MISP

INTELLIGENCE SHARING

MISP objects and templates in 2.5

Model observations, validate instances, contribute templates

CIRCL / MISP Project

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

Reference snapshot: 29 October 2026

Model meaning, not just a bag of values

- Choose an existing model before inventing one.
- Separate template identity from instance identity.
- Validate definitions, instances and sharing policy.
- Practise an offline example and a contribution workflow.

Understand the model

Attribute, object and template

- An attribute is a typed observation with its own UUID.
- An object groups attributes describing one entity or observation.
- A template describes relation names, types and constraints.

Three identifiers with different jobs

- `uuid`: the object instance.
- `template_uuid`: the reusable model.
- `template_version`: the revision used to construct it.

Reuse the current repository

- Inspect the actual definition, not just its name.
- Look under `objects/<name>/definition.json`.
- Compare the model with your source data and downstream consumers.

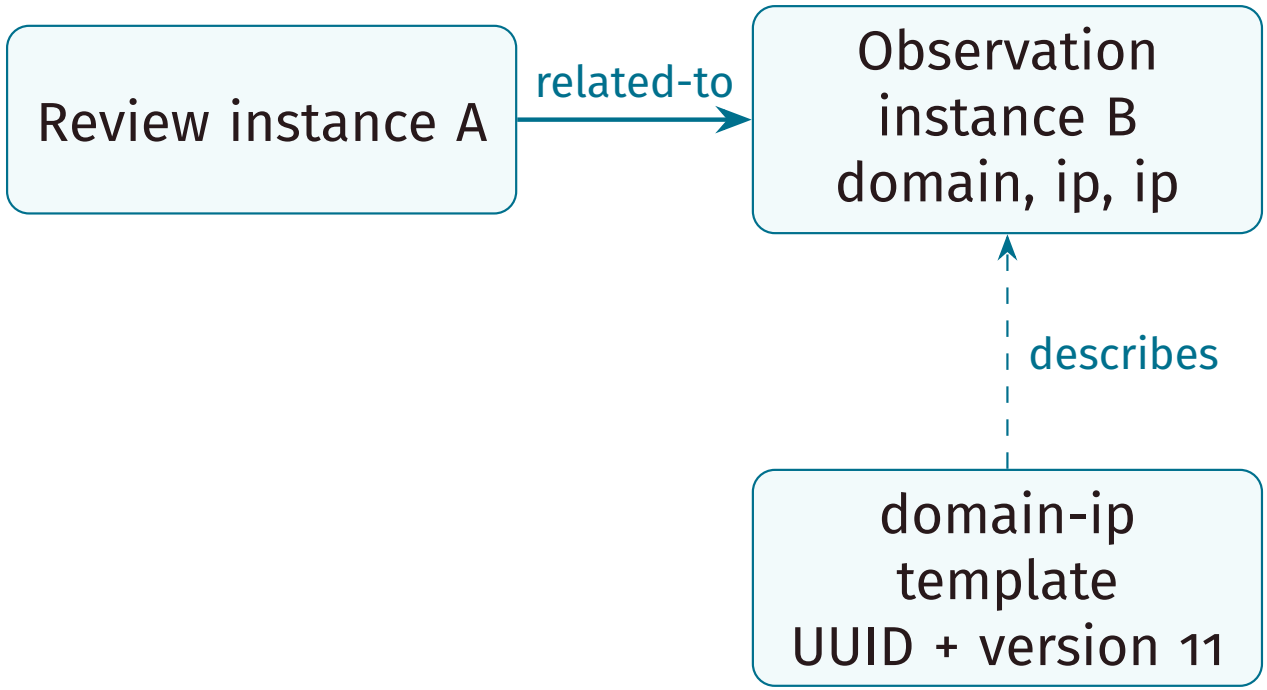
Current domain-ip: alternatives, not a mandatory pair

- `requiredOneOf`: ip, domain or hostname.
- Repeated domain and IP fields are supported.
- Use separate instances for different observation contexts.

