# Chatwoot labels, attributes, and filters

Use these 15 capabilities to manage account-level labels, custom-attribute definitions, and saved custom filters. Choose a definition operation only when the task is to maintain the reusable metadata object itself.

| | |
|---|---|
| SECTION | Connectors |
| TYPE | Connector |
| PROTOCOL | AIP 1.0 |
| SOURCE | docs/connectors/chatwoot/capabilities/labels-attributes-and-filters.md |

Applying a label, writing an attribute value, and filtering records use other capabilities. Similar names do not make their approvals or inputs interchangeable.

## Family at a glance

| Group | Operations | Reads | Mutations | High-risk deletes |
|---|---|---|---|---|
| Labels | 5 | 2 | 3 | 1 |
| Custom-attribute definitions | 5 | 2 | 3 | 1 |
| Saved custom filters | 5 | 2 | 3 | 1 |
| **Total** | **15** | **6** | **9** | **3** |

All 15 capabilities are Tools. The method split is six `GET`, three `POST`, three `PATCH`, and three `DELETE` operations.

The six reads are low risk. The six create and update operations are medium risk. The three deletes are high risk.

## Choose definition or use

| Intended task | Correct surface | Not this page |
|---|---|---|
| Create or rename an account label | `label.*` | Applying labels to a contact or conversation |
| Define a custom field | `custom_attribute.*` | Writing a value on a conversation or provider record |
| Save a reusable filter definition | `custom_filter.*` | Running `contact.filter` or `conversation.filter` |
| Read records matching criteria now | Record read or governed filter capability | Saved-filter CRUD |

`contact.filter` and `conversation.filter` use `POST` and are governed mutations in the frozen catalogue. `custom_filter.get` is a low-risk read of a saved definition; it does not execute that definition.

## Label operations

| Capability | Provider request | Purpose | Risk | Approval |
|---|---|---|---|---|
| `cap:chatwoot:label.list` | `GET /api/v1/accounts/{account_id}/labels` | List account labels | Low | No |
| `cap:chatwoot:label.create` | `POST /api/v1/accounts/{account_id}/labels` | Create a label | Medium | Required |
| `cap:chatwoot:label.get` | `GET /api/v1/accounts/{account_id}/labels/{label_id}` | Read one label | Low | No |
| `cap:chatwoot:label.update` | `PATCH /api/v1/accounts/{account_id}/labels/{label_id}` | Update one label | Medium | Required |
| `cap:chatwoot:label.delete` | `DELETE /api/v1/accounts/{account_id}/labels/{label_id}` | Delete one label | High | Required |

The source does not define what deleting a label does to existing assignments, filters, automation, or reports. Inspect provider dependencies before approval and verify the resulting account state.

## Custom-attribute-definition operations

| Capability | Provider request | Purpose | Risk | Approval |
|---|---|---|---|---|
| `cap:chatwoot:custom_attribute.list` | `GET /api/v1/accounts/{account_id}/custom_attribute_definitions` | List definitions | Low | No |
| `cap:chatwoot:custom_attribute.create` | `POST /api/v1/accounts/{account_id}/custom_attribute_definitions` | Create a definition | Medium | Required |
| `cap:chatwoot:custom_attribute.get` | `GET /api/v1/accounts/{account_id}/custom_attribute_definitions/{custom_attribute_id}` | Read one definition | Low | No |
| `cap:chatwoot:custom_attribute.update` | `PATCH /api/v1/accounts/{account_id}/custom_attribute_definitions/{custom_attribute_id}` | Update one definition | Medium | Required |
| `cap:chatwoot:custom_attribute.delete` | `DELETE /api/v1/accounts/{account_id}/custom_attribute_definitions/{custom_attribute_id}` | Delete one definition | High | Required |

The generic input schema does not validate provider field type, model scope, display rules, permitted values, or migration behavior. Those fields remain owned by the pinned Chatwoot route.

Deleting or changing a definition has no AIP compensating operation. Review current values and dependent behavior before the mutation.

