ONEDGE ENERGY

GOLDEN RECORD Canonical Schema

Byte-Level Specification GRSCHEMA-1.0.0

Normative; under STANDARD axis S-0.0.4

Issuer:	IKKUNA SpA (STANDARDCO)
Document Type:	Normative byte-level schema
Version:	GRSCHEMA-1.0.0
Parent:	S-0.0.4 (Specification-Hardening Note v0.0.4)
Status:	Initial canonical schema
Intended Audience:	SDK implementers, reference validator authors, auditors
Date:	May 18, 2026

Abstract

This document is the missing byte-level canonical schema for the GOLDEN RECORD event-level commitment, the GOLDEN RECORD aggregation root, and the verifier-side public-input vector for proof system π_k. It establishes a single canonical byte representation for all GOLDEN RECORD inputs so that implementations on EVM, Substrate / Wasm, and Cairo / Starknet runtimes produce bit-identical commitments and identical proof-verification outcomes for the same logical event. The schema follows the ECME specification's `u128`-little-endian / Keccak-256 canonicalization discipline and integrates the four-axis UVS-2 binding required by Spec-Hardening Note v0.0.4.

Contents

1. Scope and Status

GRSCHEMA-1.0.0 is normative for any artifact claiming to be a GOLDEN RECORD under S-0.0.4. Capitalized keywords *MUST*, *MUST NOT*, *SHOULD*, *SHOULD NOT*, *MAY* follow IETF RFC 2119 / RFC 8174.

This schema does *not* specify the internal layout of the proof artifact π_k. That layout is governed by the prevailing P-version registration in `onedge-manifest.json`. This schema specifies only how π_k enters the canonical commitment.

2. Encoding Primitives

2.1. Endianness and Integer Encoding

All integers MUST be encoded **little-endian** with explicit width. The canonical widths are:

Type	Width	Use
`u8`	1 byte	flags, tier indicators
`u16`	2 bytes	spec sub-versions, enum classes
`u32`	4 bytes	spec_version, vector lengths, error codes, proof_system_id
`u64`	8 bytes	block heights, timestamps (block-relative), nonces
`u128`	16 bytes	asset IDs, monetary scalars, fixed-point scalars (scale Q)
`u256`	32 bytes	balances, large fixed-point intermediates
`bytes32`	32 bytes	hashes, commitments, code hashes, governance bindings

2.2. Fixed-Point Scaling

Fixed-point scalars use the protocol constant $Q = 10^{18}$ (consistent with the ECME specification). All divisions on nonnegative integers MUST use Euclidean floor.

2.3. Hash Function

The canonical commitment hash H_{P} MUST be the hash specified by the prevailing P-version. For P-1.0.0 (registered initial value), H_{P} = Keccak-256. Implementations on Cairo / Starknet MUST hash the canonical byte sequence and MUST NOT hash native Felt representations directly, consistent with the ECME specification.

2.4. Domain Separation

Every commitment MUST be hashed under a transcript label per CRYPTO_DOMAINS.md P-1.0.0:

```
ONEDGE-SDK/v1/GR/v1.0.0/Event % event-level commitment
ONEDGE-SDK/v1/GR/v1.0.0/Aggregate % period-level Merkle root
ONEDGE-SDK/v1/GR/v1.0.0/PublicInputs % verifier-side input commit
```

The transcript label is prepended to the canonical byte sequence prior to hashing:

$$\mathsf{commit}(x) := H_{\mathsf{P}}\big(\mathsf{utf8}(\mathit{label}) \,\|\, \mathsf{u32\_LE}(\mathrm{len}(\mathit{label})) \,\|\, x\big).$$

2.5. Variable-Length Vectors

Variable-length vectors are serialized as `u32`-LE length prefix followed by the concatenated element bytes. Implementations MUST enforce the bounds declared in Section 7.

