Hermes scheduled job capabilities

Use these eight endpoint-qualified capabilities to manage jobs in one Hermes profile's persistent scheduler.

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/hermes-agent/capabilities/scheduled-jobs.md

An accepted AIP action confirms that Hermes stored or changed scheduler state. It does not confirm that a job ran, produced output, or delivered a message. In particular, `job_run` only makes a job due for later scheduler execution.

A scheduled job can invoke an agent, tools, scripts, providers, and delivery adapters configured outside this API. Inspect a job before approving any operation that creates, changes, resumes, runs, or deletes it.

Operations

Replace `` with the normalized endpoint ID returned by AIP discovery.

Operation	Provider request	Kind	Risk	Approval	Key	Retry
`jobs_list`	`GET /api/jobs`	Tool	Low	No	Optional	Safe
`job_create`	`POST /api/jobs`	Tool	Medium	Yes	Required	Unsafe
`job_get`	`GET /api/jobs/{job_id}`	Tool	Low	No	Optional	Safe
`job_update`	`PATCH /api/jobs/{job_id}`	Tool	Medium	Yes	Required	Unsafe
`job_delete`	`DELETE /api/jobs/{job_id}`	Tool	High	Yes	Required	Unsafe
`job_pause`	`POST /api/jobs/{job_id}/pause`	Tool	Medium	Yes	Required	Unsafe
`job_resume`	`POST /api/jobs/{job_id}/resume`	Tool	Medium	Yes	Required	Unsafe
`job_run`	`POST /api/jobs/{job_id}/run`	Agent	High	Yes	Required	Unsafe

All eight capabilities are stable. They require the selected endpoint's Bearer credential. Input cannot select a different endpoint or provider URL.

Every path operation requires a 12-character lowercase hexadecimal `job_id`, for example `a1b2c3d4e5f6`. A name, uppercase ID, shortened ID, or arbitrary UUID is rejected by the connector before the provider call.

Choose a schedule

Hermes parses four schedule forms.

Intent	Example	Stored kind	First eligible time
Run once after a delay	`30m`, `2h`, `1d`	`once`	Creation time plus the delay

Intent	Example	Stored kind	First eligible time
Run at an interval	`every 30m`, `every 2h`	`interval`	Creation time plus the interval
Follow a cron expression	`0 9 * * *`	`cron`	Next matching occurrence
Run once at a timestamp	`2030-01-15T09:00:00+04:00`	`once`	The supplied instant

A timestamp without an offset is interpreted in the configured Hermes timezone. Supply an explicit offset when an application must preserve one unambiguous instant across deployments.

A one-shot more than 120 seconds in the past is rejected. A one-shot created without `repeat` receives a run limit of one. Use recurring interval or cron schedules for repeated work.

Cron expressions depend on the provider's `croniter` installation. The pinned Hermes source treats it as a runtime dependency, but an unavailable dependency still makes expression parsing or next-run computation fail.

Shared AIP contract

Contract field	List and get	Create and update	Pause and resume	Delete	Run
Side effects	`read`, `external_network`	`read`, `write`, `external_network`	`read`, `write`, `external_network`	`read`, `delete`, `external_network`	`read`, `write`, `external_network`
Published data class	Internal; no PII	Confidential; contains PII	Internal; contains PII	Restricted; contains PII	Confidential; contains PII
Redaction	Required	Required	Required	Required	Required
Idempotency	Optional	Required	Required	Required	Required
Retry support	Yes	No	No	No	No
Retry safety	Safe	Unsafe	Unsafe	Unsafe	Unsafe
Completion	Sync	Sync	Sync	Sync	Sync request; delayed job
Compensation	Rollback not supported	Rollback not supported	Rollback not supported	Rollback not supported	Rollback not supported

All eight contracts declare a 60-second timeout, 5-second expected latency, 10-second maximum queue-delay hint, and `99.9%` availability target. The connector applies a 30-second HTTP timeout to each JSON request. Trusted execution is also bounded by the AIP deadline and cancellation token.

These values describe the control request only. They do not bound the later job execution. The availability value is a declaration, not observed service or qualification evidence.