Field relation is not a graph relationship

- `ip`: a field relation inside an object.
- `ip-dst`: the underlying MISP attribute type.
- `related-to`: a directed reference between instances.

Reference instances, not definitions



Build and validate

A definition has its own identity

```
{
  "name": "training-review",
  "uuid": "<fresh template UUID>",
  "version": 1,
  "meta-category": "misc",
  "description": "Teaching-only review record",
  "required": ["case-reference"],
  "requiredOneOf": ["domain", "ip"],
  "attributes": { "...": "relation definitions" }
}
```

A relation maps meaning to an existing type

```
"status": {
  "description": "Review status.",
  "misp-attribute": "text",
  "ui-priority": 1,
  "values_list": ["pending", "reviewed"],
  "to_ids": false,
  "disable_correlation": true
}
```

Required fields and repeated fields

- Every relation in required must be present.
- At least one relation in requiredOneOf must be present.
- Both rules can apply together; multiple permits repetition.

Vocabulary and suggestions are different

- `values_list` drives a vocabulary selector.
- `sane_default` offers suggestions plus manual input.
- Validate vocabulary membership explicitly when your workflow requires it.

UI priority is implemented in current MISP

- Higher-priority relations are ordered first in current object views.
- Low-priority fields can keep secondary details out of the main focus.
- Priority is presentation metadata, not a requirement or access rule.

Detection, correlation and visibility are separate

- Choose `to_ids` from the observation and deployment purpose.
- Choose correlation behaviour from the meaning of the value.
- Review event/object/attribute distribution and sharing context.

Four validation layers

- Definition schema: recognised fields and types.
- Definition semantics: identity, requirements and compatibility.
- Instance conformity: required and repeatable relations.
- Attribute validity and policy: values, flags and permissions.

Template enforcement depends on the path

- Template-backed object creation checks conformity.
- Event capture, editing and synchronisation have different paths.
- Strict client construction is useful evidence, not universal server enforcement.

Load the selected definition explicitly

```
definition = json.loads(definition_path.read_text())
obj = MISPObject(
    definition["name"], strict=True, standalone=False,
    misp_objects_template_custom=definition,
)
obj.add_attribute("domain", value="workshop.example",
                  to_ids=False)
obj.to_json(strict=True)
```

The offline workshop event

- One domain/IP object: one domain, two IPs.
- One teaching review: case reference, domain and review status.
- A related-to edge points from review to observation.

Validate, then inspect the output

```
python examples/verify_examples.py /tmp/misp-objects-lab
python examples/build_examples.py /tmp/misp-objects-lab \
  > training-event.json
```

Make failure cases part of the exercise

- Remove the required case reference.
- Remove all requiredOneOf alternatives.
- Repeat a unique relation or add an unknown relation.
- Misspell a schema field and observe the schema failure.

Deploy and contribute

Disk definitions and installed templates are separate

- Update the selected Git checkout deliberately.
- Import/update the template database through the operator workflow.
- Confirm the intended UUID and revision are available to users.

Unknown templates and revision upgrades

- Preserve the received attributes and template metadata.
- Normal UI editing needs a suitable definition.
- Review recovery/upgrade mappings and unmatched fields.

A controlled MISP lab

- Use an unpublished organisation-only event and an authorised writer.
- Add the official object, inspect fields and save.
- Link instances and export/read back the event.
- Test conformity through the actual UI/API path and clean up.

Evolve a model without losing its identity

- Existing model: retain UUID and increment version.
- New model: fresh UUID and initial version.
- Document changed semantics, requirements and consumer impact.

Contribute to the current repository

- Explain why existing templates do not cover the use case.
- Provide schema-valid JSON and synthetic positive/negative instances.
- Check formatting, UUID uniqueness and relationship consistency.
- Open a focused PR with compatibility and validation evidence.

Follow the upstream validation workflow

```
./jq_all_the_things.sh
# Review changes and commit normalised JSON.
./validate_all.sh
```

Make the result reviewable

- Show the model, an instance and the intended relationship direction.
- State the exact template and software revisions tested.
- Explain validation limits and migration choices.
- Keep real credentials and incident data out of examples.