

ODA3 INSTITUTE

IMPLEMENTATION GUIDE

UAIF Implementation Guide

Consolidated Guide to Implementing the Canonical UAIF™ v1.0 Schema

IMP-UAIF-037 | Public

© ODA3 Pvt Ltd

Document Control

Item	Value
Document Identifier	IMP-UAIF-037 (legacy identifier retained for citation continuity)
Doc Type	Implementation Guide - Consolidated
Framework Tag	UAIF™ v1.0
Canonical Schema	uaif-core-v1.0.json (JSON Schema Draft 2020-12)
Canonical SHA-256	55056e7cfa4fcf4b76cc4656833cb5b360a1b40fa90f23488d488409ccebeeca
Authoritative References	ODA3-2026-06-TCR-STD-002; ODA3-2026-06-TCR-STD-003; TCR-UAIF-014; TCR-UAIF-015; TCR-UAIF-018; IMP-UAIF-043
Publication History	Issued 14 July 2026; updated 04 October 2026
Review Cycle	On every schema or reference-package correction; otherwise annual
Audience	Engineering teams integrating UAIF; implementers; standards-body reviewers
Licence	GAISSF Ecosystem Licence (GEL) v1.0
Contact	contact@oda3.org
Classification	Public

CONSOLIDATION AND STATUS NOTICE

This remains the single consolidated public UAIF implementation guide; the earlier Quick Start item is folded into it. The guide is aligned to the frozen canonical schema and the corrected public reference set. It does not depend on the withdrawn NOR-UAIF-013 document. No public PyPI reference package is assumed by this edition: implementation starts from the published schema and standard Draft 2020-12 validation tooling.

1. Purpose and Audience

This guide takes an implementer from the canonical UAIF™ v1.0 schema to a validated Incident or Vulnerability record and points to the corrected public implementation references. It is intended for teams building intake, storage, exchange, validation, or reporting workflows.

The governing implementation references are the UAIF Technical Specification (TCR-STD-002), Schema Specification (TCR-STD-003), corrected Data Dictionary (TCR-UAIF-014), Validation Rules Catalogue (TCR-UAIF-015), Reference Incident Records (TCR-UAIF-018), and Incident Reporting Template (IMP-UAIF-043).

2. Quick Start

2.1 Obtain and Pin the Canonical Schema

Use the frozen uaif-core-v1.0.json artifact and verify its SHA-256 before integrating it: 55056e7cfa4fcf4b76cc4656833cb5b360a1b40fa90f23488d488409ccebeeca. Pin that exact artifact in your build or configuration rather than resolving an unspecified 'latest' version at runtime.

```
sha256sum uaif-core-v1.0.json
# expected:
# 55056e7cfa4fcf4b76cc4656833cb5b360a1b40fa90f23488d488409ccebeeca
```

2.2 Validate a Record with Draft 2020-12 + Format Checking

```
import json
from jsonschema import Draft202012Validator, FormatChecker

with open("uaif-core-v1.0.json", "r", encoding="utf-8") as f:
    schema = json.load(f)

with open("my-record.json", "r", encoding="utf-8") as f:
    record = json.load(f)

validator = Draft202012Validator(schema, format_checker=FormatChecker())
errors = sorted(validator.iter_errors(record), key=lambda e: list(e.path))

if errors:
    for e in errors:
        print("INVALID:", e.message)
else:
    print("Schema-valid UAIF v1.0 record")
```

2.3 Apply the Separate Semantic Precision Rule

Generic JSON Schema validation does not enforce the UAIF one-decimal score rule. `severity_analytical_score` and `severity_presentation_score` must be represented to one decimal place using `Decimal ROUND_HALF_UP`. Implement an explicit semantic check or use equivalent conformance tooling after schema validation.

2.4 Use the Reviewed Repository as a Regression Pattern

The reviewed schema repository contains 6 valid fixtures and 14 invalid fixtures, plus automated tests. Use the same pattern in CI: valid cases must pass; each invalid case should fail for its intended rule. Do not confuse these repository fixtures with the separate 15-case normative release-gate conformance corpus.

3. Canonical Record Model

Area	Canonical rule	Implementation reference
Record types	Incident or Vulnerability. Observation is not normative UAIF v1.0.	TCR-UAIF-014/015
Profiles	Core, Enterprise, Regulatory; profile is always required.	TCR-UAIF-014/015
Extensions	SOC_SIEM and AI_Security are additive declarations, not profiles.	TCR-UAIF-014/015
Severity explanation	severity_rationale is canonical; severity_justification is invalid.	TCR-UAIF-014/015
Top-level object	Closed via unevaluatedProperties: false; use declared fields or x_extensions.	TCR-STD-003
Scoring semantics	Published normative calculation and one-decimal precision rule.	TCR-STD-002

4. Build Your First Record

- Choose record_type: Incident or Vulnerability.
- Choose profile: Core, Enterprise, or Regulatory.
- Populate the 12 unconditional required fields: incident_uuid, incident_title, incident_summary, reporting_timestamp, deduplication_hash, remediation_status, record_type, profile, root_cause_primary_domain, root_cause_specific_type, sector_code, jurisdiction_country_code.
- If record_type = Incident, add severity_analytical_score, severity_presentation_score, severity_level, and severity_rationale.
- If the Incident profile is Enterprise or Regulatory, add realized_harm_categories. Do not add L3/L4 fields to Core.
- If root_cause_specific_type = Adversarial_Attack, add exploit_path_description.
- If regulatory_trigger is present, add trigger_confidence and human_review_recommended.
- If calibration_status = provisional, add weight_provisional_notice.
- If using atlas_technique_id, declare SOC_SIEM in extensions. If using any gated AI Security L5 field, declare AI_Security.
- Validate with Draft 2020-12 + FormatChecker, then run the separate one-decimal score-precision check.

