Work Order AI Inspections Service

Usage guide for the AI video-inspection vendor

opsenseapi · Version 1.1 · 18 September 2026

1. How you will use this service

You will receive a `work_order_completed` webhook, containing a `work_order_id`, as soon as a work order finishes. On receiving it:

1. Call **Get Inspection Queue** only if you need to recover work orders you may have missed a webhook for — it is not part of the normal flow.
2. Fetch the two videos for that work order (the URLs are already public once you have the references).
3. Analyse them and build your existing report JSON.
4. Call **Submit Inspection** with the report. This is the only call required for normal operation.

Nobody on our side reviews or approves a submission before it is processed — treat this as a fully automated integration.

2. Authentication

Every call to either service below requires your `api_key`, sent as a normal request parameter (query string on a GET call, POST field on a POST call) — not as an HTTP header.

Situation	Response
`api_key` missing	`error_code 10015`
`api_key` invalid or inactive	`error_code 10016`

Every response is wrapped the same way:

```
{"result": { ... success or error body ... }}
```

3. Get Inspection Queue (recovery use only)

Lists your completed work orders that have both videos but no inspection stored yet for their current videos. If a video is replaced after you already submitted a report for it, the work order reappears here automatically.

```
GET https://<api-host>/v1/SRVGetInspectionQueue
```

Request parameters

Field	Required	Notes

`api_key`	**Required**	See Authentication.
`limit`	Optional	Default 50, maximum 200.
`since`	Optional	Format `Y-m-d H:i:s`. Filters on the work order's completion date.

Response fields (per row)

Field	Description
`work_order_id`	Use this value when you call Submit Inspection.
`room_name`, `branch_name`	For your own logging/UI only.
`end_date`	When the work order was completed.
`video_before_ref`, `video_after_ref`	Echo these back exactly when you call Submit Inspection.
`video_before_url`, `video_after_url`	Ready-to-download URLs for the two videos.

Example

```
curl -G "https://<api-host>/v1/SRVGetInspectionQueue" \ --data-urlencode
"api_key=YOUR_ACCESS_TOKEN" \ --data-urlencode "limit=50"
```


```
{"result":{"result":"success","data":[ { "work_order_id": 12345, "room_name": "1.17 Highbury",
"branch_name": "Highbury House", "end_date": "2026-09-18 09:40:00", "video_before_ref":
"r2:before_12345_ab12cd.mp4", "video_after_ref": "r2:proof_12345_ef34gh.mp4",
"video_before_url": "https://opsense-cdn.co.uk/before_12345_ab12cd.mp4", "video_after_url":
"https://opsense-cdn.co.uk/proof_12345_ef34gh.mp4" } ]}}
```


4. Submit Inspection (the call you make for every completed work order)

Sends one inspection report for one work order. No batching — one report per call.

```
POST https://<api-host>/v1/SRVSubmitInspection
```

Request fields

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`work_order_id`	**Required**	Positive integer, from the webhook or the queue.

`report`	**Required**	Your existing report JSON, **as a JSON-encoded string field** — max 512 KB. Must contain a `scores` object.
`video_before_ref`, `video_after_ref`	Optional, recommended	Send the exact values you received from the webhook/queue. Lets a re-submission after a video swap be told apart from a duplicate call.
`model`	Optional, recommended	Your model / prompt version, so score changes over time can be tracked against it.
`analysed_at`	Optional	ISO-8601 or any common date format. If omitted, the current time is used.
`report_id`	Optional	Your own report id, stored for your reference.

Safe to retry: calling again with the same `work_order_id` + `video_before_ref` + `video_after_ref` + `model` returns the original result with `"duplicate": true` instead of creating a second entry — retry freely on a timeout or network error.

Success response

```
{"result":{"result":"1","inspection_id":842,"issues":2,"room_ref_match":1}}
```

Duplicate (safe retry)

```
{"result":{"result":"1","duplicate":true,"inspection_id":842}}
```

Error response shape

```
{"result":{"error_code":"10033","error_text":"report must be a JSON object with a scores key."}}
```

Error codes

Code	Meaning
10015	Missing `api_key`.
10016	`api_key` invalid or inactive.
10032	Missing or invalid `work_order_id`.
10033	`report` is not valid JSON, or has no `scores` object.
10034	Work order not found for your account.
10035	More than 120 submissions in the last minute — slow down and retry shortly.
10036	`report` exceeds the 512 KB size limit.

10037	Could not be saved — safe to retry.

Example

```
curl "https://<api-host>/v1/SRVSubmitInspection" \ --data-urlencode
"api_key=YOUR_ACCESS_TOKEN" \ --data-urlencode "work_order_id=12345" \ --data-urlencode
"video_before_ref=r2:before_12345_ab12cd.mp4" \ --data-urlencode
"video_after_ref=r2:proof_12345_ef34gh.mp4" \ --data-urlencode "model=gemini-2.x / prompt-v4"
\ --data-urlencode "analysed_at=2026-09-18T10:22:00Z" \ --data-urlencode
"report_id=optional-vendor-uuid" \ --data-urlencode "report=$(cat report.json)"
```

5. After you submit

Your full report is kept for audit, and the actionable parts (issues, non-"none" checklist actions, coverage gaps, missing/left items) drive follow-up work on our side. If your report has `scores.requiresReview: true`, or contains any actionable item, the relevant supervisor is notified automatically — no further action is needed from you beyond a successful submission.