Wazuh Ingest API (Agent-Ingest) — Documentation

This project runs a Wazuh agent plus a small HTTP API that accepts JSON events and forwards them into the local Wazuh agent queue. The Wazuh agent then forwards them to the manager like normal agent data.

Components

- **Ingestion API (FastAPI)**: Receives JSON over HTTP and forwards it to the Wazuh queue socket.
- **Wazuh agent**: Enrolled to a Wazuh manager and forwards events upstream.
- **Readiness webserver**: A simple HTTP server that serves `ready.html` when the agent is connected.

Network Ports

- **9000/tcp**: FastAPI ingestion service.
- **9001/tcp** (internal): Readiness static webserver (serves `ready.html` when connected). The port is configurable using `READY_PORT`.

Authentication

The API supports optional API key authentication:

- Request header: `X-API-Key`

- Environment variable: `API_KEY`

Behavior:

- If `API_KEY` is **not** set or is empty, all requests are allowed.
- If `API_KEY` is set, requests must include `X-API-Key: <API_KEY>` or the API returns `403 {"detail":"Could not validate credentials"}`.

Endpoints

GET /health

Returns whether the Wazuh queue socket is available inside the container.

Example:

```
curl -sS http://<agent-ip>:9000/health
```

Response:

- `{"status":"healthy","wazuh_socket":"connected"}` when `/var/ossec/queue/sockets/queue` exists.
- `{"status":"unhealthy","wazuh_socket":"disconnected"}` otherwise.

PUT /

Ingest a single JSON event. The payload must be a JSON object.

Example (no API key):

```
curl -X PUT "http://<agent-ip>:9000/" \
  -H "Content-Type: application/json" \
  -d '{"message":"UNIQUE_TEST_12345","event_type":"api_test"}'
```

Example (with API key):

```
curl -X PUT "http://<agent-ip>:9000/" \
  -H "Content-Type: application/json" \
  -H "X-API-Key: <your-api-key>" \
  -d '{"message":"UNIQUE_TEST_12345","event_type":"api_test"}'
```

Successful response:

```
{"status":"success","message":"Event sent to Wazuh"}
```

Error responses:

- `{"status":"error","message":"Wazuh agent is not running or socket is missing"}`
- `{"status":"error","message":"Message too long"}`
- `{"status":"error","message":"Wazuh must be running."}`

PUT /batch

Ingest multiple events in one request. The payload must be a JSON array of objects.

Example:

```
curl -X PUT "http://<agent-ip>:9000/batch" \
  -H "Content-Type: application/json" \
  -H "X-API-Key: <your-api-key>" \
  -d '[{"message":"one","event_type":"api_test"},{"message":"two","event
```

Response:

```
{
  "status": "batch_processed",
  "total": 2,
  "errors": 0,
  "details": [
    {"status":"success","message":"Event sent to Wazuh"},
    {"status":"success","message":"Event sent to Wazuh"}
  ]
}
```

Event Forwarding Format

The API forwards events to the Wazuh agent queue socket:

- Socket path: `/var/ossec/queue/sockets/queue`
- Socket type: Unix datagram socket (`AF_UNIX`, `SOCK_DGRAM`)

- Message format:

- A decoder header prefix, then JSON payload

- Default header is controlled by `WAZUH_DECODER_HEADER`

Decoder Header Selection

The header is decided as follows:

- If the incoming JSON payload includes a `decoder` field, the header becomes:

- `1:<decoder>:`

- Otherwise, the header defaults to:

- `WAZUH_DECODER_HEADER` (default: `1:Wazuh-AWS:`)

The API also adds:

- `ingest: "api"`

Example payload:

```
{"decoder":"json","message":"UNIQUE_TEST_12345","event_type":"api_test"}
```

Results in (conceptually):

```
1:json:{"decoder":"json","message":"UNIQUE_TEST_12345","event_type":"api
```

Environment Variables

Wazuh Enrollment / Connection

- `MANAGER_URL`: Manager host/IP used for enrollment.
- `MANAGER_PORT`: Enrollment port (typically `1515/tcp`).
- `SERVER_URL`: Manager/worker host/IP used for agent data channel.
- `SERVER_PORT`: Agent data port (typically `1514/tcp`).
- `NAME`: Agent name (should be stable and unique).
- `GROUP`: Agent group.
- `ENROL_TOKEN`: Optional enrollment password/token. If empty, the container removes `authd.pass`.

API

- `API_KEY`: Optional API key required via `X-API-Key` if set.
- `API_PORT`: FastAPI listen port (default `9000`).
- `WAZUH_DECODER_HEADER`: Default decoder header prefix if request doesn't include `decoder`.

Readiness

- `READY_PORT`: Readiness static webserver port (default `9001`).

How to Test End-to-End

1) Confirm API health

```
curl -sS http://<agent-ip>:9000/health
```

2) Send a unique test event

```
curl -X PUT "http://<agent-ip>:9000/" \
  -H "Content-Type: application/json" \
  -H "X-API-Key: <your-api-key>" \
  -d '{"decoder":"json","message":"UNIQUE_TEST_12345","event_type":"api_
```

3) Confirm the agent is connected and sending

```
docker exec -it wazuh-log-pipeline-agent-ingest-1 bash -lc "cat /var/oss
```

4) Confirm the manager received the event

On the manager, search archives:

```
sudo grep -R "UNIQUE_TEST_12345" /var/ossec/logs/archives/archives.json
```

5) Make it visible as an alert in the dashboard (recommended)

Many dashboards focus on `wazuh-alerts-*` indices, which require Wazuh rules to generate alerts.

Create a local rule that matches the ingested JSON fields (example):

```
<group name="api_ingest,">
  <rule id="100200" level="5">
    <decoded_as>json</decoded_as>
    <field name="data.event_type">api_test</field>
    <field name="data.ingest">api</field>
    <description>API ingest event (api_test)</description>
  </rule>
</group>
```

After adding the rule, restart the manager and resend the event. Then in Discover, search:

- `data.message: UNIQUE_TEST_12345`
- `data.event_type: api_test`
- `rule.id: 100200`

Operational Notes

- If you see HTML “File not found” when calling `/health`, you are hitting the readiness static webserver, not FastAPI. FastAPI health is on port 9000.
- Keep `NAME` stable across restarts to avoid creating duplicate agents.
- Persist `/var/ossec/etc` if you want stable agent keys across container recreation.