Use IMP-UAIF-043 as the human intake form and TCR-UAIF-018 for worked schema-valid examples.

5. Profile and Extension Boundaries

Selection	Use	Key boundary
Core	Base record for the minimum supported data depth.	L3/L4 fields are prohibited.
Enterprise	Expanded operational/evidence depth.	Enterprise Incidents require realized_harm_categories.
Regulatory	Expanded profile where regulatory-routing/evidence depth is needed.	Regulatory Incidents require realized_harm_categories.
SOC_SIEM extension	Security-operations mapping such as atlas_technique_id.	Declare SOC_SIEM before gated fields appear.
AI_Security extension	Selected L5 AI/agentic/control-plane fields.	Declare AI_Security before gated fields appear.

6. Severity and Calibration

UAIF severity scoring semantics are publicly specified in TCR-STD-002. Do not replace them with local single-score formulas or infer severity_level arbitrarily from organisation-specific tiers.

- Default harm weights remain provisional pending sector validation.
- Use the published score bands: level 1 <2.0; level 2 = 2.0-<4.0; level 3 = 4.0-<6.5; level 4 = 6.5-<8.5; level 5 >=8.5.
- When calibration_status = provisional, include weight_provisional_notice.
- If using a sector-specific or custom calibration, record the applicable calibration_source where available and document methodology/assumptions.
- Do not state that a sector calibration is suitable for regulatory submission merely because an internal reliability threshold is met; legal/regulatory sufficiency is a separate determination.

7. Conformance Testing in CI

Treat schema and semantic validation as build-time controls. The reviewed repository provides a useful public pattern: validation on change, valid and intentionally invalid fixtures, checksum verification, and regression tests.

```
# Example CI sequence
python scripts/validate_examples.py
```

```
pytest -q
sha256sum --check checksums/SHA256SUMS
```

For your own implementation, add fixtures that exercise every rule you depend on, including profile prohibitions, extension gating, regulatory-trigger companions, adversarial exploit-path requirements, provisional calibration notices, and score precision.

8. Versioning and Change Control

The legacy NOR-UAIF-013 reference has been removed from this public guide because that document is private and removed from the public register. For v1.0 implementation, pin the canonical schema hash and the corrected public specifications. Before adopting a future UAIF version, review its published release notes, compatibility statement, and migration guidance, then revalidate your stored record corpus before cutover.

9. Security, Privacy, and Data Handling

UAIF records can carry sensitive content, especially in free-text fields and x_extensions. Implement data minimisation, least-privilege access, retention controls, and redaction appropriate to your organisation and applicable law.

- Never place credentials, authentication secrets, private keys, or other security secrets in UAIF narrative or extension fields.
- Avoid unnecessary personal data and unredacted third-party confidential information; use structured flags or controlled references where possible.
- Treat free-text fields such as incident_summary, exploit_path_description, and severity_rationale notes as higher re-identification/confidentiality risk than enums or booleans.
- Regulatory-routing fields are technical routing assistance only. Whether a legal notification obligation applies requires a separate legal determination.

10. Common Pitfalls

Pitfall	Correction
Using uaif-core-v1.0.schema.json	Use the canonical filename uaif-core-v1.0.json and verify the frozen SHA-256.
Using Observation	Observation is not a normative v1.0 record_type; classify as Incident or Vulnerability before submission.
Using severity_justification	Remove it. severity_rationale is the sole canonical severity explanation.
Omitting profile	profile is one of the 12 unconditional required fields.
Adding realized_harm_categories to Core	Core prohibits that L3 field; use Enterprise/Regulatory when the record legitimately requires that depth.
Treating realized_harm_categories as required for every Incident	It is additionally required only for Enterprise/Regulatory Incidents.
Using atlas_technique_id without SOC_SIEM	Declare SOC_SIEM in extensions or omit the field.
Using AI Security L5 fields without AI_Security	Declare AI_Security in extensions or omit gated fields.
Assuming JSON Schema enforces one-decimal precision	Run a separate semantic precision check.
Assuming schema validation proves legal sufficiency	Schema conformance is structural/semantic framework conformance, not a legal filing or regulator determination.

11. Notably Absent

- No dependency on NOR-UAIF-013; it is private and removed from the public register.
- No claim that oda3-uaif is currently available on PyPI or that an installation package is required for v1.0 implementation. Public implementation can begin from the canonical schema and standard validators.
- No five-profile model. UAIF v1.0 has three base profiles plus two additive extensions.
- No controlled/private severity algorithm claim; the normative v1.0 calculation semantics are publicly documented in TCR-STD-002.
- No certification-tier mapping or current certification programme is required to implement UAIF. Certification is not currently offered; a future programme would require separate published governance.
- No product-specific SOC, GRC, or procurement integration guide is asserted as part of this document.
- No claim that schema validation establishes regulatory compliance, legal sufficiency, independent assurance, or production maturity.

12. Revision Note - 04 October 2026

This corrected reissue preserves legacy identifier IMP-UAIF-037 for citation continuity and aligns the guide to the frozen canonical uaif-core-v1.0.json schema (SHA-256 55056e7cfa4fc4b76cc4656833cb5b360a1b40fa90f23488d488409ccebceca) and the corrected TCR-UAIF-014/015/018 and IMP-UAIF-043 references. It removes all NOR-UAIF-013 citations; removes unsupported public PyPI/package assumptions; corrects the schema filename, record_type vocabulary, profile/extension model, required-field set, Incident/profile conditional rules, score precision, severity-level guidance, public scoring-semantics status, certification wording, and Public classification/contact details. No new UAIF normative behavior is introduced.

GAISSF Ecosystem developed by ODA3 Institute, licensed under GEL v1.0.