Independent Submission
Internet-Draft
Intended status: Informational
Expires: 29 November 2026

C. Hopley
AlgoVoi

28 May 2026

JCS Canonicalisation Discipline for Agentic-Payment Receipts
draft-hopley-x402-canonicalisation-jcs-v1-01

Abstract

This document specifies a canonicalisation discipline for agentic-payment receipt formats. The discipline pins JSON Canonicalization Scheme (JCS, RFC 8785) as the canonical preimage form, plus a small set of schema-normalisation rules that must be applied before canonicalisation to preserve byte-determinism across independent implementations and across statutory retention periods.

The discipline is identified by the URN urn:x402:canonicalisation:jcs-rfc8785-v1. Receipt formats that reference this discipline carry an in-band canon_version field recording the version under which they were emitted, enabling year-N re-verification of retained bytes without dependence on an out-of-band rule registry.

The discipline is byte-for-byte cross-validated across eight independent JCS implementations in eight programming languages: Python (rfc8785), TypeScript (canonicalize), Go (gowebpki/jcs), Rust (serde_jcs), Java (cyberphone/json-canonicalization, by the RFC 8785 editor), PHP (root23/php-json-canonicalization), C#/.NET (Baqhub.Packages.JsonCanonicalization), and Ruby (json-canonicalization). The attestation record covering 192 byte-for-byte agreements is published at the AlgoVoi conformance vectors repository.

This document is normatively referenced by [draft-hopley-x402-compliance-receipt], [draft-hopley-x402-refund-receipt], and successor AlgoVoi-authored receipt-format Internet-Drafts. It is complementary to [draft-vauban-x402-stark-receipts], which uses a compatible canonicalisation discipline for its cryptographic settlement proofs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 29 November 2026.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Motivation 3
  1.2. Scope 4
  1.3. Relationship to other Internet-Drafts 4
  1.4. Authorship and provenance 5
2. Conventions and Definitions 5
  2.1. Notation 5
  2.2. Definitions 5
3. The Canonicalisation Discipline 6
4. Schema Normalisation Requirements 6
  4.1. Substrate Rule 2 -- integer-millisecond timestamps . . . 6
  4.2. Field names are load-bearing opaque bytes 7
  4.3. Array element order is preserved 7
  4.4. Numeric values follow RFC 8785 Section 3.2.2.3 7
  4.5. Type validation precedes canonicalisation 8
  4.6. 4.6. In-band rule pin (canon_version) 8
5. Retention Property 8
6. Versioning 9
7. Cross-Implementation Reproducibility 9
8. 8. Scope Conventions for action_ref (Non-Normative) 11

9. 9. Transactional action_ref Lifecycle (Non-Normative) . . . 12
10. IANA Considerations 12
10.1. URN Namespace Registration 13
10.2. canon_version Value Registration 13
11. Security Considerations 13
11.1. Cross-implementation hash equivalence 13
11.2. Timestamp ambiguity 14
11.3. Field-name aliasing 14
11.4. Version-rollback attack 14
11.5. Numeric precision 14
Appendix A. References 14
A.1. Normative References 14
A.2. Informative References 15
Appendix B. Appendix A. Example Canonical Forms (Informative) 15
B.1. A.1. Integer-timestamp normalisation (Substrate Rule 2) 15
B.2. A.2. Array order is load-bearing 16
B.3. A.3. canon_version pin 16
Appendix C. Appendix B. Reference Implementations (Informative) 17
Appendix D. Appendix C. Known Adopters (Informative) 17
Appendix E. Appendix D. Acknowledgments 18
Author’s Address 18

1. Introduction

1.1. Motivation

Agentic-payment receipt formats (including the AlgoVoi compliance receipt, refund receipt, and successor formats in this series) take a content hash over a canonicalised JSON object:

```
content_hash = SHA-256( canonicalize( object ) )
```

If two implementations of a receipt format canonicalise the same logical object differently, they produce different content hashes. Downstream verifiers see hash mismatch and reject the receipt; audit chains break; year-N supervisor re-verification of retained bytes fails.

