Chatwoot contacts and companies

Use these 31 capabilities to find and maintain contacts, notes, companies, and their relationships. Treat every result as restricted customer data, including reads that need no human approval.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/chatwoot/capabilities/contacts-and-companies.md

The catalogue freezes capability IDs, provider methods, paths, and AIP policy contracts. It does not freeze provider-specific contact or company body fields.

Family at a glance

Group	Operations	Reads	Mutations	High risk
Contact records and relationships	16	7	9	3
Contact notes	5	2	3	1
Companies and relationships	10	5	5	2
Total	**31**	**14**	**17**	**6**

All 31 capabilities are Tools. The method split is 14 `GET`, ten `POST`, three `PATCH`, and four `DELETE` operations.

The 14 reads are low risk. Eleven mutations are medium risk. The `contact.import` and `contact.export` POST operations plus the four deletes are high risk.

Contact record and relationship operations

Capability	Provider request	Purpose	Risk	Approval
`cap:chatwoot:contact.list`	`GET /api/v1/accounts/{account_id}/contacts`	List contacts	Low	No
`cap:chatwoot:contact.create`	`POST /api/v1/accounts/{account_id}/contacts`	Create a contact	Medium	Required
`cap:chatwoot:contact.get`	`GET /api/v1/accounts/{account_id}/contacts/{contact_id}`	Read one contact	Low	No
`cap:chatwoot:contact.update`	`PATCH /api/v1/accounts/{account_id}/contacts/{contact_id}`	Update one contact	Medium	Required
`cap:chatwoot:contact.delete`	`DELETE /api/v1/accounts/{account_id}/contacts/{contact_id}`	Delete one contact	High	Required
`cap:chatwoot:contact.active`	`GET /api/v1/accounts/{account_id}/contacts/active`	List active contacts	Low	No
`cap:chatwoot:contact.search`	`GET /api/v1/accounts/{account_id}/contacts/search`	Search contacts	Low	No
`cap:chatwoot:contact.filter`	`POST /api/v1/accounts/{account_id}/contacts/filter`	Filter contacts	Medium	Required
`cap:chatwoot:contact.import`	`POST /api/v1/accounts/{account_id}/contacts/import`	Import contacts	High	Required
`cap:chatwoot:contact.export`	`POST /api/v1/accounts/{account_id}/contacts/export`	Export contacts	High	Required
`cap:chatwoot:contact.conversation.list`	`GET /api/v1/accounts/{account_id}/contacts/{contact_id}/conversations`	List contact conversations	Low	No
`cap:chatwoot:contact.inbox.create`	`POST /api/v1/accounts/{account_id}/contacts/{contact_id}/contact_inboxes`	Associate a contact with an inbox	Medium	Required
`cap:chatwoot:contact.label.list`	`GET /api/v1/accounts/{account_id}/contacts/{contact_id}/labels`	List contact labels	Low	No
`cap:chatwoot:contact.label.update`	`POST /api/v1/accounts/{account_id}/contacts/{contact_id}/labels`	Replace contact labels	Medium	Required
`cap:chatwoot:contact.merge`	`POST /api/v1/accounts/{account_id}/actions/contact_merge`	Merge one contact into another	Medium	Required
`cap:chatwoot:contact.contactable_inbox.list`	`GET /api/v1/accounts/{account_id}/contacts/{contact_id}/contactable_inboxes`	List attachable inboxes	Low	No

`contact.filter` is governed as a mutation because its provider method is `POST`. Use `contact.list`, `contact.active`, or `contact.search` when one of those read contracts satisfies the task.

`contact.merge` has no contact ID in its route. The provider body must identify the source and destination contacts. The AIP contract declares no rollback or compensating capability.