The six mutations require tenant-policy approval. Approval lasts 15 minutes and records a reason, input snapshot, and policy decision. No operation publishes transaction support, and no compensation capability is available.

The `job_run` contract understates possible downstream effects. A stored job can later send a message or execute code even though the capability declares only read, write, and external-network effects. Base policy on the retrieved job and the endpoint's Hermes configuration, not that generic list alone.

List jobs

`jobs_list` returns enabled jobs by default:

```
JSON
{}
```

Include paused and otherwise disabled records when reconciling scheduler state:

```
JSON
{
  "include_disabled": true
}
```

The provider returns one object containing a `jobs` array. There is no limit, cursor, offset, total count, or documented sort. Results retain storage order.

The connector also accepts a `query` object and forwards scalar values. Prefer the typed top-level `include_disabled` field. Supplying the same key in both places creates duplicate query parameters with provider-dependent selection.

The response contains full normalized job records rather than a safe summary. A record can include its prompt, skills, model and provider routing, delivery target, origin network metadata, schedule, timestamps, errors, and execution state. Jobs created by other Hermes surfaces can expose fields that AIP cannot set.

The list is profile-scoped, not caller-scoped. It covers every job in the selected endpoint's active `HERMES_HOME`, including jobs created outside AIP.

Create a job

Use top-level fields for the frozen AIP schema:

```
JSON
{
  "name": "Reconcile support queue",
  "schedule": "every 30m",
  "prompt": "Summarize unresolved queue items and record the totals.",
  "deliver": "local",
  "skills": ["support-reporting"],
  "repeat": 24
}
```

Field	Requirement	Frozen AIP bound
`name`	Required, non-empty string	200 characters
`schedule`	Required, non-empty string	512 characters
`prompt`	Optional string	5,000 characters
`deliver`	Optional string	256 characters
`skills`	Optional string array	256 items, 256 characters each
`repeat`	Optional positive integer	At least one

The schema rejects other top-level fields. Although a `body` property exists for transport compatibility, mixing it with typed fields can cause the wrapper to replace those fields before the provider call. Use the typed form above.

The provider trims `name` and `schedule`, parses the schedule, and generates a 12-hex job ID. When the optional prompt scanner imports successfully, it also rejects blocked patterns in a non-empty prompt. An API-created job records `api_server` origin metadata and defaults delivery to `local`.

Hermes stores the initial job as enabled with state `scheduled`. It calculates `next_run_at`, initializes run counts and status, and returns:

```
JSON
{
  "job": {
    "id": "a1b2c3d4e5f6",
    "name": "Reconcile support queue",
    "schedule": {
      "kind": "interval",
      "minutes": 30,
      "display": "every 30m"
    },
    "schedule_display": "every 30m",
    "repeat": {
      "times": 24,
      "completed": 0
    },
    "enabled": true,
    "state": "scheduled",
```

```
    "last_run_at": null,
    "last_status": null,
    "last_error": null,
    "last_delivery_error": null,
    "deliver": "local"
  }
}
```

The actual response contains more fields. Treat the returned ID as the only valid identifier for later AIP operations.

Put the required idempotency key on the AIP Action, not in this JSON. The connector forwards it as `Idempotency-Key`; the pinned provider handler does not interpret that header.

Inspect one job

Use `job_get` before every governed mutation:

```
JSON
{
  "job_id": "a1b2c3d4e5f6"
}
```

The provider returns `{"job": }` or `404` when the ID is absent. Inspect at least these fields:

- `enabled`, `state`, `next_run_at`, and `schedule` for eligibility;
- `prompt`, `skills`, and any script or toolset fields for execution scope;
- `provider`, `model`, and delivery fields for external destinations;
- `repeat`, `last_run_at`, `last_status`, and errors for prior outcomes.

An origin record is audit context. It is not proof that the current caller created or owns the job.