This problem is well-understood in the JSON canonicalisation literature. RFC 8785 (JCS) provides the standard solution at the canonicalisation layer: a single deterministic JSON encoding rule that every conforming implementation produces identically.

What RFC 8785 does NOT specify is the *schema-normalisation discipline that must be applied BEFORE canonicalisation*. Receipt formats that emit timestamp as RFC 3339 strings, that allow duplicate keys in source JSON, or that fail to pin field-name normalisation produce divergent canonical bytes despite both implementations being RFC 8785 conformant. The receipt format itself must impose pre-canonicalisation discipline.

This document specifies the discipline.

1.2. Scope

This document specifies:

* The canonicalisation rule used for content hashes (Section 3).
* The schema-normalisation requirements applied BEFORE canonicalisation (Section 4).
* The retention property (Section 5) that motivates the discipline for statutory record-keeping under MiCA, AMLR, and DORA.
* The version pin (canon_version field) and its compatibility semantics (Section 6).
* Cross-implementation reproducibility evidence (Section 7).
* Non-normative recommendations for the action_ref primitive (Sections 8 and 9).

This document does NOT specify:

* A specific receipt format. Receipt formats are specified by dedicated I-Ds (e.g. draft-hopley-x402-compliance-receipt, draft-hopley-x402-refund-receipt) that normatively reference this canonicalisation discipline.
* A wire protocol. The discipline applies at the canonicalisation-of-an-object layer; the wire protocol that transports the canonical bytes is out of scope.
* Cryptographic settlement proofs. Cryptographically-strong settlement proofs are an orthogonal mechanism specified elsewhere (out of scope for this canonicalisation discipline).

1.3. Relationship to other Internet-Drafts

This document is normatively referenced by:

* target="admission-time compliance screening receipts" (admission-time compliance screening receipts)
* target="post-settlement refund receipts" (post-settlement refund receipts)

* Future AlgoVoi-authored receipt-format I-Ds (settlement attestation, cancellation receipt, mandate revocation receipt, etc.)

1.4. Authorship and provenance

This document, the canonicalisation discipline it specifies, and the conformance vectors derived from it are AlgoVoi work under sole AlgoVoi authorship. Substrate authorship history is catalogued at target="https://docs.algovoi.co.uk/substrate-authorship-provenance" (https://docs.algovoi.co.uk/substrate-authorship-provenance).

2. Conventions and Definitions

2.1. Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Definitions

canonicalisation discipline: the combination of (a) the canonicalisation rule (Section 3) and (b) the schema-normalisation requirements (Section 4). A receipt format that references this discipline must conform to both.

canon_version: an in-band string identifying which version of this canonicalisation discipline applies. Fixed value jcs-rfc8785-v1 for the discipline specified in this document. Future versions of this discipline MUST increment canon_version.

content hash: SHA-256, lowercase hex, of the canonical bytes produced by applying this discipline to a JSON object.

JCS: JSON Canonicalization Scheme as specified in [RFC8785].

load-bearing under canonicalisation: a field whose value, type, or ordering produces byte-distinct canonical bytes from any variation. Implementations MUST preserve load-bearing properties exactly during emission and verification.

Substrate Rule 2: the schema-normalisation requirement that timestamp fields MUST be JSON integers (milliseconds since Unix epoch) and MUST NOT be RFC 3339 strings. So-named because it was the second normative rule formalised in the canonicalisation discipline; the first was the JCS pin itself.

3. The Canonicalisation Discipline

The canonicalisation discipline specified by this document is identified by the URN:

urn:x402:canonicalisation:jcs-rfc8785-v1

A content hash produced under this discipline is computed as:

content_hash = SHA-256(JCS(object)), lowercase hex

Where:

* object is a JSON object that has been through the schema-normalisation requirements of Section 4.
* JCS is the JSON Canonicalization Scheme of [RFC8785].
* SHA-256 is the cryptographic hash function of [RFC6234].

The lowercase-hex encoding of the SHA-256 output is canonical. Uppercase-hex or base64-encoded forms are NOT acceptable for content_hash values that participate in cross-implementation hash equality checks.

