Chatwoot automation, macros, and canned responses

Use these 16 capabilities to manage reusable text and behavior in the configured Chatwoot account. Review both the immediate provider request and the later effects of the object being created, changed, cloned, or executed.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/chatwoot/capabilities/automation-macros-and-canned-responses.md

The generic catalogue marks mutations as `write` plus `external_network`. Those markers do not describe every future message, assignment, label, or workflow effect that provider configuration may cause.

Family at a glance

Group	Operations	Reads	Mutations	High risk
Canned responses	4	1	3	1
Automation rules	6	2	4	1
Macros	6	2	4	2
Total	**16**	**5**	**11**	**4**

The method split is five `GET`, five `POST`, three `PATCH`, and three `DELETE` operations. Five reads are low risk, seven mutations are medium risk, and four mutations are high risk.

Fourteen capabilities are Tools. `automation_rule.clone` and `macro.execute` are Workflows. Both still use synchronous execution without AIP async, streaming, or cancellation support.

Canned-response operations

Capability	Provider request	Purpose	Risk	Approval
`cap:chatwoot:canned_response.list`	`GET /api/v1/accounts/{account_id}/canned_responses`	List canned responses	Low	No
`cap:chatwoot:canned_response.create`	`POST /api/v1/accounts/{account_id}/canned_responses`	Create a canned response	Medium	Required
`cap:chatwoot:canned_response.update`	`PATCH /api/v1/accounts/{account_id}/canned_responses/{canned_response_id}`	Update one canned response	Medium	Required
`cap:chatwoot:canned_response.delete`	`DELETE /api/v1/accounts/{account_id}/canned_responses/{canned_response_id}`	Delete one canned response	High	Required

The reviewed catalogue has no `canned_response.get` capability. Use the list result only when its provider shape contains enough information for the task. Do not synthesize a get ID or call an unadmitted provider path.

Canned text can later be sent to customers. Review restricted data, placeholders, links, and tenant policy before creating or updating it.

Automation-rule operations

Capability	Provider request	Purpose	Risk	Approval
`cap:chatwoot:automation_rule.list`	`GET /api/v1/accounts/{account_id}/automation_rules`	List automation rules	Low	No
`cap:chatwoot:automation_rule.get`	`GET /api/v1/accounts/{account_id}/automation_rules/{automation_rule_id}`	Read one rule	Low	No
`cap:chatwoot:automation_rule.create`	`POST /api/v1/accounts/{account_id}/automation_rules`	Create a rule	Medium	Required
`cap:chatwoot:automation_rule.update`	`PATCH /api/v1/accounts/{account_id}/automation_rules/{automation_rule_id}`	Update one rule	Medium	Required
`cap:chatwoot:automation_rule.delete`	`DELETE /api/v1/accounts/{account_id}/automation_rules/{automation_rule_id}`	Delete one rule	High	Required
`cap:chatwoot:automation_rule.clone`	`POST /api/v1/accounts/{account_id}/automation_rules/{automation_rule_id}/clone`	Clone one rule	Medium	Required

The source classifies clone as a Workflow but does not publish a transaction, background-run handle, or rollback operation. Treat the returned provider body as the only immediate result, then inspect the cloned rule.

The generic body schema does not validate conditions, actions, ordering, activation, or other provider rule fields. Approval must cover the complete provider body and the behavior it can trigger.

Macro operations

Capability	Provider request	Purpose	Risk	Approval
`cap:chatwoot:macro.list`	`GET /api/v1/accounts/{account_id}/macros`	List macros	Low	No
`cap:chatwoot:macro.get`	`GET /api/v1/accounts/{account_id}/macros/{macro_id}`	Read one macro	Low	No
`cap:chatwoot:macro.create`	`POST /api/v1/accounts/{account_id}/macros`	Create a macro	Medium	Required
`cap:chatwoot:macro.update`	`PATCH /api/v1/accounts/{account_id}/macros/{macro_id}`	Update one macro	Medium	Required
`cap:chatwoot:macro.delete`	`DELETE /api/v1/accounts/{account_id}/macros/{macro_id}`	Delete one macro	High	Required
`cap:chatwoot:macro.execute`	`POST /api/v1/accounts/{account_id}/macros/{macro_id}/execute`	Execute a macro against its declared target	High	Required

`macro.execute` is the only non-delete high-risk operation in this family. It is a synchronous Workflow and has no AIP rollback. The body that identifies the target and execution context remains provider-owned.

