MISP

INTELLIGENCE SHARING

Working with the MISP 2.5 REST API

Query, interpret and automate

CIRCL / MISP Project

https://www.misp-project.org/

Reference snapshot: 29 October 2026

A useful integration asks a precise question

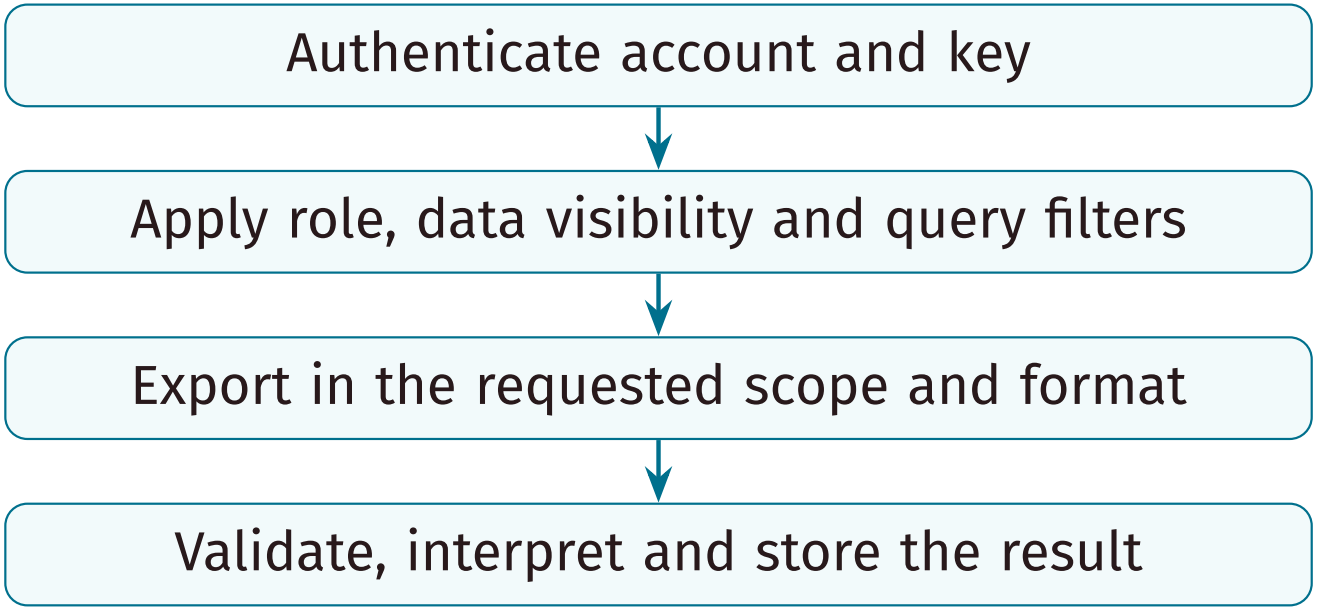
- Authenticate with the right account and key.
- Choose event or attribute scope.
- Set filters and export policy explicitly.
- Traverse pages without losing records.
- Handle updates, failures and removals deliberately.

Reference snapshot

MISP 2.5.50, reviewed 29 October 2026. Check your deployed version.

Make one request

The API uses your account’s visibility



Set up a key for the task

- The account needs API access: `perm_auth`.
- Send the raw key in `Authorization`.
- Advanced keys can restrict expiry, source IP and writes.
- Keep keys outside scripts, notebooks and shared collections.

Read-only does not mean GET-only

A read-only key can POST a search, while event creation is denied.

First request: inspect the instance

```
curl --fail-with-body --silent --show-error \
  --connect-timeout 5 --max-time 60 \
  -H "Authorization: $MISP_API_KEY" \
  -H 'Accept: application/json' \
  "$MISP_URL/servers/getVersion"
```

Read `version` and `pymisp_recommended_version`. Keep TLS verification enabled; supply your organisation's CA bundle if needed.

Choose the endpoint by the result you need

Endpoint	Result
`/events/index`	Event summaries
`/events/restSearch`	Full events or metadata
`/attributes/restSearch`	Matching attributes
`/events/view/{uuid}`	One known event

Use POST with a JSON body for search. Read each endpoint's contract.