4. Schema Normalisation Requirements

Schema normalisation MUST be applied BEFORE canonicalisation. The following requirements are normative.

4.1. Substrate Rule 2 -- integer-millisecond timestamps

Timestamp fields whose semantic is "an instant in time" MUST be encoded as JSON integers representing milliseconds since the Unix epoch (1970-01-01T00:00:00Z).

RFC 3339 string forms (e.g. "2026-05-28T12:00:00Z") are NOT acceptable as canonical preimage form.

Rationale: RFC 3339 admits multiple lexically distinct encodings of the same semantic instant (e.g. "2026-05-28T12:00:00Z" vs "2026-05-28T12:00:00+00:00" vs "2026-05-28T13:00:00+01:00"). These produce different JCS canonical bytes and therefore different SHA-256 hashes despite representing the same instant. The integer form is

unambiguous: a given instant has exactly one millisecond representation, and JCS encodes integers deterministically per [RFC8785] Section 3.2.2.3.

Implementations MUST reject non-integer timestamp inputs (RFC 3339 strings, floating-point milliseconds, ISO duration strings, etc.) at the validation step BEFORE canonicalisation. Coercion of non-conforming inputs is non-conforming behaviour (Section 4.5).

4.2. Field names are load-bearing opaque bytes

Field names are load-bearing under RFC 8785. Renaming a field while preserving its value produces a different canonical content hash.

Schemas using this discipline MUST pin field names exactly. Aliases at the wire layer (e.g. accepting tx_id as an alias for transaction_id) MUST be normalised to the canonical name before canonicalisation.

This rule is consistent with [RFC8785] Section 3.2.3, which sorts object keys lexicographically by code point but does not normalise or alias keys.

4.3. Array element order is preserved

Array element order is preserved under RFC 8785 ([RFC8785] Section 3.2.3). Arrays are NOT sorted during canonicalisation.

["EU", "UK"] and ["UK", "EU"] produce different canonical bytes and different content hashes.

Schemas requiring a canonical array order MUST specify it at the schema level (e.g. "the jurisdiction_flags array MUST be ordered primary-jurisdiction-first"), NOT rely on JCS to impose ordering.

4.4. Numeric values follow RFC 8785 Section 3.2.2.3

Numeric values in JSON objects are canonicalised per [RFC8785] Section 3.2.2.3. Implementations MUST emit numbers in the canonical form specified there.

For amount-typed fields (e.g. payment amounts, refund amounts), schemas SHOULD encode amounts as decimal-digit strings in the asset’s minor unit rather than as JSON numbers. Rationale: large amounts may exceed JavaScript’s Number.MAX_SAFE_INTEGER (2^53 - 1), and floating-point JSON numbers admit representation-precision loss across implementations. String encoding avoids both classes of failure.

4.5. Type validation precedes canonicalisation

Type validation occurs BEFORE canonicalisation. Verifiers MUST reject non-conforming inputs at the parse or schema-validation step, NOT by coercing to canonical form before computing content hashes.

Specifically, verifiers MUST reject (not coerce) inputs with:

* Wrong scalar type for a field (e.g. RFC 3339 string for a timestamp; floating-point number for a minor-unit amount).
* Missing required fields.
* Duplicate keys in source JSON.
* Non-normalised Unicode where the schema pins a normalisation form (e.g. NFC for DID URIs).

Producer-side rule violations should fail loudly at conformance test. Verifier-side coercion fails silently as cross-observer disagreement months later, AND breaks re-verifiability at audit time: the coercion step is verifier-local and will not be replayed identically by a supervisor running the canonical rule against the raw retained object.

4.6. 4.6. In-band rule pin (canon_version)

Objects produced under this discipline SHOULD include a canon_version field carrying the discipline version under which they were emitted. The current value is jcs-rfc8785-v1.

Producers emitting objects under a statutory retention obligation (see Section 5) MUST include canon_version. The rule version must be determinable at re-verification time without reference to the emitting system’s current configuration.

canon_version is itself canonicalised and contributes to the content hash. A receipt emitted under one discipline version cannot be silently re-hashed under a successor version.

