Manage Chatwoot contacts and companies

Use this guide to find authoritative customer records, resolve ambiguous matches, attach or detach one contact and company, or merge two contacts. Every mutation ends with a provider read instead of relying on the write response.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/chatwoot/guides/contact-and-company-workflows.md

The safest path is discovery before mutation. Search results identify candidates; they do not prove that two records represent the same person or organization.

When to use this guide

Use this procedure for one bounded customer-data intent:

- find a contact or company without changing provider state;
- attach one known contact to one known company;
- detach one known contact from one known company;
- merge one reviewed contact into another.

Use the complete capability reference for import, export, notes, labels, inbox associations, CRUD, and other operations.

This guide does not define Chatwoot search fields, relationship body fields, or merge participants. Those fields belong to the pinned provider route, not the generic connector schema.

Prerequisites

Before starting, obtain:

- an authenticated AIP caller bound to the intended tenant;
- an admitted Chatwoot connector instance for the intended account;
- the exact pinned Chatwoot request schema for any body-bearing mutation;
- authority to read restricted customer data;
- an approver for every POST, PATCH, or DELETE operation;
- a protected evidence location for identifiers, inputs, and results.

Do not put `account_id` in Action input. The connector host inserts its configured account into every provider route.

Choose the narrowest capability

Intent	Capability	Provider method	Risk	Approval
Search contacts	`cap:chatwoot:contact.search`	GET	Low	No
Apply a provider filter	`cap:chatwoot:contact.filter`	POST	Medium	Required
Read one contact	`cap:chatwoot:contact.get`	GET	Low	No
Search companies	`cap:chatwoot:company.search`	GET	Low	No
Read one company	`cap:chatwoot:company.get`	GET	Low	No
List company contacts	`cap:chatwoot:company.contact.list`	GET	Low	No
Attach a contact	`cap:chatwoot:company.contact.create`	POST	Medium	Required
Detach a contact	`cap:chatwoot:company.contact.delete`	DELETE	High	Required
Merge contacts	`cap:chatwoot:contact.merge`	POST	Medium	Required

`contact.filter` has a read-like name but a POST provider method. The connector therefore publishes it as a governed mutation with unsafe retry. Prefer the GET search or list surface when it can answer the task.

1. Pin route and Action identity

Read capability discovery for the authenticated tenant. Record:

- connector type, immutable version, instance, and configured account;
- exact capability ID and catalogue revision;
- authenticated actor and tenant;
- credential revision without credential bytes;
- policy version used for any approval.

Stop if the route, account, or capability differs from the reviewed intent. Connector admission, capability discovery, actor authorization, and customer identity are separate decisions.

Create a new Action ID for each read. For a mutation, assign one Action ID and one external-account-scoped idempotency key before approval. Keep the same Action, key, and canonical input while settling an uncertain result.

2. Find candidate records

Use `contact.search` or `company.search` first. Both are low-risk GET capabilities and advertise safe retry.

The common input envelope can carry a `query` object:

```
JSON
{
  "query": {
    "...": "fields from the pinned Chatwoot search route"
  }
}
```

This is a structural example, not a copy-ready provider query. The connector bounds query size and scalar values but does not freeze provider field names or matching semantics.

Record the exact query and all candidate IDs needed for the decision. Treat names, email addresses, phone numbers, external identifiers, and company attributes as restricted data.

If provider filtering is necessary, review its body as a mutation:

```
JSON
{
  "body": {
    "...": "fields from the pinned Chatwoot filter route"
  }
}
```

Approve the exact filter body and retention purpose. A completed filter result does not turn POST into a retry-safe read.

3. Read authoritative before-state

Read every candidate that could be changed:

- `contact.get` for each source, destination, or relationship candidate;
- `company.get` for the target company;
- `company.contact.list` for the company's current relationships.

For a contact merge, also inspect the records and relationships that tenant policy considers material. The catalogue exposes contact conversations, labels, notes, and company relationships, but it does not publish one aggregate merge preview.

Stop when:

- more than one candidate remains plausible;
- source and destination identities are equal or unclear;
- records belong to another configured account;
- current relationships conflict with the requested intent;
- the provider schema revision is unknown;
- approval cannot bind the exact body.

Do not resolve ambiguity by choosing the first search result.

4. Attach a contact to a company

Use `cap:chatwoot:company.contact.create`. The connector builds:

```
TEXT
POST /api/v1/accounts/{configured_account_id}/companies/{company_id}/contacts
```

`company_id` is a top-level Action input. The contact relationship fields belong inside the provider-owned `body`:

```
JSON
{
  "company_id": "88",
  "body": {
    "...": "reviewed fields from the pinned relationship route"
  }
}
```

Before dispatch, bind approval to the company, intended contact, canonical body, account, actor, capability, and policy hash. Keep the exact request with the idempotency record.