Update safe fields

`job_update` accepts a path ID and a partial body:

```
JSON
{
  "job_id": "a1b2c3d4e5f6",
  "name": "Reconcile priority support queue",
  "schedule": "every 1h",
  "prompt": "Summarize unresolved priority items and record the totals."
}
```

The provider whitelists `name`, `schedule`, `prompt`, `deliver`, `skills`, legacy `skill`, `repeat`, and `enabled`. Other provider fields are discarded. An effective patch with no whitelisted field returns `400`.

A changed schedule is reparsed. If the job is not paused, Hermes recalculates its next eligible time. A non-empty changed prompt is scanned again.

Do not patch `repeat` in the pinned version. The REST handler forwards an integer, but the scheduler later expects an object with `times` and `completed`. This source-level shape mismatch can leave the job unreadable to execution paths. To change the limit:

1. Pause and inspect the old job.
2. Create a replacement with the intended schedule and `repeat` value.
3. Inspect the replacement and then delete the old job.

Do not use a direct `enabled` patch as a substitute for pause or resume. It does not maintain `state`, `paused_at`, `paused_reason`, and next-run timing as one lifecycle operation.

As with create, use typed top-level fields instead of combining them with a `body` wrapper. The connector removes `job_id` and transport-only fields from the provider body.

Pause and resume

Pause an enabled job with only its ID:

```
JSON
{
  "job_id": "a1b2c3d4e5f6"
}
```

`job_pause` sets `enabled` to false, state to `paused`, records `paused_at`, and returns the updated job. This AIP route has no pause-reason input, so the provider stores no caller-supplied reason.

`job_resume` clears pause metadata, enables the job, and recomputes `next_run_at` from the time of the request. Resuming an active job can shift its next run, so call it only for a record whose state you just verified as paused.

An expired one-shot cannot be resumed. The provider returns an error when its stored time is beyond the one-shot grace window. Create a replacement instead of retrying the same resume request.

Trigger a run

`job_run` uses the same ID-only input. Despite its name, the provider does not execute the job in the HTTP request. It performs this state transition:

```
JSON
{
  "enabled": true,
  "state": "scheduled",
  "paused_at": null,
  "paused_reason": null,
  "next_run_at": "<provider-current-time>"
}
```

It then returns `{"job": }`. The later scheduler owns model or script execution, output storage, delivery, and final status.

The built-in gateway ticker normally checks every 60 seconds. A configured external scheduler can use different timing. Unlike create, update, delete, pause, and resume, the pinned `job_run` handler does not send the best-effort jobs-changed notification to an external scheduler provider. Verify its behavior separately before promising immediate execution on that topology.

Treat a successful `job_run` response as “made due,” not “ran successfully.” For a recurring job, poll `job_get` until `last_run_at` advances, then inspect `last_status`, `last_error`, and `last_delivery_error`.

A finite one-shot can be removed automatically when its run limit is reached, so `404` after dispatch is not proof of either success or failure. Correlate it with retained output, delivery, and endpoint operational evidence. The AIP job family has no endpoint for reading saved run-output files.

Delete a job

`job_delete` removes the selected record and its per-job output directory. A successful provider response is:

```
JSON
{
  "ok": true
}
```

Deletion cannot recall messages already delivered, reverse tool or script effects, remove data written elsewhere, or prove that an in-flight execution was cancelled. Pause first, verify that no run is active through operational signals, and then delete.

The provider resolves and validates the output path before saving the record removal. A legacy unsafe ID therefore fails closed instead of partially deleting scheduler state. AIP only accepts current 12-hex IDs.

Record removal and output cleanup are not one transaction. Hermes saves the record removal before deleting the output directory. If that filesystem step fails, the provider can return `500` after the job is already absent and leave output behind. Re-list before deciding whether any cleanup remains.

Persistence and execution boundary

Jobs are stored under the active profile at `HERMES_HOME/cron/jobs.json`. Writes use a temporary file plus atomic replacement. Directories and files are set to owner-only permissions where the operating system supports them.

Mutations use an in-process lock and a best-effort cross-process advisory lock. If the advisory lock cannot be opened, Hermes logs a warning and continues with in-process protection. This is not a distributed database or a cross-instance consensus boundary.