A bounded attribute request

```
{
  "returnFormat": "json",
  "type": "domain",
  "value": "%.example",
  "published": [0, 1],
  "to_ids": [0, 1],
  "includeEventTags": true,
  "limit": 100,
  "page": 1
}
```

POST /attributes/restSearch

The response envelope depends on the scope

Attribute search

```
{"response": {"Attribute": [
  {"uuid": "...", "type": "domain", "value": "..."}
]}}
```

Event search

```
{"response": [
  {"Event": {"uuid": "...", "info": "..."}}
]}
```

Search with intent

A matching event can contain other attributes

Event scope

Find an event through a domain match; full JSON can also include the event's IP address and contextual attributes.

Attribute scope

Return the attributes that match the attribute query.

Use `metadata: true` for lighter event results.

Exact values and wildcard values are different

Filter value	Intent
`workshop.example`	Exact attribute value
`%.example`	Suffix match
`%REST API training%`	Event-info substring

Start with one known value or UUID. Add filters one at a time.

Select publication and detection flags explicitly

Raw JSON filter	Meaning
`published: 1`	Published events
`published: 0`	Unpublished events visible to the caller
`to_ids: 1`	Detection-marked attributes
`to_ids: 0`	Contextual attributes
`[0, 1]`	Either state, for these raw JSON filters

Express tag logic with explicit operators

```
{
  "tags": {
    "AND": ["tlp:clear"],
    "OR": ["training:reviewed", "training:verified"],
    "NOT": ["training:exclude"]
  }
}
```

Require the AND tag, one OR tag, and no NOT tag.

Tags may come from the event

- Event and attribute tags can both influence a match.
- `includeEventTags` adds inherited event tags to the result.
- Inspect the `inherited` marker before interpreting attachment.

Keep context when exporting

A tag-driven match does not mean that tag was attached directly to every returned attribute.

Pick the clock that answers your question

Field	Clock
`timestamp`	Change time in the selected scope
`event_timestamp`	Parent event change time
`publish_timestamp`, `last`	Event publication time
`from`, `to`	Event date bounds
`first_seen`, `last_seen`	Observation time

Changing the format can change the selection

JSON / CSV

Non-restrictive exporters in this snapshot. State selection explicitly when comparing results.

Detection text

Defaults to published events and detection-marked attributes when those filters are omitted.

An analysis result is not automatically a deployment policy.

A detection-policy exercise

```
{
  "returnFormat": "text",
  "type": "domain", "value": "%.example",
  "published": 1, "to_ids": 1,
  "tags": {"AND": ["tlp:clear"],
           "NOT": ["training:exclude"]},
  "enforceWarninglist": true,
  "limit": 100, "page": 1
}
```

Output is text. The unpublished workshop event correctly contributes nothing.

Warninglist annotations and filtering differ

Annotate

`includeWarninglistHits` adds matching-list information to JSON.

Filter

`enforceWarninglist` can remove rows after query pagination.

Empty formatted output is not always proof that later pages have no candidates.

Automate and verify

A role cap can make every page short

Requested	Returned	Continue?
Page 1, limit 100	2	Yes
Page 2, limit 100	2	Yes
Page 3, limit 100	1	Yes
Page 4, limit 100	0	Stop for this plain JSON query

Five records were available. Stopping at the first short page loses three.

Bound the traversal and reject incomplete results

- Start at page 1; request a deterministic order.
- Validate the envelope and attribute UUIDs.
- Continue through short pages.
- Fail on repeated UUIDs or a maximum-page bound.
- Publish output and checkpoints only after success.

Use the supplied read-only Python client

```
python examples/search_attributes.py \
  --page-size 100 --max-pages 20 \
  > attributes.ndjson.tmp \
  && mv attributes.ndjson.tmp attributes.ndjson
```

- Environment-provided URL and key; trusted TLS.
- Timeouts, no redirects, JSON/envelope checks.
- A failure leaves partial output unpublished.