## Saved-custom-filter operations

| Capability | Provider request | Purpose | Risk | Approval |
|---|---|---|---|---|
| `cap:chatwoot:custom_filter.list` | `GET /api/v1/accounts/{account_id}/custom_filters` | List saved filters | Low | No |
| `cap:chatwoot:custom_filter.create` | `POST /api/v1/accounts/{account_id}/custom_filters` | Create a saved filter | Medium | Required |
| `cap:chatwoot:custom_filter.get` | `GET /api/v1/accounts/{account_id}/custom_filters/{custom_filter_id}` | Read one saved filter | Low | No |
| `cap:chatwoot:custom_filter.update` | `PATCH /api/v1/accounts/{account_id}/custom_filters/{custom_filter_id}` | Update one saved filter | Medium | Required |
| `cap:chatwoot:custom_filter.delete` | `DELETE /api/v1/accounts/{account_id}/custom_filters/{custom_filter_id}` | Delete one saved filter | High | Required |

The catalogue does not say that a saved filter is private, shared, executable, or bound to a particular record type. Interpret those provider fields from the pinned API response rather than the capability name.

## Account and path inputs

The host inserts the configured `account_id`. It is not an Action input.

| Input | Required by |
|---|---|
| `label_id` | Label get, update, and delete |
| `custom_attribute_id` | Custom-attribute get, update, and delete |
| `custom_filter_id` | Saved-filter get, update, and delete |

Each ID is a non-blank string of at most 512 bytes without control characters. The connector inserts it as one encoded URL segment.

List and create operations use account-level collection routes. Create fields belong in the provider body, not in a caller-constructed path.

## 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 the provider fields of any label, attribute definition, or saved filter.

Use the exact pinned provider schema and preserve one canonical body with approval evidence. A generic schema-valid empty body is not proof of a valid create or update.

## Read one custom-attribute definition

Use an ID returned by the configured account:

```
SH
aipctl action call \
  "$AIP_URL" \
  cap:chatwoot:custom_attribute.get \
  --action-id act_chatwoot_custom_attribute_get_001 \
  --input '{"custom_attribute_id":"12"}'
```

The connector sends:

```
TEXT
GET /api/v1/accounts/{configured_account_id}/custom_attribute_definitions/12
```

The read needs no capability-level approval and advertises safe retry. The result is still classified as restricted and PII-present by the capability contract.

## Review a metadata mutation

Before create or update:

1. Read the current definition when it exists.
2. Confirm the exact pinned provider fields and intended account.
3. Identify records, workflows, reports, or users that can depend on it.
4. Retain one canonical body with the approval record.
5. Assign one external-account-scoped idempotency key.
6. Dispatch once and read the resulting definition.

Before delete, additionally preserve the current object and review provider dependencies. The AIP contract publishes no cascade semantics, rollback, or compensating capability.

## Shared mutation contract

All nine 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 |

The connector forwards the idempotency key to Chatwoot. The contract does not prove provider deduplication or prevent dependency changes after approval.

## Result contract

Every successful operation returns:

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

`body` contains parsed JSON, `null` for an empty response, or a `text` wrapper for non-JSON bytes. `provider_request_id` uses `x-request-id` first and `x-runtime` second.

The output schema does not freeze provider definition fields. Validate the shape the application needs and preserve unknown fields.

## Failures and recovery

| Failure code | Typical cause | Safe response |
|---|---|---|
| `connector.chatwoot.invalid_action` | Unknown or unadmitted ID, missing path input, or malformed generic envelope | Refresh discovery or correct input before dispatch |
| `connector.chatwoot.missing_idempotency_key` | Mutation lacks a valid key | Restore the key for the original definition intent |
| `connector.chatwoot.authentication` | Provider returned `401` or `403` | Verify account binding and credential revision |
| `connector.chatwoot.remote_rejected` | Provider rejected fields or current state | Correct authoritative input before a new 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 six read contracts separately advertise safe retry.

For an uncertain create, update, or delete, list or read definitions before another Action. Do not recreate a definition with a new ID merely because the first response was lost.

## Verify an integration

Require all of the following:

- discovery admits the exact definition capability;
- the chosen surface matches definition management rather than record use;
- each path ID belongs to the configured account;
- provider body fields match the pinned route;
- approval includes dependency and impact review;
- each mutation retains one Action, canonical input, and idempotency key;
- a read confirms the intended definition after completion;
- restricted metadata follows redaction and retention policy;
- uncertain mutations are reconciled instead of repeated.

These checks validate application handling. They do not establish cascade behavior, provider payload compatibility, or live qualification.

## Related documentation

- Chatwoot capability index

- Conversations and messages
- Contacts and companies
- Approvals and policy
- Error reference