. Do not widen the filter after approval.

Permanently destroy one record

`cap:twenty:record.destroy` uses the same outer input as single soft delete:

```
JSON
{
  "object": "companies",
  "id": "0190c42f-2d5a-7000-8000-000000000040"
}
```

The connector sends `soft_delete=false` and classifies the operation as critical. No connector capability restores the permanently destroyed record.

Approval evidence should identify data-retention authority, dependencies, backups, legal holds, and a separately tested recovery plan when one exists. An idempotency key is not a backup.

Permanently destroy a filtered set

`cap:twenty:record.destroy_many` requires an explicit non-empty filter:

```
JSON
{
  "object": "companies",
  "query": {
    "filter": "id[in]:[0190c42f-2d5a-7000-8000-000000000040]"
  }
}
```

The connector adds `soft_delete=false`. The filter guard prevents an empty or obvious whole-collection request, but it does not prove that a syntactically non-empty provider filter is narrow.

Treat filter semantics, selected IDs, count, workspace schema, and permanent consequences as approval inputs. Re-read the exact target set immediately before authorization when deployment policy requires it.

Restore one soft-deleted record

`cap:twenty:record.restore` accepts an object, ID, and optional depth:

```
JSON
{
  "object": "companies",
  "id": "0190c42f-2d5a-7000-8000-000000000040",
  "query": {
    "depth": 0
  }
}
```

The pinned provider source declares a single-record restore route, but its core-path parser rejects that three-segment form. The connector therefore uses the filtered collection restore route with `filter=id[eq]:`.

This compatibility mapping has the intended single-record target. It remains a provider collection request and should be requalified when the pinned provider revision changes.

Restore cannot recover a permanent destroy. It also does not prove that every relationship, automation, or downstream index returns to its prior state.

Restore a filtered set

`cap:twenty:record.restore_many` accepts a non-empty filter and optional depth:

```
JSON
{
  "object": "companies",
  "query": {
    "filter": "id[in]:[0190c42f-2d5a-7000-8000-000000000040]",
    "depth": 0
  }
}
```

The connector applies the same non-empty filter guard used by other bulk mutations. It does not verify that each selected record is soft-deleted or that the restored set matches an earlier delete Action.

Use a separate Action and key for restore. Do not reuse the soft-delete key to represent the inverse intent.

Merge duplicate records

`cap:twenty:record.merge` accepts only `ids`, `conflictPriorityIndex`, and optional `dryRun` inside `body`:

```
JSON
{
  "object": "companies",
  "body": {
    "ids": [
      "0190c42f-2d5a-7000-8000-000000000040",
      "0190c42f-2d5a-7000-8000-000000000041"
    ],
    "conflictPriorityIndex": 0,
    "dryRun": true
  },
  "query": {
    "depth": 1
  }
}
```

The ID array must contain two through nine unique UUIDs. The priority index must be non-negative and smaller than the actual array length. `dryRun`, when present, must be Boolean.

`depth` is the only accepted query key. Additional body keys fail before provider dispatch.

The connector classifies merge as a mutation even when `dryRun` is true. It still requires approval and an idempotency key and still advertises unsafe retry. The provider owns dry-run response semantics.

Apply approval and idempotency

Every operation requires a tenant-policy decision with a reason, input snapshot, and policy decision. The approval validity window is 900,000 ms.

Mutation keys must contain 1 through 1,024 ASCII graphic bytes. The contract scope is the external account, collision behavior revalidates input hash, and the published TTL is 604,800,000 ms.

Bind approval and key to the exact:

- capability and object;
- ID, ordered merge IDs, or encoded filter;
- merge priority and dry-run value;
- configured external account and workspace;
- expected target count and provider schema evidence;
- Action input hash and policy hash.

Approval does not make destruction reversible. Idempotency does not turn an unsafe operation into an automatically retryable one.

Interpret results

A completed result contains:

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

The connector passes through decoded provider JSON. It records the provider-effect-committed checkpoint after a successful response.

Verify actual record state with an authorized read. For bulk operations, compare the observed IDs and count with the approved target set. For merge, verify the surviving record and intended field priority.

Handle failures and unknown outcomes

All seven operations advertise unsafe retry. Provider `429` or `5xx`, an ambiguous transport failure, an invalid or oversized response after dispatch, or runtime cancellation can carry `uncertain_outcome: true`.

Observation	Response
Invalid object, ID, filter, merge body, key, or query	Correct input under a new governed intent
Provider `401` or `403`	Repair fixed credential or workspace binding; do not replay
Definitive provider `4xx`	Correct the rejected intent after checking current state
Temporary or transport failure	Preserve the original Action and key; do not retry automatically
Invalid provider response	Inspect current provider state before any new mutation
Pending or incomplete AIP Action	Read the same Action and receipts

The connector exposes no automatic reconciliation. Use bounded record reads and retained provider request identity to establish the current state before a separate approved recovery action.

Security and data boundaries

All seven contracts classify input and output as confidential, possibly containing personal information, with redaction required. Delete and merge evidence can itself reveal sensitive record identity and retention decisions.

Apply least-privilege provider credentials, field and object policy, target count limits, separation of duties, evidence redaction, legal-retention checks, and post-operation verification outside the connector.

Related documentation

- Twenty capability index
- Record create, update, and batch upsert
- Record query and aggregation
- Approvals and policy
- Errors and retry decisions