3. Domain Types

3.1. Asset State x_k

Field	Type	Semantics
`soc_q`	`u128`	State of charge, fixed-point scale Q (range $[0, Q]$)
`t_cell_q16`	`u128`	Cell temperature in °C, fixed-point scale 2^{16}
`t_ambient_q16`	`u128`	Ambient temperature in °C, fixed-point scale 2^{16}
`energy_avail_mwh_q`	`u256`	Physical energy available, MWh fixed-point scale Q
`energy_contractual_mwh_q`	`u256`	Contractual energy (per bridging appendix)
`extras_root`	`bytes32`	Merkle root over asset-class-specific extra state fields

Fields are serialized in the order shown. Total fixed size: $16 + 16 + 16 + 32 + 32 + 32 = 144$ bytes.

3.2. Dispatch Action u_k

Field	Type	Semantics
`p_charge_kw_q`	`u128`	Charge power command, kW fixed-point scale Q
`p_discharge_kw_q`	`u128`	Discharge power command, kW fixed-point scale Q
`duration_blocks`	`u64`	Action duration in block-height units
`control_class`	`u16`	Enum: `0=arbitrage`, `1=reserve`, `2=frequency`, `3=ramp`, `4=manual`, `>=128=reserved`
`control_flags`	`u16`	Bitfield (bit 0: dispatch-from-CEN, bit 1: shadow-mode, bit 2: emergency, others reserved = 0)

Total fixed size: $16 + 16 + 8 + 2 + 2 = 44$ bytes.

3.3. Dispatch Context d_k

Field	Type	Semantics
`market_id`	`u32`	Registered market identifier (CEN-SEN = `0x00000001`, ERCOT = `0x00000002`, ...)
`instruction_root`	`bytes32`	Hash of the originating system instruction or P2P request
`counterparty_root`	`bytes32`	Hash of the counterparty / numeraire context (zero for unilateral dispatch)
`freshness_h`	`u64`	Block height at which d_k was observed

Total fixed size: $4 + 32 + 32 + 8 = 76$ bytes.

3.4. Telemetry y_k

The telemetry field is a variable-length, sorted, deduplicated vector of attested observations. Each entry has the structure:

Field	Type	Semantics
`channel_id`	`u32`	Registered telemetry channel (e.g. `0x0100` = t_cell, `0x0200` = soc, ...)
`sample_h`	`u64`	Block height anchor for the sample
`value`	`u256`	Channel-typed value (fixed-point scale per channel registry)
`tier`	`u8`	Provenance Tier $\tau(y_{k,j}) \in \{0,1,2,3,4\}$
`attestation_root`	`bytes32`	Hash of the attestation chain justifying tier

Each entry is $4+8+32+1+32 = 77$ bytes. The vector header is a `u32`-LE count. Entries MUST be sorted ascending by (channel_id, sample_h).

3.5. Envelope Parameter Set θ Reference

The GOLDEN RECORD does not inline θ. It references the active FINGERPRINT BASELINE for the asset:

Field	Type	Semantics
`fb_hash`	`bytes32`	h_{θ_a}, the canonical FINGERPRINT BASELINE hash (see Spec-Hardening v0.0.4 Def. 2)
`fb_issuance_h`	`u64`	t_a^0, FINGERPRINT BASELINE issuance block height
`fb_revocation_h`	`u64`	0 if active; non-zero block height if revoked

Total fixed size: $32+8+8 = 48$ bytes.

3.6. Provenance Tier and Predicate

Field	Type	Semantics
`t_required`	`u8`	Minimum required tier $T \in \{0,1,2,3,4\}$ for this statement
`tau_attained`	`u8`	Effective tier $\tau(y_k) = \min_j \tau(y_{k,j})$ over the required component set
`predicate_bit`	`u8`	$P_T(y_k) \in \{0,1\}$ (redundant; derivable from tau_attained $\geq$ t_required)

Total fixed size: 3 bytes.

3.7. Envelope Evaluation and Classification

Field	Type	Semantics
`envelope_eval`	`u8`	$E(x_k, u_k; \theta) \in \{0, 1\}$
`sigma`	`u8`	Ternary classification: `0=Opacity`, `1=Compliant`, `2=Breach`
`breach_mask`	`u32`	Bitfield identifying which envelope constraints were violated (used iff sigma $= 2$)

Total fixed size: $1 + 1 + 4 = 6$ bytes.