5. Retention Property

Canonicalisation determinism is a retention obligation as well as a cross-observer property. Objects produced for frameworks with retention obligations under European Union law MUST be re-verifiable against the canonicalisation rule version under which they were emitted. Applicable frameworks include:

* Markets in Crypto-Assets Regulation, Article 80 (Regulation (EU) 2023/1114): operator records of crypto-asset transactions must be retained and re-verifiable for the statutory period.

* Anti-Money Laundering Regulation, Article 56 (Regulation (EU) 2024/1624): obliged entity records must be retained for five years and made available to competent authorities.
* Digital Operational Resilience Act, Article 14 (Regulation (EU) 2022/2554): ICT-related incident and operational records must be retained and audit-traceable.

A supervisor re-verifying retained bytes at year five against a contemporaneous canonicalisation rule needs identical canonical bytes across that five-year gap. The rule version must be determinable from the retained bytes alone, without reference to the emitting system's current state.

This document's canon_version discipline (Section 4.6) makes the rule version self-describing on the retained bytes. A verifier reading a receipt years after emission can determine which version of this canonicalisation discipline to apply and reproduce the content hash byte-identical.

Equivalent record-keeping obligations apply in jurisdictions outside the European Union; the discipline is jurisdiction-neutral but the retention property is named-target for the EU regulations because their re-verifiability requirement is explicit.

6. Versioning

This document specifies version jcs-rfc8785-v1 of the canonicalisation discipline.

Future revisions that change any normative requirement in Section 4 or Section 3 MUST increment the canon_version (e.g. jcs-rfc8785-v2). Successor versions will be published as follow-on Internet-Drafts in the same document series (draft-hopley-x402-canonicalisation-jcs-v<n>).

Downstream receipt-format specifications that reference this discipline MUST cite a specific canon_version value. A receipt format MAY support multiple discipline versions concurrently (e.g. accept both v1 and v2 on input) but each individual emitted object carries exactly one canon_version value identifying which rule was applied at emission.

7. Cross-Implementation Reproducibility

The canonicalisation discipline has been byte-for-byte cross-validated across *eight independent JCS implementations in eight programming languages*, executed on 2026-05-24:

Language	Library	Author / authoring entity
Python	rfc8785 0.1.4	Trail of Bits
TypeScript	canonicalize 3.0.0	Samuel Erdtman
Go	gowebpki/jcs v1.0.1	Web PKI Working Group
Rust	serde_jcs 0.2.0	independent author
Java	io.github.erdtman:java-json-canonicalization 1.1	Anders Rundgren (RFC 8785 editor) and Samuel Erdtman
PHP	root23/php-json-canonicalization 1.0.1	root23
C# / .NET	Baqhub.Packages.JsonCanonicalization 1.0.1	Baqhub
Ruby	json-canonicalization 1.0.0	RubyGems community

Table 1

All eight implementations were independently authored by non-overlapping author sets, including the editor of RFC 8785 itself (Anders Rundgren via the Java implementation).

The matrix validates 24 conformance vectors across three anchor sets (action_ref_namespace_v0, action_ref_transactional_v0, compliance_receipt_v1), producing 192 byte-for-byte agreements (24 vectors x 8 implementations). The attestation record is at:

https://github.com/chopmob-cloud/algovoi-jcs-conformance-vectors/blob/main/_attestations/2026-05-24-8-impl-cross-validation.md

A receipt-format implementation conforming to this canonicalisation discipline, written in any of the eight listed languages, produces identical canonical bytes for any receipt under the discipline.

8. 8. Scope Conventions for action_ref (Non-Normative)

The action_ref atomic primitive is defined elsewhere in the substrate as:

```
action_ref = SHA-256( JCS( { agent_id, action_type, scope, timestamp_ms } ) )
```

At the canonicalisation layer, the scope field is typed as a non-empty string with no closed enumeration. This document records a non-normative convention for scope namespacing observed across production emitters.

Recommended portable form: <emitter>:<scope> namespace prefix.