PyMISP offers a typed view of the same task

```
rows = misp.search(
    controller="attributes", return_format="json",
    type_attribute="domain", value="%.example",
    include_event_tags=True, include_event_uuid=True,
    limit=100, page=1, pythonify=True,
)
```

Check for an error dictionary before treating the result as a list of objects.
This is one page, not a complete export.

Keep one fixed window for an incremental run

1. Capture an upper timestamp at run start.
2. Set the lower bound from the last successful run, with overlap.
3. Fetch every page using the same inclusive bounds.
4. Upsert by UUID and complete downstream writes.
5. Advance the checkpoint only when the whole run succeeds.

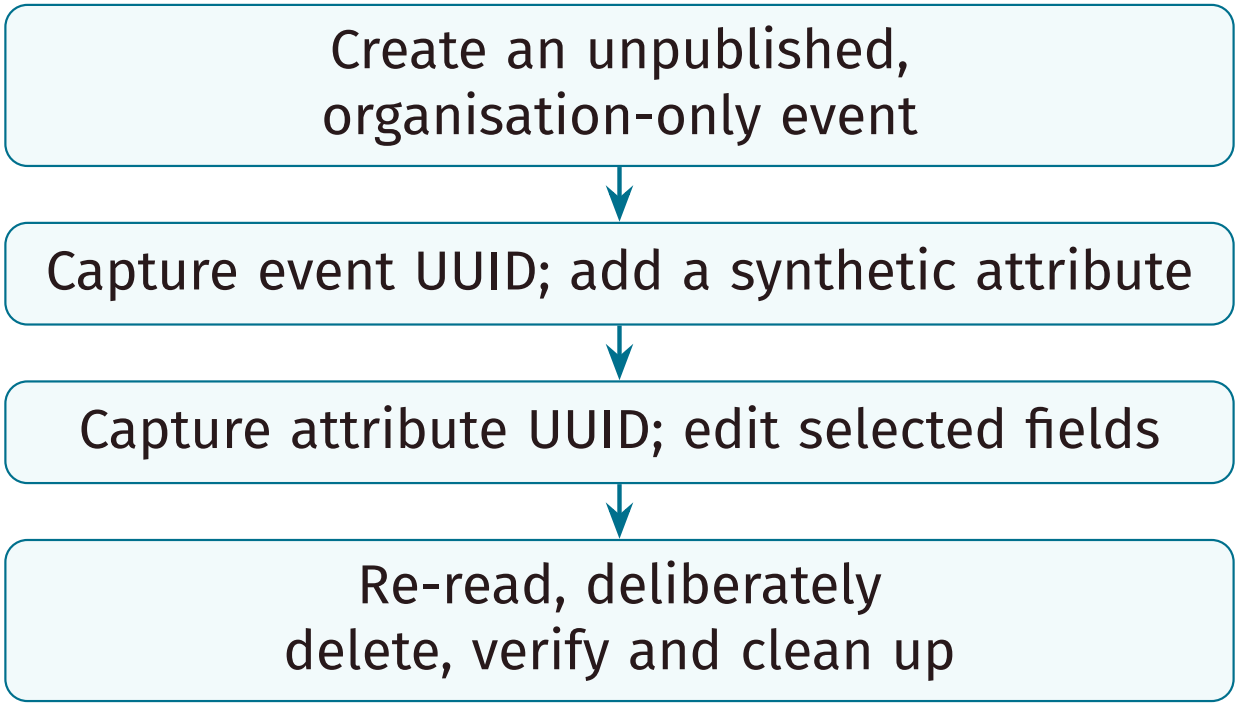
Plan for removals and lost visibility

- Normal searches omit soft-deleted attributes.
- Tombstone visibility depends on permissions and endpoint.
- Hard deletion and lost access may provide no visible tombstone.
- Consumers need reconciliation, expiry and removal rules.

An empty deletion search proves very little

Check the caller’s actual deletion-view contract.