3.8. Proof Artifact π_k

Field	Type	Semantics
`proof_system_id`	`u32`	Resolves to a P-version in the manifest
`vk_id`	`bytes32`	Verification key identifier for the statement
`statement_id`	`bytes32`	Statement-class identifier under the P-version
`proof_bytes`	`bytes(*)`	Variable-length, u32-LE length-prefixed proof body

For $\sigma =$ Opacity the proof body MAY be a witness-of-insufficient-tier in the form prescribed by the registered P-version; for $\sigma \in \{$Compliant, Breach$\}$ the proof body MUST satisfy the verifier-side public-input vector defined in Section 6.

3.9. UVS-2 Binding

Field	Type	Semantics
`s_version`	`u32`	S-version, packed as `(MAJOR«24)|(MINOR«16)|(PATCH«8)|PRE`
`i_version`	`u32`	I-version, same packing
`p_version`	`u32`	P-version, same packing
`g_binding`	`bytes32`	Hash resolving to a G-version in the manifest
`manifest_root`	`bytes32`	Merkle root of `onedge-manifest.json` valid at issuance

Total fixed size: $4 + 4 + 4 + 32 + 32 = 76$ bytes.

4. Event-Level Commitment GR_k

4.1. Logical Pre-Image

The event-level pre-image (before hashing) is the ordered concatenation:

1. k: u64-LE event index (8 bytes)
2. asset_id: u128-LE (16 bytes)
3. x_k: 144 bytes (§3.1)
4. u_k: 44 bytes (§3.2)

5. d_k: 76 bytes (§3.3)
6. y_k: variable (§3.4)
7. theta_ref: 48 bytes (§3.5)
8. provenance: 3 bytes (§3.6)
9. evaluation: 6 bytes (§3.7)
10. x_kplus1: 144 bytes (asset state after transition)
11. pi_k: variable (§3.8)
12. uvs2: 76 bytes (§3.9)
13. gr_kminus1: 32 bytes (parent hash; `0x00...00` for genesis)

4.2. Commitment

$$GR_k := H_{\mathsf{P}}\Big(\mathsf{utf8}(\text{“ONEDGE-SDK/v1/GR/v1.0.0/Event”}) \,\|\, \mathsf{u32\_LE}(\mathrm{len}_{\mathsf{label}}) \,\|\, \mathsf{preimage}_k\Big).$$

The output is a `bytes32` commitment.

5. Period-Level Aggregation $GR_{\mathcal{K}}$

For an ordered, contiguous set of event indices $\mathcal{K} = \{k_1, \ldots, k_n\}$ on a single asset, the aggregation is a Merkle tree over the sorted commitments:

- **Leaf encoding:** $\mathsf{leaf}_j := H_{\mathsf{P}}(\mathsf{utf8}(\text{“ONEDGE-SDK/v1/GR/v1.0.0/Aggregate”}) \,\|\, \mathsf{u32\_LE}(\mathrm{len}_{\mathsf{label}}) \,\|\, \mathsf{u64\_LE}($
- **Internal node:** $\mathsf{node}(L, R) := H_{\mathsf{P}}(L \,\|\, R)$.
- **Empty subtree:** represented by 32 zero bytes; padding is permitted to round up to a power of two but the padding leaves MUST be the all-zero hash.
- **Root:** the resulting Merkle root is $GR_{\mathcal{K}}$.

The aggregation MUST be reproducible from the event commitments alone, without re-hashing pre-images.

The relationship to the SDK `SECURITY_MODEL.md` state root commitment is that the NOMT state root MUST embed the latest $GR_{\mathcal{K}}$ as the canonical anchor for *audit and replay* purposes. The two are not identical artifacts: $GR_{\mathcal{K}}$ is a per-asset, per-period GOLDEN RECORD Merkle root; the NOMT root is the system-level state root that includes (among other things) the registry of $GR_{\mathcal{K}}$ commitments.