Current production usage observed across the substrate’s emitter set as of 2026-05-24:

scope value	Emitter surface
algovoi:compliance_screen	AlgoVoi /compliance/screen
algovoi:refund	AlgoVoi refund receipt issuer
algovoi:settlement	AlgoVoi settlement attestation issuer

Table 2

Additional emitters using this discipline are expected to namespace their own scope values under a <emitter>:<scope> convention. The substrate does not enumerate third-party emitters in this specification.

The namespaced value is hashed into action_ref like any other string, so the dedup / idempotency property of the primitive is preserved. The recommendation gives reputation consumers and downstream verifiers an unambiguous mapping target where multiple emitters would otherwise collide on unprefixed short-form scopes (e.g. two emitters both using payment for genuinely different semantic concepts).

Byte-level reference digests for the four named anchors above plus four unprefixed equivalents and four pair invariants are pinned in the action_ref_namespace_v0 conformance vector set:

https://github.com/chopmob-cloud/algovoi-jcs-conformance-vectors/tree/main/vectors/action_ref_namespace_v0/

Spec-level closure of the scope value-space would lock out future emitters arriving with valid new scopes and is intentionally avoided.

9. 9. Transactional action_ref Lifecycle (Non-Normative)

For transactional flows that traverse multiple state transitions (authorisation -> settlement -> refund; issuance -> execution -> revocation; admission -> review -> close), the action_ref primitive serves as a stable identity anchor across the full lifecycle.

The four-field preimage { agent_id, action_type, scope, timestamp_ms } is fixed at the moment the action is first declared and does not change as the action progresses through state transitions.

Per-transition lifecycle metadata (for example settlement-proof timestamps, refund-window expiry, authority-verification timestamps, revocation-check timestamps) lives OUTSIDE the action_ref preimage. These are separate claims that may evolve as the action moves through its states. Keeping them outside the preimage preserves the invariant: the action_ref digest is stable across every state transition; the identity of the action does not change when the authority state, settlement state, or refund state does.

This is the load-bearing property that makes action_ref composable across the substrate’s emitter set. A downstream verifier auditing a single transition (e.g. the settlement step of a payment) does not need to replay the full chain to bind the action; the action_ref digest alone is sufficient. Per-transition timestamp claims at each step provide the temporal proof that the transition was valid at that moment, independently verifiable.

The integer timestamp_ms requirement (Substrate Rule 2, Section 4.1) applies to the preimage timestamp as well as to any per-transition timestamps emitted alongside it. RFC 3339 string forms are NOT acceptable for either the preimage or the lifecycle metadata. The integer-timestamp invariant is independently anchored in [I-D.vauban-x402-stark-receipts] Section 7.1, which explicitly rejects timestamp (RFC 3339 string) as a wire form.

10. IANA Considerations

10.1. URN Namespace Registration

This document requests registration of the URN namespace:

urn:x402:canonicalisation:jcs-rfc8785-v1

* Namespace ID: x402
* Sub-namespace: canonicalisation
* Version identifier: jcs-rfc8785-v1
* Purpose: identifies the JSON canonicalisation discipline specified by this document, including JCS RFC 8785 plus the schema-normalisation requirements in Section 4.
* Registry: IETF URN sub-namespace (if accepted by ART area) or AlgoVoi-published registry for the x402 namespace.

10.2. canon_version Value Registration

The string value jcs-rfc8785-v1 is registered as the discipline version identifier carried in canon_version fields produced under this document.

Successor versions will register jcs-rfc8785-v2, jcs-rfc8785-v3, etc. as follow-on revisions of this Internet-Draft are published.

11. Security Considerations

11.1. Cross-implementation hash equivalence

The discipline guarantees that two conforming implementations, applied to byte-identical normalised JSON objects, produce byte-identical content hashes. This property is fundamental to audit-chain integrity and supervisor re-verification.

A non-conforming implementation that coerces input rather than rejecting it (Section 4.5) breaks this guarantee silently. Such an implementation appears to function correctly at point of emission but produces hashes that diverge from supervisor re-verification. The non-conformance can go undetected until audit, at which point retained records are unverifiable.