Write a small, explicit change

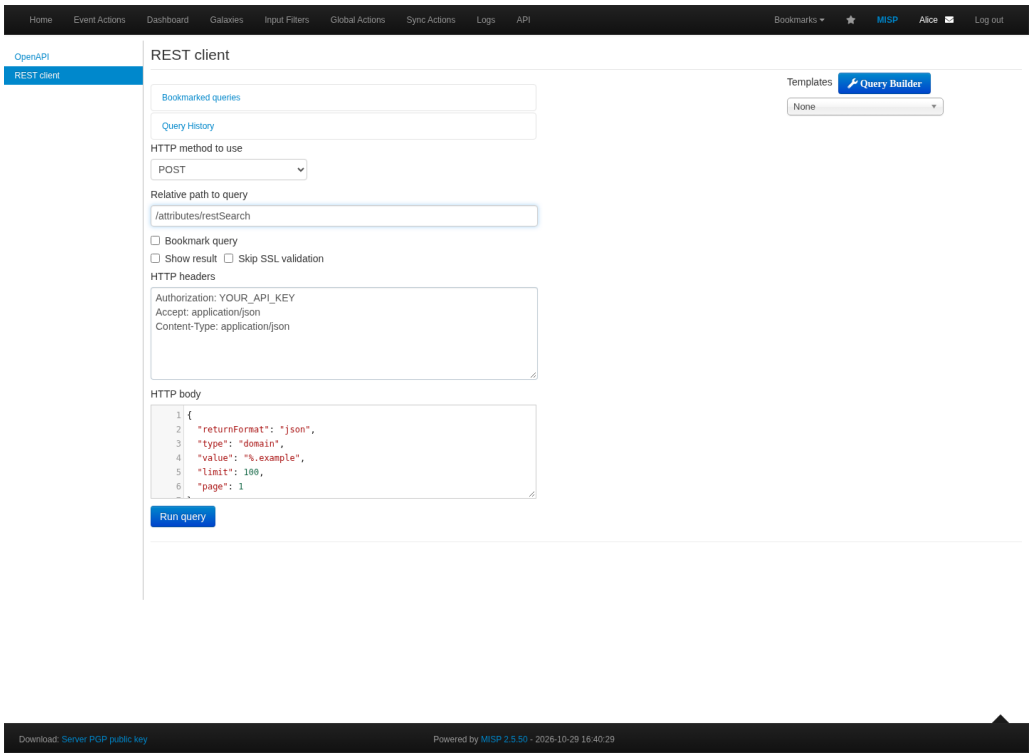


Update only the fields you intend

```
{
  "Attribute": {
    "comment": "Reviewed in the REST API workshop.",
    "to_ids": false
  }
}
```

`POST /attributes/edit/{attribute-uuid}` Re-read the attribute and verify the comment.

Use the built-in tools to inspect a request



- `/api/rest`
- Templates and query builder.
- History and bookmarks.
- Generated request code.

Try the read-only notebook and Postman collection

Notebook

Raw page, bounded traversal and typed PyMISP page. No saved keys or outputs.

Postman

Version, attribute JSON, event metadata, policy text and CSV: five requests, ten assertions.

Keep the filled Postman environment and notebook results local.

Classify failure before retrying

Symptom	Next check
401 / 403	Key, expiry, IP, role and endpoint permissions
404	Base path, UUID, existence and caller visibility
400 / 405	Method, JSON shape, types and validation
HTML / redirect	Login or proxy page; do not treat as empty data
429 / 5xx / timeout	Bounded read retry; reconcile uncertain writes
200, no rows	Scope, flags, tags, clocks, ACL and exporter defaults

Workshop: prove the boundaries

1. Create the private fixture and capture its UUIDs.
2. Query it as the owning organisation's reader.
3. Compare another organisation and a read-only key.
4. Traverse the role-capped pages.
5. Edit, re-read and deliberately soft-delete the attribute.
6. Inspect the owner deletion view; remove the lab event.

Evidence an integration reviewer can reproduce

- Fourteen offline client-boundary tests.
- Live MISP checks for ACLs, paging, filters and writes.
- Newman assertions and an executable read-only notebook.
- Pinned MISP/PyMISP sources and documented limits.

Continue from the repository

See `a.7-rest-API/README.md` for commands, source audit and the complete workshop.