6. Verifier-Side Public-Input Vector

The proof artifact π_k MUST verify against the following ordered public-input vector:

1. $\mathsf{utf8}(\text{“ONEDGE-SDK/v1/GR/v1.0.0/PublicInputs”})$
2. $\mathsf{u32\_LE}(\mathrm{len}_{\mathsf{label}})$
3. $\mathsf{u64\_LE}(k)$

4. u128_LE(asset_id)
5. $H_{\mathsf{P}}(\mathsf{encode}(x_k))$
6. $H_{\mathsf{P}}(\mathsf{encode}(u_k))$
7. $H_{\mathsf{P}}(\mathsf{encode}(d_k))$
8. $H_{\mathsf{P}}(\mathsf{encode}(y_k))$
9. h_{θ_a} (the Fingerprint Baseline hash)
10. u8(t_required)
11. u8(tau_attained)
12. u8(predicate_bit)
13. u8(envelope_eval)
14. u8(sigma)
15. $H_{\mathsf{P}}(\mathsf{encode}(x_{k+1}))$
16. p_version (u32-LE)
17. manifest_root (bytes32)
18. GR_{k-1} (bytes32; parent commitment)

These items are concatenated in order. The hash of this concatenation, under the label ONEDGE-SDK/v1/GR/v1.0.0/I
MUST equal the public_inputs_hash the proof was generated against.

7. Bounds

To prevent codec DoS, the following bounds MUST be enforced at deserialization. They MAY be tightened by the manifest but MUST NOT be loosened.

Bound	Default	Semantics
GR_MAX_PREIMAGE_BYTES	65536	Hard cap on $\mathsf{preimage}_k$ size
GR_MAX_TELEMETRY_ENTRIES	1024	Max entries in y_k
GR_MAX_PROOF_BYTES	32768	Hard cap on proof_bytes
GR_MAX_AGGREGATION_DEPTH	20	Max Merkle depth ($\Rightarrow$ up to 2^{20} leaves)

Violations MUST be reported as ERR_CODEC_INVALID.

8. Error Codes

The following reason codes are introduced under GRSCHEMA-1.0.0 in addition to those already published in RUNTIME_ABI.md (I-1.0):

Code	Semantics
`ERR_GR_PREIMAGE_TOO_LARGE`	$\mathsf{preimage}_k > \mathsf{GR\_MAX\_PREIMAGE\_BYTES}$
`ERR_GR_TELEMETRY_ORDER`	y_k entries not sorted ascending by $(\mathsf{channel\_id}, \mathsf{sample\_h})$
`ERR_GR_TELEMETRY_DUPLICATE`	Duplicate $(\mathsf{channel\_id}, \mathsf{sample\_h})$ key in y_k
`ERR_GR_TIER_INCONSISTENT`	$\mathsf{predicate\_bit} \neq \mathbb{1}[\mathsf{tau\_attained} \geq \mathsf{t\_required}]$
`ERR_GR_SIGMA_INCONSISTENT`	sigma does not match $(\mathsf{predicate\_bit}, \mathsf{envelope\_eval})$ per Spec-Hardening v0.0.3
`ERR_GR_PROOF_PUBLIC_INPUTS`	Proof verification hash $\neq$ §6
`ERR_GR_FB_MISMATCH`	fb_hash not registered or revoked at k
`ERR_GR_GOVERNANCE_UNREGISTERED`	g_binding not present in manifest entry for active s_version
`ERR_GR_MANIFEST_ROOT_STALE`	manifest_root not within the accepted window
`ERR_GR_PARENT_HASH_MISMATCH`	GR_{k-1} not consistent with the local hash chain for asset_id

9. Consistency Rules

A canonical GOLDEN RECORD MUST satisfy all of the following at deserialization or commit time. Any failure MUST produce the named reason code without producing a commitment.