Operators MUST validate their implementations against the conformance vector corpus (Section 7) BEFORE relying on the discipline for retention-obligated records.

11.2. Timestamp ambiguity

The integer-millisecond timestamp requirement (Section 4.1) is a security property as well as a determinism property. RFC 3339 string timestamps admit ambiguity in timezone representation (e.g. Z vs +00:00 vs -00:00), leap-second handling, and sub-second precision. An attacker emitting receipts with ambiguous timestamps could produce two byte-distinct receipts representing the same logical instant; a verifier checking content hashes would treat them as distinct events.

The integer-millisecond form eliminates this attack class: each instant has exactly one valid millisecond representation.

11.3. Field-name aliasing

The field-name discipline (Section 4.2) prevents an aliasing attack: a producer emitting tx_id while a downstream verifier expects transaction_id produces a hash mismatch that the verifier sees as tampering, even though no semantic tampering occurred. The discipline requires the wire-layer alias to be normalised to the canonical name before canonicalisation; post-normalisation, the hash check is unambiguous.

11.4. Version-rollback attack

The canon_version field is itself canonicalised. A producer cannot silently downgrade a receipt from jcs-rfc8785-v2 to jcs-rfc8785-v1 post-emission without changing the content hash. A verifier seeing a v1-claimed hash that does not validate against v1 canonicalisation rejects the receipt; a verifier seeing a v2-claimed hash treats it as v2.

11.5. Numeric precision

The amount-as-string recommendation (Section 4.4) prevents a precision-loss attack: large amounts emitted as JSON numbers may lose precision when round-tripped through a verifier whose JSON parser uses 64-bit floats, producing a different hash than the producer expected. String encoding in minor units preserves precision exactly.

Appendix A. References

A.1. Normative References

* [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

* [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011.
* [RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017.
* [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.
* [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017.
* [RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020.

A.2. Informative References

* [I-D.hopley-x402-compliance-receipt] Hopley, C., "Categorical Compliance Screening Receipt Format for Agentic-Payment Flows", draft-hopley-x402-compliance-receipt-00, May 2026.
* [I-D.hopley-x402-refund-receipt] Hopley, C., "Categorical Refund Receipt Format for Agentic-Payment Flows", draft-hopley-x402-refund-receipt-00, May 2026.
* [AlgoVoi-JCS-8-impl] AlgoVoi, "Eight-implementation cross-validation attestation for JCS RFC 8785", 2026-05-24, target="https://github.com/chopmob-cloud/algovoi-jcs-conformance-vectors/blob/main/_attestations/2026-05-24-8-impl-cross-validation.md" (https://github.com/chopmob-cloud/algovoi-jcs-conformance-vectors/blob/main/_attestations/2026-05-24-8-impl-cross-validation.md)
* [AlgoVoi-Substrate-Authorship] AlgoVoi, "Substrate Authorship and Provenance", target="https://docs.algovoi.co.uk/substrate-authorship-provenance" (https://docs.algovoi.co.uk/substrate-authorship-provenance)
* EU Markets in Crypto-Assets Regulation (MiCA, Regulation (EU) 2023/1114), Article 80.
* EU Anti-Money Laundering Regulation (AMLR, Regulation (EU) 2024/1624), Article 56.
* EU Digital Operational Resilience Act (DORA, Regulation (EU) 2022/2554), Article 14.

Appendix B. Appendix A. Example Canonical Forms (Informative)

B.1. A.1. Integer-timestamp normalisation (Substrate Rule 2)

Input object (non-conforming, RFC 3339 string):

```
{
  "agent_id": "did:web:api.algovoi.co.uk",
  "action_type": "compliance_screen",
  "scope": "algovoi:compliance_screen",
  "timestamp": "2024-05-28T12:00:00Z"
}
```

Validator MUST reject (Section 4.5). Conforming object:

```
{
  "agent_id": "did:web:api.algovoi.co.uk",
  "action_type": "compliance_screen",
  "scope": "algovoi:compliance_screen",
  "timestamp_ms": 1716897600000
}
```

JCS canonical bytes (sorted keys, no whitespace):

```
{"action_type":"compliance_screen","agent_id":"did:web:api.algovoi.co.uk","scope":"algovo
i:compliance_screen","timestamp_ms":1716897600000}
```

SHA-256 content_hash (lowercase hex):

```
7528529a8be2044488e603b7913efaa4f83620dbcc63010d4a1478cf7e9a473c
```

This digest is the action_ref value for the example.

