Audit record reference

- Common fields
- Categories
- Decisions
- Enforcement modes
- Approval fields
- Action types
- HTTP request payload
- Sample record

Docker AI Governance audit records use one schema across delivery modes. Local JSON Lines files and cloud-delivered records contain the same metadata fields.

Records capture metadata only. They don't contain prompt content, agent output, or parameter values. Parameter keys may appear when they help identify an action.

Common fields

Field	Description
`audit_event_id`	Unique ID for the audit event.
`timestamp`	UTC time when Docker recorded the event.
`schema_version`	Version of the record schema. Pin SIEM field mappings to this value.
`category`	Event category, such as `AUDIT_CATEGORY_MANAGEMENT`, `AUDIT_CATEGORY_EVALUATION`, or `AUDIT_CATEGORY_EXECUTION`.
`decision`	Governance decision for evaluation records.
`username`	The signed-in Docker user's Docker Hub username.
`user_email`	The signed-in Docker user's email address.
`org_id`	ID of the organization whose governance policy is in effect.

Field	Description
`org_name`	Display name of the organization whose governance policy is in effect.
`audit_session_id`	Identifies the daemon session that produced the record.
`resource_id`	Target of the evaluation, such as a host and port, file path, or tool.
`os`	Operating system that produced the record.
`app_version`	Version of the Docker component that produced the record.
`client_name`	Source component, such as `sbx` for Docker Sandboxes.
`hostname`	Hostname of the machine that produced the record.
`deny_reason`	Why a denied request was blocked. Present on deny decisions.
`enforcement_mode`	Governance mode in effect when Docker evaluated the request. See Enforcement modes.
`approval`	Approval details. Present when the decision involves an approval. See Approval fields.
`action_type`	Payload discriminator that identifies the action-specific object in the record.
`agent`	AI agent associated with the event, when Docker knows it.

Categories

Category	Description
`AUDIT_CATEGORY_MANAGEMENT`	Session lifecycle, policy sync, and configuration events.
`AUDIT_CATEGORY_EVALUATION`	Governance policy decisions, such as allow, deny, or consent.
`AUDIT_CATEGORY_EXECUTION`	Outcomes after an evaluated action runs.

Decisions

Decision	Description
`AUDIT_DECISION_ALLOW`	Docker allowed the action.
`AUDIT_DECISION_DENY`	Docker denied the action.

Decision	Description
`AUDIT_DECISION_APPROVAL_REQUIRED`	The policy requires a user to approve the action.
`AUDIT_DECISION_APPROVAL_ALLOW`	A user approved a pending request.
`AUDIT_DECISION_APPROVAL_DENY`	A user denied a pending request.

Enforcement modes

The `enforcement_mode` field shows how governance applied to the request when Docker evaluated it.

Mode	Description
`enforce`	Your organization's governance policy is enforced.
`audit`	Docker records the decision for observability but doesn't enforce policy.
`off`	Governance isn't active. Docker still writes the record to the local audit log.

A record that resolves a network approval doesn't include `enforcement_mode`, and neither does the record of a declined MCP approval. An approved MCP request is evaluated again, so its record includes `enforcement_mode`.

Approval fields

When network access requires approval, Docker blocks the request and writes an `AUDIT_DECISION_APPROVAL_REQUIRED` record. If the user approves, Docker writes a second record with `AUDIT_DECISION_APPROVAL_ALLOW`. Both records carry the same `approval.approval_request_id`, so you can join them to reconstruct the full approval. The approval applies to later requests from the sandbox. It doesn't retry the request that was blocked. If the user dismisses the request, it isn't resolved, so no second record is written and the destination is requested again the next time the sandbox reaches it. For how approval works, see Approval-required access.

For MCP approvals, an approved request is evaluated again and recorded with `AUDIT_DECISION_APPROVAL_ALLOW`, and a declined request is recorded with `AUDIT_DECISION_APPROVAL_DENY`. Neither record includes `approval.approval_request_id` or `approval.grant_scope`.

Field	Description

`approval.reason_codes`	Policy reasons for requiring approval. Present on `AUDIT_DECISION_APPROVAL_REQUIRED` records when the policy supplies reasons, and on declined MCP approvals when reasons are available.
`approval.approval_request_id`	ID that links an approval required record to the record that resolves it.
`approval.grant_scope`	Scope the user accepted. Present only on `AUDIT_DECISION_APPROVAL_ALLOW` records.
`approval.context_digest`	SHA-256 digest of the request that was resolved, which shows it matches the request that was held. Present only on declined MCP approvals.