Rule 1 (Tier-predicate consistency)**.** $\mathsf{predicate\_bit} = \mathbb{1}[\mathsf{tau\_attained} \geq \mathsf{t\_required}]$ (else `ERR_GR_TIER_INCONSISTE`

Rule 2 (Classification consistency)**.** sigma is forced by (predicate_bit, envelope_eval) per the table below (else `ERR_GR_SIGMA_INCONSISTENT`):

predicate_bit	envelope_eval	sigma
0	—	0 (Opacity)
1	1	1 (Compliant)
1	0	2 (Breach)

Rule 3 (Breach-mask consistency)**.** If $\mathsf{sigma} \neq 2$, then $\mathsf{breach\_mask} = 0$. If $\mathsf{sigma} = 2$, then breach_mask MUST be non-zero.

Rule 4 (Parent-hash consistency)**.** gr_kminus1 MUST equal the canonical hash of the prior event for asset_id recorded in the verifier state; for $k = 0$, the parent hash is 32 zero bytes (else `ERR_GR_PARENT_HASH_MISMATCH`).

Rule 5 (FINGERPRINT BASELINE activeness)**.** fb_hash MUST be registered in the STANDARDCO FINGERPRINT BASELINE registry, with $\mathsf{fb\_issuance\_h} \leq \mathsf{sample\_h}_{\min}(y_k)$ and either $\mathsf{fb\_revocation\_h} = 0$ or $\mathsf{fb\_revocation\_h} > \mathsf{sample\_h}_{\max}(y_k)$.

Rule 6 (Governance binding registered)**.** g_binding MUST resolve, via manifest_root, to a G-version listed as compatible with the prevailing s_version.

10. Cross-Runtime Conformance Requirements

Conformant implementations MUST produce bit-identical commitments for the same logical pre-image. This implies:

- All integer fields are serialized little-endian; no implementation MAY use big-endian, even on internally big-endian platforms.
- Hashing MUST be performed on the canonical byte sequence; implementations on Cairo / Starknet MUST NOT hash native Felt representations.
- Variable-length vectors MUST be length-prefixed (`u32`-LE).
- Telemetry vectors MUST be sorted ascending by (channel_id, sample_h) before hashing; duplicate keys are forbidden.
- The transcript label MUST be prepended exactly as the UTF-8 byte sequence in this document.

A **Cross-Runtime Conformance Vector** suite SHALL be published under `conformance-vectors/grschema-1.0` in the manifest distribution. Each vector contains a logical pre-image, the expected canonical bytes (hex), and the expected GR_k (hex). Implementations MUST reproduce all vectors exactly.

11. Aggregation Replay and Audit

For an asset a, the audit-replay procedure is:

1. Recover the ordered $\{(k_j, GR_{k_j})\}$ for the period $\mathcal{K}$.
2. Recompute the Merkle aggregation per §5.
3. Verify that the computed $GR_{\mathcal{K}}$ matches the recorded value.
4. Optionally, for each k_j, recompute GR_{k_j} from the archived pre-image and verify against the recorded commitment.

The NOMT state-root inclusion proof MAY be used to verify that $GR_{\mathcal{K}}$ was recorded by the SDK at issuance.

12. Forward Compatibility

GRSCHEMA-1.0.0 reserves the following extension points:

- **Reserved enum values:** `control_class` ≥ 128, `channel_id` $\geq$ `0xFFFF0000`, $\sigma \geq 3$ are reserved for future schema MAJOR revisions and MUST NOT be used by GRSCHEMA-1.0.x.
- **Reserved bits:** all bits not explicitly defined in any flags field MUST be set to zero; non-zero reserved bits MUST cause `ERR_CODEC_INVALID`.
- **Extras root:** the `extras_root` field of x_k is the only sanctioned site for asset-class-specific state extensions in GRSCHEMA-1.0.x.

Breaking changes (new fields in fixed-size structures, changed widths, changed sort order, new mandatory invariants) MUST bump the schema MAJOR.

Disclaimer

This document is a normative byte-level schema prepared by IKKUNA SpA. It does not constitute legal, financial, investment, regulatory, engineering, tax, or professional advice. ONEDGE ENERGY remains a proposed open standard and candidate specification in pre-TRL-7 validation status.