Hermes chat and Responses capabilities

Use these six endpoint-qualified capabilities to run a Hermes model turn, stream it, continue a Responses chain, read a stored response, or delete that stored response. They are for applications using a Hermes endpoint already admitted to t

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/hermes-agent/capabilities/chat-and-responses.md

Chat and Responses calls can run tools and create external effects. They are not read-only text generation, and an interrupted call is not safe to replay blindly. This page covers non-streaming request, result, continuity, storage, and retry behavior. The later streaming guide and cursor reference cover individual SSE events, truncation, publication, and cancellation.

Operations

Replace `` with the normalized endpoint ID present in AIP discovery.

Operation	Provider request	Risk	Approval	Key	Retry	Stability
`chat`	`POST /v1/chat/completions`	Medium	No	Optional	Unsafe	Stable
`chat_stream`	`POST /v1/chat/completions`	Medium	No	Optional	Unsafe	Stable
`responses`	`POST /v1/responses`	Medium	No	Optional	Unsafe	Stable
`responses_stream`	`POST /v1/responses`	Medium	No	Optional	Unsafe	Stable
`response_get`	`GET /v1/responses/{response_id}`	Low	No	Optional	Safe	Experimental
`response_delete`	`DELETE /v1/responses/{response_id}`	High	Yes	Required	Unsafe	Experimental

All six are AIP Agent capabilities and require the selected endpoint's Bearer token. The endpoint comes from the capability ID. Input cannot select a different endpoint or provider URL.

Choose chat or Responses

Need	Use	Continuity owner	Result
Submit OpenAI-compatible messages for one turn	`chat` or `chat_stream`	Explicit Hermes `session_id`, optional long-term-memory `session_key`, or supplied message history	Chat completion object or stream aggregate
Continue work by a stored response ID	`responses` or `responses_stream`	`previous_response_id` in the Hermes response store	Responses object or stream aggregate
Continue work by an application name	`responses` or `responses_stream`	`conversation` mapped to the latest stored response	Responses object or stream aggregate
Supply all prior history yourself	`responses` or `responses_stream`	`conversation_history` in the current request	Responses object or stream aggregate
Inspect or remove one stored Responses object	`response_get` or `response_delete`	`response_id` returned by a stored Responses call	Stored public object or deletion receipt

Use the durable-runs capabilities when an application needs a pollable run, structured lifecycle events, or an approval continuation. Use the session capabilities when the application needs an addressable persistent session resource. Neither lifecycle is created merely by calling `chat`.

Shared AIP contract

The four model-executing operations publish these values:

Contract field	Published value
Side effects	`read`, `write`, `send_message`, `external_network`
Data	Confidential; contains PII; redaction required
Transaction	None
Compensation	Rollback not supported
Expected latency	5000 ms (5 s) for non-streaming; 30000 ms (30 s) for streaming
AIP contract timeout	60000 ms (60 s) for non-streaming; 300000 ms (300 s) for streaming
Maximum queue-delay hint	10000 ms (10 s)
Availability target	`99.9%`, declared but not observed by this page

Every operation supports synchronous execution. The two stream operations also advertise async, streaming, and cancellation support, with streaming expected completion. `async_expected` remains false for both. The non-streaming operations do not advertise async, streaming, cancellation, or connector retry.

`response_get` publishes `read` and `external_network` side effects. It is the only retry-supported operation in this family. `response_delete` publishes `read`, `delete`, and `external_network`, requires tenant-policy approval, and has no rollback.

Submit a chat request

The published AIP schema selects exactly one input branch:

- `prompt`: one non-empty user prompt;
- `query`: an alias for `prompt`; or
- `messages`: one non-empty OpenAI-compatible message array.

Use only one branch. The connector's defensive runtime parser gives a non-empty `messages` array precedence, then `prompt`, then `query`, but normal AIP schema validation rejects an input that satisfies more than one `oneOf` branch.