Do not infer safe repetition from the verb `execute` or from the idempotency key. A lost response can hide provider-side effects on the declared target.

Account and path inputs

The connector injects its configured `account_id`. Callers cannot select another account through Action input.

Input	Required by
`canned_response_id`	Canned-response update and delete
`automation_rule_id`	Rule get, update, delete, and clone
`macro_id`	Macro get, update, delete, and execute

Each path value is a non-blank string of at most 512 bytes without control characters. The URL builder encodes it as one path segment.

Create and list operations use account-level collection routes. Their provider identifiers or fields belong in `query` or `body`, not in a guessed 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 a canned-response payload, rule definition, macro definition, clone option, or execution target.

Use the exact pinned provider schema. Retain the canonical body with the approval evidence and idempotency record.

Read one macro

Use a macro ID returned by the same configured account:

```
SH
aipctl action call \
  "$AIP_URL" \
  cap:chatwoot:macro.get \
  --action-id act_chatwoot_macro_get_001 \
  --input '{"macro_id":"41"}'
```

The connector sends:

```
TEXT
GET /api/v1/accounts/{configured_account_id}/macros/41
```

The read needs no capability-level approval and advertises safe retry. Treat the provider body as restricted data because it can describe customer-visible or operational behavior.

Review a mutation by intent

Intent	Review before approval	Verify after completion
Create reusable text	Content, placeholders, links, audience, and owner	List canned responses
Update reusable text	Current object, exact changed fields, and downstream use	List canned responses and compare
Create or update a rule	Complete rule definition from the pinned provider schema	Read the rule
Clone a rule	Source identity and provider-owned clone options	Read the returned or discovered clone
Create or update a macro	Complete provider definition and intended use	Read the macro
Execute a macro	Macro revision, exact target, execution body, and expected effects	Read the target and related conversation state
Delete any reusable object	Current object, references, owner, and removal intent	Confirm provider state and audit evidence

The source does not prove that list results expose every field required for review. Use `automation_rule.get` or `macro.get` when exact reads exist. For a canned response, retain authoritative provider evidence available through the admitted list surface.

Shared mutation contract

All 11 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 idempotency key identifies one governed intent. The connector forwards it to Chatwoot but does not establish provider deduplication.

Result contract

Every successful catalogue call returns:

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

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

The result does not include an AIP Workflow run ID. A Workflow kind here describes capability semantics, not asynchronous execution.

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 governed input
`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 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`. Only the five read contracts advertise safe retry.

For an uncertain create, update, clone, or delete, inspect the provider object before another Action. For uncertain `macro.execute`, inspect the exact target and resulting conversation state before risking repeated effects.

No mutation has an AIP compensating capability. A corrective change requires new approval against current provider state.

Verify an integration

Require all of the following:

- discovery admits the exact reusable-behavior capability;
- every path ID belongs to the configured account;
- provider body fields match the pinned route;
- approval covers both immediate changes and expected future behavior;
- each mutation retains one Action ID, one canonical input, and one

idempotency key;

- clone and execute are treated as synchronous, unsafe-to-retry Workflows;
- the completed result belongs to the submitted Action;
- reads confirm the created, changed, cloned, or deleted object when possible;
- restricted reusable content and rule detail follows redaction and retention

policy;

- uncertain mutations are reconciled instead of repeated.

These checks validate application behavior. They do not qualify a provider rule engine, macro effect, or customer-visible result.

Related documentation

- Chatwoot capability index
- Conversations and messages
- Chatwoot connector configuration
- Approvals and policy

- Error reference