The built-in scheduler prevents overlapping dispatch of the same job in one runtime and advances recurring schedules before execution. One-shots use durable run claims and repeat accounting. These controls reduce duplicate dispatch; they do not establish exactly-once model, tool, script, or delivery effects across every crash and external system.

Each execution writes a Markdown output file under `HERMES_HOME/cron/output/`. The pinned default keeps the newest 50 files per job. Output is saved before delivery. Execution failure and delivery failure are recorded separately.

Protect job data and authority

Treat every job record as Confidential and potentially containing PII. The published AIP contract labels list/get results Internal with `contains_pii` false, but real records can disclose prompts, network origin, delivery addresses, errors, model routing, skills, and operational timestamps.

Provider handlers authenticate the endpoint Bearer token. They do not perform per-job principal, creator, or tenant ownership checks. An AIP tenant binding can restrict which endpoint a tenant reaches, but it does not add row-level authorization inside the shared profile store.

Use a separate endpoint and `HERMES_HOME` for each isolation domain, or add an authorization layer that filters job IDs and list results. Never expose an unfiltered shared-profile job catalogue to mutually untrusted principals.

Redact prompts, origins, delivery targets, provider details, and errors from logs and support bundles. Store endpoint credentials outside action input.

Idempotency and retry decisions

Required mutation keys are scoped to the full endpoint-qualified capability ID and retained by the AIP runtime for 24 hours. Reusing a key returns the original AIP result. The provider itself does not enforce the forwarded `Idempotency-Key` header.

The replay record does not revalidate an input hash. Reusing the same key with a different job ID or body can return the earlier result without applying the new intent. Use a new key whenever the selected job or mutation changes.

Situation	Decision
List/get failed before a response	Retry within the trusted deadline
Mutation returned a definitive provider response	Do not retry with a new key
Mutation timed out with the original AIP action unresolved	Reconcile with list/get before any new action
`job_run` returned success	Poll status; do not submit another run immediately

Situation	Decision
Resume rejected an expired one-shot	Create a replacement with a new key
Repeat limit must change	Pause, recreate, verify, and delete; do not patch `repeat`

An AIP timeout or cancellation can race a provider mutation that already persisted. Read current state before deciding whether compensation is needed.

Failures

Failure	Meaning	Operator response
Local invalid-input error	Missing or malformed 12-hex ID, query, or body	Correct input; no provider request occurred
`401` or `403`	Endpoint credential absent, invalid, or denied	Repair endpoint authentication
`400`	Invalid ID, field, prompt, repeat, or effective update	Correct the request; do not replay unchanged
`404`	Job does not exist in this profile	Re-list including disabled jobs
`501`	Hermes cron module is unavailable	Repair the endpoint installation
`500`	Parsing, storage, scheduler, or provider-side handler failure	Reconcile state; inspect redacted endpoint logs
AIP deadline or cancellation	Trusted execution stopped waiting	Reconcile before issuing a new mutation

Schedule parsing exceptions can surface as provider `500` in the pinned REST handler rather than a more specific `400`. Read the redacted error text and the submitted schedule before treating it as an infrastructure outage.

Verify an integration

Before enabling production scheduling, verify from source and controlled endpoint evidence that:

1. Discovery exposes exactly eight job capabilities for the intended endpoint.
2. Tenant binding and the endpoint Bearer credential fail closed.
3. IDs are returned by create and never guessed from names.
4. List without `include_disabled` hides a paused job; list with it shows one.
5. Resume recalculates timing and an expired one-shot fails visibly.
6. `job_run` returns before execution and status changes only after a tick.
7. Execution failure and delivery failure remain distinguishable.
8. A repeated AIP mutation key returns the original result.
9. A changed mutation uses a new key and a fresh approval decision.
10. Repeat limits are changed only through replacement in this pinned version.
11. Job records and output files follow the intended profile isolation.
12. Delete removes local record/output but is not treated as effect rollback.

Do not report live qualification from this source-only checklist. Record actual environment, revision, time, and retained artifacts on the qualification page.

Related documentation

- Hermes capability index
- Durable runs and approvals

- Persistent sessions
- Approvals and policy
- Errors and retry decisions
- Authentication and endpoints