After a completed result, call `company.contact.list` and confirm the expected relationship once. Do not infer relationship creation from HTTP success alone.

5. Detach a contact from a company

Use `cap:chatwoot:company.contact.delete` only after the before-state list confirms the exact relationship. Its provider route contains both identities:

```
TEXT
DELETE /api/v1/accounts/{configured_account_id}/companies/{company_id}/contacts/{contact_id}
```

An exact Action input can be submitted as follows:

```
SH
COMPANY_ID=88
CONTACT_ID=205
DETACH_ACTION=act_chatwoot_company_detach_001
DETACH_KEY=chatwoot-company-88-contact-205-detach-v1

aipctl action call \
  "$AIP_URL" \
  cap:chatwoot:company.contact.delete \
  --action-id "$DETACH_ACTION" \
  --idempotency-key "$DETACH_KEY" \
  --input "{\"company_id\":\"$COMPANY_ID\",\"contact_id\":\"$CONTACT_ID\"}"
```

The operation removes the relationship. Its catalogue description does not claim that it deletes the contact record.

This DELETE is high risk, unsafe to retry, and has no rollback. Re-list company contacts after settlement.

6. Merge two contacts

Use `cap:chatwoot:contact.merge` only after a human has reviewed distinct source and destination records. The connector builds:

```
TEXT
POST /api/v1/accounts/{configured_account_id}/actions/contact_merge
```

The route has no contact path parameter. Both identities and any merge options belong to a provider-owned body:

```
JSON
{
  "body": {
    "...": "source, destination, and fields from the pinned merge route"
  }
}
```

Before approval, retain:

1. authoritative snapshots of both contacts;
2. the reviewed source-to-destination direction;
3. collision decisions for names and customer identifiers;
4. material conversations, notes, labels, inboxes, and company relationships;
5. the exact canonical provider body;
6. the intended post-merge verification fields.

The connector publishes no merge preview, rollback, or per-related-object transaction model. Do not describe the operation as reversible or atomic across all related data.

After settlement, read the destination contact and material relationships. Determine source behavior from provider state rather than assuming deletion, redirection, or archival.

Understand mutation transport

For catalogue mutations, the connector:

- requires a non-blank idempotency key of at most 512 bytes;
- sends `X-AIP-Action-ID` with the Action ID;
- sends `Idempotency-Key` with the supplied key;
- injects the configured account into the route;
- sends either a JSON `body` or `multipart`, never both;
- returns the provider HTTP status, parsed body, and request ID when exposed.

All relationship and merge mutations require approval, advertise unsafe retry, contain restricted PII, and declare rollback unsupported. Forwarding a key does not prove that a particular Chatwoot route deduplicates repeated requests.

Verify the intended state

Use new read Action IDs after a mutation. Verify only authoritative fields needed by the task:

Mutation	Required final read
Attach	List company contacts and identify the intended contact once
Detach	List company contacts and confirm the intended relationship is absent
Merge	Read the destination and each material relationship selected before approval

Retain the mutation Action ID, idempotency key hash, approval record, exact input, configured account and route, provider status, provider request ID when present, final read Action IDs, and checked fields.

Do not retain credentials or unrelated customer fields merely to enlarge the evidence set.

Recover an uncertain mutation

Symptom	Read first	Decision
Attach response lost	Company contacts	Accept the existing relationship or escalate; do not attach blindly
Detach response lost	Company contacts and audit evidence	Accept confirmed absence or escalate; do not repeat a high-risk delete blindly
Merge response lost	Destination, source when addressable, and material relationships	Accept reconciled state or escalate; do not submit a second merge
Provider rejected body	Current objects and pinned provider schema	Correct the intent before creating another Action
Authentication failure	Connector route, account binding, and credential revision	Repair configuration; never move the token into Action input
Oversized or non-JSON result	Provider state and retained request identity	Verify through bounded reads; do not infer failure from response shape

Transport, provider-temporary, cancellation, and oversized-response failures can leave mutation outcome uncertain. Every connector failure reports `retryable: false` even though read contracts separately advertise safe retry.

A corrective relationship or record change is a new governed intent against current state. It is not rollback of the earlier Action.

Completion criteria

The workflow is complete only when:

- route, account, actor, and capability match the intended tenant;

- search ambiguity was resolved through authoritative reads;
- each mutation used one exact input, Action ID, key, and approval;
- attach or detach state was verified through company contacts;
- merge direction and material relationships were reviewed and verified;
- restricted evidence follows tenant redaction and retention policy;
- no uncertain mutation was replaced with an uncorrelated retry.

These criteria validate application handling. They do not establish provider payload compatibility, customer identity correctness, or live qualification.

Related documentation

- Contacts and companies
- Labels, attributes, and filters
- Authentication and account isolation
- Approvals and policy
- Error reference