The connector builds a `messages` array from `prompt` or `query`, defaults `model` to `hermes-agent`, and forces `stream` to match the selected capability. It accepts these additional typed fields:

Field	Purpose or constraint
`model`	Hermes model alias; defaults to `hermes-agent`
`session_id`	Hermes transcript continuity; at most 256 characters; sent as `X-Hermes-Session-Id`
`session_key`	Long-term-memory scope; at most 256 characters; sent as `X-Hermes-Session-Key`
`temperature`, `top_p`	Numeric sampling controls; the AIP schema adds no range
`max_tokens`	Non-negative integer in the typed connector request
`stop`	OpenAI-compatible stop value
`tools`, `tool_choice`	OpenAI-compatible tool declarations and selection
`response_format`	OpenAI-compatible response-format value
`user`	Application-supplied user identifier
`metadata`	Application metadata object

Unknown fields are accepted and forwarded for provider compatibility. Do not put credentials or trusted AIP identity claims in those fields. The connector removes `session_id`, `session_key`, and `idempotency_key` from the provider JSON body because they belong to headers. Trusted AIP execution overwrites the header idempotency value with `Action.idempotency_key`.

A bounded non-streaming input can look like this:

```
JSON
{
  "messages": [
    {
      "role": "system",
      "content": "Answer with a short operational summary."
    },
    {
      "role": "user",
      "content": "Summarize the current incident notes."
    }
  ],
  "model": "hermes-agent",
  "temperature": 0.2,
  "session_id": "incident-review-2030-01",
  "session_key": "support:incident-review"
}
```

Hermes concatenates `system` messages into an ephemeral system prompt. The pinned handler takes the final supported `user` or `assistant` message as the current agent input without checking that its role is `user`. End every request with a visible `user` message. Earlier supported messages become history, and other roles do not enter this chat history path.

When `session_id` is present, Hermes attempts to load that transcript's history from its session database instead of using the earlier messages in the request. Supply a stable explicit ID from the first turn when the application owns continuity.

If `session_id` is absent, Hermes derives an ID and returns it only in an HTTP header. The current connector returns the JSON body, not provider response headers.

If the session database is unavailable, the handler can continue with the request history. If history loading raises an error, it logs the failure and continues with empty history. A successful chat result therefore does not by itself prove that an earlier transcript was loaded.

Read a chat result

A successful non-streaming result follows the chat-completion shape:

```
JSON
{
  "id": "chatcmpl-0123456789abcdef0123456789abc",
  "object": "chat.completion",
  "created": 1893456000,
  "model": "hermes-agent",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "The incident is contained and verification is pending."
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 120,
    "completion_tokens": 11,
    "total_tokens": 131
  }
}
```

Read `choices[0].message.content`, but also inspect `finish_reason`. When Hermes has usable partial text, it can return HTTP 200 with `finish_reason` set to `length` or `error` and an additional `hermes` object describing completion, partial, failed, and redacted error state. A failed or partial run with no usable text returns an OpenAI-style error with HTTP 502.

Submit a Responses request

Use a top-level `body` object for the provider request. Keep connector control fields beside it so they become headers rather than provider JSON:

```
JSON
{
  "body": {
    "model": "hermes-agent",
    "input": "Summarize the current incident notes.",
    "instructions": "Return three concise findings.",
    "store": true,
    "truncation": "auto"
  },
  "session_key": "support:incident-review"
}
```

Set `Action.idempotency_key` as a sibling of `Action.input`. Do not create a second business key inside the provider body. During trusted execution, the connector sends the action key as `Idempotency-Key` and forces `stream` to `false` or `true` for the selected capability.

The AIP schema is intentionally open for forward-compatible Responses fields, but the pinned Hermes provider requires `input`. It accepts a string or an array of strings or message objects. These continuity rules then apply:

Field	Behavior
`previous_response_id`	Loads retained history, session ID, and prior instructions when new instructions are absent
`conversation`	Resolves an application name to its latest stored response; a missing name starts a new chain
`conversation_history`	A non-empty array supplies explicit history and takes precedence over `previous_response_id`
`store`	Defaults to `true`; controls response storage and future lookup or chaining
`truncation`	`auto` keeps the newest 100 history messages; omission keeps the full retained history

`conversation` and `previous_response_id` cannot appear together. An explicit `conversation_history` must be an array whose entries contain `role` and `content`. The last normalized item in `input` becomes the current agent input; preceding items join the history. End an input array with a visible `user` item because the handler does not require that role on the last item.