Contact-note operations

Capability	Provider request	Purpose	Risk	Approval
`cap:chatwoot:contact.note.list`	`GET /api/v1/accounts/{account_id}/contacts/{contact_id}/notes`	List contact notes	Low	No
`cap:chatwoot:contact.note.create`	`POST /api/v1/accounts/{account_id}/contacts/{contact_id}/notes`	Create a note	Medium	Required
`cap:chatwoot:contact.note.get`	`GET /api/v1/accounts/{account_id}/contacts/{contact_id}/notes/{note_id}`	Read one note	Low	No
`cap:chatwoot:contact.note.update`	`PATCH /api/v1/accounts/{account_id}/contacts/{contact_id}/notes/{note_id}`	Update one note	Medium	Required
`cap:chatwoot:contact.note.delete`	`DELETE /api/v1/accounts/{account_id}/contacts/{contact_id}/notes/{note_id}`	Delete one note	High	Required

Notes can contain free-form customer or operational detail. The restricted, PII-present, redaction-required contract applies even when the provider does not label a field as sensitive.

Company and relationship operations

Capability	Provider request	Purpose	Risk	Approval
`cap:chatwoot:company.list`	`GET /api/v1/accounts/{account_id}/companies`	List companies	Low	No
`cap:chatwoot:company.create`	`POST /api/v1/accounts/{account_id}/companies`	Create a company	Medium	Required
`cap:chatwoot:company.get`	`GET /api/v1/accounts/{account_id}/companies/{company_id}`	Read one company	Low	No
`cap:chatwoot:company.update`	`PATCH /api/v1/accounts/{account_id}/companies/{company_id}`	Update one company	Medium	Required
`cap:chatwoot:company.delete`	`DELETE /api/v1/accounts/{account_id}/companies/{company_id}`	Delete one company	High	Required
`cap:chatwoot:company.search`	`GET /api/v1/accounts/{account_id}/companies/search`	Search companies	Low	No
`cap:chatwoot:company.contact.list`	`GET /api/v1/accounts/{account_id}/companies/{company_id}/contacts`	List linked contacts	Low	No
`cap:chatwoot:company.contact.create`	`POST /api/v1/accounts/{account_id}/companies/{company_id}/contacts`	Attach a contact	Medium	Required
`cap:chatwoot:company.contact.delete`	`DELETE /api/v1/accounts/{account_id}/companies/{company_id}/contacts/{contact_id}`	Detach a contact	High	Required
`cap:chatwoot:company.conversation.list`	`GET /api/v1/accounts/{account_id}/companies/{company_id}/conversations`	List company conversations	Low	No

`company.contact.delete` removes a relationship through the frozen route. Its catalogue description does not claim that the contact record itself is deleted.

Account and path inputs

The connector inserts its configured `account_id`. Do not send an account ID in Action input.

Input	Used by
`contact_id`	Contact reads, updates, deletes, conversations, inbox association, labels, notes, contactable inboxes, and company-contact detach
`note_id`	Contact-note get, update, and delete
`company_id`	Company get, update, delete, contact, and conversation routes

Every path value must be a non-blank string of at most 512 bytes without control characters. Each value is encoded as one URL segment.

`contact.merge`, `contact.import`, `contact.export`, and `contact.filter` use account-level routes without a contact path ID. Their provider bodies own the selection or identity fields.

Generic provider input

Every operation accepts the common catalogue envelope:

Property	Connector boundary
`query`	Optional object; at most 128 keys; scalar values or bounded scalar arrays
`body`	Optional object; at most 512 properties and 16 MiB encoded
`multipart`	Optional bounded fields-and-files object instead of `body`

The connector rejects `body` and `multipart` together. It does not validate contact names, identifiers, email addresses, phone numbers, company attributes, merge participants, import columns, or export filters.

Use the payload schema for the exact pinned provider route. Preserve one canonical body with the approval record; do not add or normalize customer data after approval.

Read one contact

Use an ID returned by the configured account route:

```sh
SH
aipctl action call \
  "$AIP_URL" \
  cap:chatwoot:contact.get \
  --action-id act_chatwoot_contact_get_001 \
  --input '{"contact_id":"205"}'
```

The connector calls:

```
TEXT
GET /api/v1/accounts/{configured_account_id}/contacts/205
```

This low-risk read advertises safe retry and needs no capability-level human approval. It still returns restricted PII. Limit the response to the actor and purpose authorized by tenant policy.