B.2. A.2. Array order is load-bearing

Two otherwise-identical receipt fragments varying only jurisdiction_flags array order:

```
{ "jurisdiction_flags": ["UK", "EU"], ... }
{ "jurisdiction_flags": ["EU", "UK"], ... }
```

These produce different JCS canonical bytes (RFC 8785 Section 3.2.3 preserves array order) and therefore different content hashes. The receipt format using this discipline MUST specify a canonical array order at the schema layer (e.g. "primary jurisdiction first").

B.3. A.3. canon_version pin

Two otherwise-identical receipts varying only canon_version:

```
{ "canon_version": "jcs-rfc8785-v1", ... }
{ "canon_version": "jcs-rfc8785-v2", ... }
```

These produce different canonical bytes. A receipt emitted under v1 cannot be silently re-hashed under v2.

Appendix C. Appendix B. Reference Implementations (Informative)

The following open-source implementations conform to this canonicalisation discipline:

Language	Package	Function
Python	target="https://pypi.org/project/algovoi-substrate/" (https://pypi.org/project/algovoi-substrate/)	algovoi_substrate.canonicalize
TypeScript	target="https://www.npmjs.com/package/@algovoi/substrate" (https://www.npmjs.com/package/@algovoi/substrate)	canonicalize, sha256Jcs

Table 3

Both packages depend transitively on rfc8785 (Python) / canonicalize (TypeScript) for the JCS layer and add the schema-normalisation discipline of Section 4 on top.

The conformance vector corpus that anchors this discipline:

https://github.com/chopmob-cloud/algovoi-jcs-conformance-vectors

The eight-implementation cross-validation attestation referenced in Section 7 is at:

_attestations/2026-05-24-8-impl-cross-validation.md

Appendix D. Appendix C. Known Adopters (Informative)

The following downstream parties have published artefacts that anchor to the canonicalisation discipline specified by this document. Inclusion in this list is informational and reflects public adoption only; it does not imply endorsement or normative authority from the listed party.

Adopter	Surface	Anchor
AlgoVoi (api.algovoi.co.uk)	Production compliance and refund-receipt issuer	All AlgoVoi-emitted receipts under canon_version: jcs-rfc8785-v1
Supership (andysalvo)	service_trust_v0 conformance vectors	Downstream-adopter vector set anchored to the canonicalisation discipline; submitted to the AlgoVoi corpus on 2026-05-23 and recorded as an independent adoption signal

Table 4

Adopters publishing vector sets or receipt-format extensions that anchor to this discipline are encouraged to publish them in adopter-controlled repositories with canon_version recorded in-band, so each adopter’s authorship is unambiguous and their artefact is independently citable.

This appendix is maintained as a record of observed adoption at the time of revision; absence from this list is not normative.

Appendix E. Appendix D. Acknowledgments

This document, and the canonicalisation discipline it specifies, are AlgoVoi work under sole AlgoVoi authorship.

The canonicalisation discipline persists under AlgoVoi-controlled surfaces: this Internet-Draft, the substrate authorship provenance record at target="https://docs.algovoi.co.uk/substrate-authorship-provenance" (https://docs.algovoi.co.uk/substrate-authorship-provenance), the algovoi-substrate reference implementation on PyPI and npm, and the eight-implementation cross-validation attestation record in the AlgoVoi conformance vectors corpus.

The author acknowledges Anders Rundgren as the editor of RFC 8785, the JSON Canonicalization Scheme on which this discipline builds.

Author’s Address

Christopher Hopley
AlgoVoi
Email: chopmob@gmail.com