Read and delete stored responses

A non-streaming Responses result contains the current turn's public output:

```
JSON
{
  "id": "resp_0123456789abcdef0123456789ab",
  "object": "response",
  "status": "completed",
  "created_at": 1893456000,
  "model": "hermes-agent",
  "output": [
    {
      "type": "message",
      "role": "assistant",
      "content": [
        {
          "type": "output_text",
          "text": "Three findings are ready for review."
        }
      ]
    }
  ],
  "usage": {
    "input_tokens": 120,
    "output_tokens": 11,
    "total_tokens": 131
  }
}
```

When `store` is `true`, pass its ID to either focused capability:

```
JSON
{
  "response_id": "resp_0123456789abcdef0123456789ab"
}
```

`response_get` returns the stored public response object. It does not return the internal conversation history. `response_delete` removes the stored record and any conversation-name pointer that currently refers to it, then returns:

```
JSON
{
  "id": "resp_0123456789abcdef0123456789ab",
  "object": "response",
  "deleted": true
}
```

Deletion does not undo model calls, tool effects, long-term memory, or session history. It has no compensation capability. Complete its tenant-policy AIP approval and use a required capability-scoped idempotency key before invocation. The approval record has a 900000 ms (15-minute) policy TTL and requires a reason, an input snapshot, and the policy decision as evidence.

The provider DELETE handler does not implement its own idempotent replay. The published 86400000 ms (24-hour) contract and original-result replay belong to the AIP runtime boundary; their durability depends on the runtime idempotency backend.

The connector rejects an empty response ID and percent-encodes the value as one provider path segment. A response created with `store: false` still has an ID, but later GET, deletion, `previous_response_id`, and conversation lookup cannot resolve it.

This family has no response-list operation. If an interrupted creation never returned its `response_id`, do not invent a path or replay the model request. Read the original AIP action result and escalate an unresolved provider

outcome.

Understand response retention

Hermes stores at most 100 Responses objects by default. The store uses least-recently-used eviction, and a successful GET refreshes the access time. It has no time-based expiry. Eviction and deletion also clear a conversation pointer that names the removed response.

The normal store is a SQLite database under `HERMES_HOME` and can survive a gateway restart. Hermes attempts to set the database and existing SQLite sidecars to owner-only mode `0600`. Permission-change failure does not stop startup, so verify the effective filesystem mode.

If Hermes cannot open the file, it falls back to in-memory SQLite; that fallback is lost at restart. AIP does not add cross-instance replication or guarantee that a later request reaches the same response database.

The database contains full conversation history, including tool payloads and results, even though `response_get` exposes only the public response object. Protect the Hermes home directory as confidential application state.

Keep continuity identifiers separate

Identifier	Owner	Purpose
AIP action ID	AIP application and runtime	Identifies one governed invocation and its durable result
AIP session ID	AIP envelope and runtime	Groups native protocol messages and actions
Hermes `session_id`	Chat input and Hermes session database	Continues a chat transcript
Hermes `session_key`	Application and Hermes memory layer	Scopes long-term memory across transcripts
`response_id`	Hermes Responses store	Retrieves or chains one stored Responses object
`conversation`	Application and Hermes Responses store	Names the latest stored response in one chain

The connector can send the AIP session ID as `X-AIP-Session-Id`, but it does not silently copy that value into `X-Hermes-Session-Id`. Set Hermes continuity explicitly when the application needs it.

Trusted execution can also add principal, tenant, authentication issuer, trace, correlation, delegation-chain, and deadline headers. The Hermes endpoint token authenticates the connector host; these context headers do not replace that token or grant provider authority by themselves.

Apply idempotency and retry safely

The AIP contract uses capability-scoped keys and `return_original_result` collision behavior. For `response_delete`, the key is required and retained for 86400000 ms (24 hours) by the declared contract. The other five operations publish an optional key and no TTL.

`return_original_result` does not revalidate a changed input. Reusing one key for a different business request can return the first retained AIP result. Allocate one key per exact intent and keep it with the original action ID.

Hermes also has a narrower provider cache for non-streaming `chat` and `responses`: it is process memory, retains at most 1000 entries for 300 seconds, and reuses only the same key with the same route-specific fingerprint.

The cache reuses the agent execution result but still constructs a fresh HTTP response and response ID. A different fingerprint can execute again. Streaming, retrieval, and deletion do not use this cache. It cannot replace AIP's durable result and key record.