Govern consequential customer-data changes

Operation class	Primary risk	Evidence before dispatch	Verification after dispatch
Create or update	Wrong or duplicate customer identity	Canonical provider body and intended account	Read the exact record
Contact or company delete	Loss of customer data or relationships	Current object, dependencies, and deletion intent	Compare provider state and audit evidence
Company-contact attach or detach	Wrong relationship	Both authoritative IDs and intended direction	List company contacts
Contact-inbox association	Wrong channel relationship	Contact, inbox, and account identity	List contactable or actual provider state
Label replacement	Unintended removal of existing labels	Complete intended replacement set	Read labels
Import	Broad customer-data change	Source artifact identity, scope, mapping, and expected count	Reconcile provider result and sampled records
Export	Disclosure of a broad PII set	Purpose, scope, destination, retention, and access	Verify artifact custody and provider result
Merge	Loss of distinct identity and relationship history	Source, destination, collision review, and authoritative snapshots	Read destination and related objects

The Rust catalogue does not publish a per-item transaction or partial-success model for import, export, or merge. A successful AIP response contains the provider result but does not add an atomic rollback guarantee.

Do not include a copy-ready import, export, or merge body from an unpinned API version. The generic Rust schema is not enough to establish those fields.

Shared mutation contract

All 17 mutations declare:

Contract field	Value
Approval	Required; tenant-policy selector
Approval validity	900,000 ms
Approval evidence	Reason, input snapshot, policy decision
Idempotency	Required; external-account scope; 86,400,000 ms published TTL
Connector retry	Not supported; unsafe
Side effects	`write`, `external_network`
Data	Restricted; contains PII; redaction required
Execution	Synchronous; no async, streaming, or cancellation support
Compensation	Rollback not supported

An idempotency key binds one canonical intent. It does not prove provider deduplication for imports, merges, creates, or relationship changes.

Result contract

Every successful catalogue call returns:

```JSON
{
  "http_status": 200,
  "body": {},
  "provider_request_id": null
}
```

The connector parses JSON into `body`, returns `null` for an empty body, and wraps non-JSON bytes in a `text` object. It reads `provider_request_id` from `x-request-id` and then `x-runtime`.

The output schema does not freeze a contact, note, company, import, export, or merge result shape. Validate the fields the application requires and tolerate additional provider fields.

Failures and recovery

Failure code	Typical cause	Safe response
`connector.chatwoot.invalid_action`	Unknown or unadmitted operation, missing path ID, or malformed generic input	Refresh discovery or correct input before dispatch
`connector.chatwoot.missing_idempotency_key`	Mutation lacks a valid key	Restore the key bound to the original body and intent
`connector.chatwoot.authentication`	Provider returned `401` or `403`	Verify account route and credential revision
`connector.chatwoot.remote_rejected`	Provider rejected fields or current state	Correct authoritative input before another Action
`connector.chatwoot.remote_temporary`	Provider returned `429` or a server error	Retry a read within policy; reconcile a mutation
`connector.chatwoot.transport`	Provider exchange failed	Retry a read within policy; treat mutation outcome as uncertain
`connector.chatwoot.response_too_large`	Response crossed the active bound	Preserve evidence and reconcile a mutation

All connector failures set `retryable: false`. The 14 read contracts separately advertise safe retry.

For an uncertain create, import, merge, relationship change, or note write, inspect authoritative provider state before any new Action. For an uncertain export, determine whether provider-side work or output already exists before initiating another disclosure.

For an uncertain delete, absence alone may not explain when or why the object disappeared. Correlate the original Action, provider request identity, audit evidence, and dependent objects.

Verify an integration

Require all of the following:

- discovery admits the exact capability for the configured account;
- path IDs and provider-body identities come from authoritative state;
- contact merge binds distinct reviewed source and destination identities;
- import and export approval records include bounded scope and artifact

handling;

- every mutation has one retained idempotency key and exact input snapshot;
- the completed result belongs to the submitted Action;
- consequential changes are read back through the configured account;
- customer fields, notes, relationships, exports, and provider errors follow

restricted-data redaction and retention policy;

- uncertain mutations are reconciled instead of repeated.

These checks validate application handling. They do not establish provider payload compatibility, live customer-data correctness, or qualification.

Related documentation

- Chatwoot capability index
- Conversations and messages
- Authentication and account isolation
- Approvals and policy
- Error reference