For network approvals, Docker Sandboxes records `approval.grant_scope` as `AUDIT_APPROVAL_GRANT_SCOPE_PERSISTENT`, which means that the approval applies to later requests.

If a user approves a request but Docker can't apply the grant, the record has an `AUDIT_DECISION_DENY` decision and includes the `approval` object.

Action types

The `action_type` field identifies the action-specific payload in the record.

Action type	Description
`session`	Sandbox daemon session lifecycle event.
`network_egress`	Network access evaluation.
`http_request`	HTTP request evaluation. See HTTP request payload.
`filesystem_mount`	Filesystem mount or path access evaluation.
`tool_invocation`	Tool invocation evaluation.
`resource_read`	Resource read evaluation.
`server_registration`	Server registration event.
`prompt`	Prompt-related metadata event.
`network_egress_execution`	Outcome of a network action.
`filesystem_mount_execution`	Outcome of a filesystem action.
`tool_execution`	Outcome of a tool invocation.
`resource_read_execution`	Outcome of a resource read.
`prompt_execution`	Outcome of a prompt request.
`policy_sync`	Policy synchronization event.

Action type	Description
`pii_detection`	Metadata event for a data detection result.
`c_score_report`	Metadata event for a C-score report.
`policy_action`	Policy configuration or policy action event.

HTTP request payload

When a network policy requires each HTTP request to be authorized, Docker evaluates the request method and path in addition to the network connection. These evaluations produce records with an `action_type` of `http_request`, separate from the `network_egress` record for the connection.

Field	Description
`http_request.method`	HTTP method, such as `HTTP_METHOD_GET` or `HTTP_METHOD_POST`.
`http_request.host`	Destination hostname or IP address. IPv6 addresses appear without brackets.
`http_request.port`	Destination port.
`http_request.path`	URL path, without the query string or fragment.

Sample record


```
{
  "audit_event_id": "95e7257f-93c9-4f29-bde7-88830e2dae80",
  "timestamp": "2026-05-28T19:15:00.728933Z",
  "schema_version": "1.82.0",
  "category": "AUDIT_CATEGORY_EVALUATION",
  "decision": "AUDIT_DECISION_DENY",
  "username": "jordandoe",
  "user_email": "jordandoe@example.com",
  "org_id": "9f8e7d6c-5b4a-3210-fedc-ba9876543210",
  "org_name": "Acme Inc",
  "audit_session_id": "8a3bc076-79d0-4502-baf3-cc6ad35fb578",
  "resource_id": "example.com:443",
  "os": "macos",
  "app_version": "v0.31.0",
  "client_name": "sbx",
```

```
  "hostname": "host-machine",
  "deny_reason": [
    "no applicable policies for op(action=net:connect:tcp,
resource=net:domain:example.com:443)"
  ],
  "action_type": "network_egress",
  "network_egress": { "protocol": "tcp" },
  "agent": "claude"
}
```


The following sample shows a user approving an HTTP request that required approval:


```
{
  "audit_event_id": "3c1f6a2e-8b47-4d0a-9e55-2f7b1d6c9a04",
  "timestamp": "2026-09-25T17:42:08.114502Z",
  "schema_version": "1.216.0",
  "category": "AUDIT_CATEGORY_EVALUATION",
  "decision": "AUDIT_DECISION_APPROVAL_ALLOW",
  "username": "jordandoe",
  "user_email": "jordandoe@example.com",
  "org_id": "9f8e7d6c-5b4a-3210-fedc-ba9876543210",
  "org_name": "Acme Inc",
  "audit_session_id": "8a3bc076-79d0-4502-baf3-cc6ad35fb578",
  "resource_id": "api.example.com:443",
  "os": "macos",
  "app_version": "v0.46.0",
  "client_name": "sbx",
  "hostname": "host-machine",
  "approval": {
    "approval_request_id": "b7e2c9d1-4f60-4a8e-93c2-5d1e8f7a6b30",
    "grant_scope": "AUDIT_APPROVAL_GRANT_SCOPE_PERSISTENT"
  },
  "action_type": "http_request",
  "http_request": {
    "method": "HTTP_METHOD_POST",
    "host": "api.example.com",
    "port": "443",
    "path": "/v1/items"
  },
  "agent": "claude"
}
```

□□□ #2
□ □□□ □□□ 2026-10-31 19:01:15 UTC
□ □□□ □□□ 2026-10-31 19:01:17 UTC