The safe recovery rule is:

1. retain the original AIP action ID and idempotency key;

2. after interruption, read the durable AIP action result first;
3. retry `response_get` only when the typed error permits it;
4. do not resubmit chat, Responses, a stream, or deletion automatically;
5. reconcile provider state or escalate when the result remains unknown.

An optional key does not change the four model operations from retry-unsafe to retry-safe. Model execution can call tools or send messages before the connector observes a transport, deadline, decode, or provider error.

Respect data and execution limits

Limit or classification	Reviewed behavior
Provider request body	At most 10000000 bytes (10 MB)
Provider concurrency	10 agent runs by default across chat, Responses, and runs; `0` disables the cap
Saturation	Provider HTTP 429 with `Retry-After: 1`; the connector's typed failure does not retain that header
Non-streaming chat/Responses HTTP timeout	300 seconds inside the connector
Response GET/delete HTTP timeout	30 seconds inside the connector
Trusted AIP deadline	Can end the operation earlier than the HTTP timeout
Connector response body	No explicit byte limit in this connector path

The capability contract classifies chat and Responses data as confidential, PII-bearing, and redaction-required. It classifies deletion as restricted.

The current generated contract classifies `response_get` as internal and `contains_pii: false` because it is a generic read operation. That declaration is too weak for a stored model or tool result, which can contain confidential or personal data.

Until the contract is corrected, handle `response_get` input, output, storage, logs, and traces as confidential and potentially PII-bearing. The generated contract still requires redaction for this read. It classifies `response_delete` as PII-bearing and redaction-required as well as restricted.

Streaming boundary

`chat_stream` and `responses_stream` force provider streaming and aggregate SSE frames into an object containing endpoint, operation, HTTP status, ordered events, best-effort text, terminal state, and optional truncation state. A positive `max_events` value can stop collection before a terminal event.

Advertised AIP async support does not create provider background delivery after the HTTP turn. Hermes binds this API-server channel with asynchronous delivery disabled; the stream must settle through the active request and AIP lifecycle.

Do not infer completion from a closed client connection or accumulated text. Use the streaming guide for the application procedure and the cursor and cancellation reference for exact terminal, replay, publication, and stop behavior.

Failures and recovery

Symptom	Meaning	Decision
Schema rejection or HTTP 400	Branch, message, `input`, history, or continuity fields are invalid	Correct the request; do not retry unchanged
`connector.hermes_agent.authentication`	Endpoint token was rejected	Correct the endpoint credential; do not broaden caller scopes
`connector.hermes_agent.tenant_mismatch`	Trusted AIP tenant does not own the selected endpoint	Select the admitted tenant route
HTTP 404 on a response ID	The record was never stored, was evicted or deleted, or is absent from the reached database	Check `store`, the ID, the target instance, and the original AIP result
HTTP 413	Provider request exceeded 10000000 bytes	Reduce bounded input and history
HTTP 429	Hermes concurrent-run gate is full	Preserve the action; follow its typed result instead of blindly replaying model work
HTTP 502 with `agent_incomplete`	Hermes produced no usable text after a failed or partial run	Treat tool and model outcome as uncertain
Transport, decode, deadline, or provider 5xx on a model call	Hermes may have begun model or tool work	Read durable AIP state and reconcile; do not auto-retry
Repeated deletion returns 404	The record is already absent or the request reached another store	Read the original approved action result; do not create a new delete attempt blindly

The connector maps temporary and transport failures as retryable only for `response_get`. It conservatively marks uncertain outcomes on non-read invocations. Use protocol error fields rather than matching provider message text.

Verify an integration

Before accepting a chat or Responses path:

1. discover the exact endpoint-qualified capability ID;
2. confirm its risk, stability, execution, approval, key, and data contract;
3. submit one bounded input and retain the AIP action ID and key;
4. inspect finish, partial, failure, and usage fields instead of content alone;
5. for a stored Responses call, read the returned ID through `response_get`;
6. verify that logs and traces redact prompts, outputs, tool data, and IDs;
7. if deletion is required, complete approval, delete once, and retain the

original result as the audit record.

This verifies request mapping and lifecycle handling. It does not establish model quality, tool safety, provider availability, or production qualification.

Related documentation

- Hermes Agent capabilities
- [Authenticate and bind Hermes

endpoints](../getting-started/authentication-and-endpoints.md)

- Actions and sessions
- Approvals and policy
- Errors and retry decisions