Network Working Group
Internet-Draft
Intended status: Informational
Expires: 24 November 2026

S. Jovancevic
SKGO, IKT Support

23 May 2026

Verifiable Data Access Contract (VDAC)
draft-jovancevic-vdac-01

Abstract

This document specifies the Verifiable Data Access Contract (VDAC), a protocol for cryptographically verifiable bilateral agreement between a content publisher and an automated agent regarding the terms of programmatic data access. VDAC defines the mechanism by which a site publishes an access offer, an agent accepts that offer, both parties sign the resulting contract, and per-request references bind individual interactions to agreed terms.

VDAC is the protocol-layer realization of the bilateral commitment principle introduced in Section 6.6 of the Verifiable Identity Claims and Delegation Model [VICDM] and operates as a companion specification to the Signed Agent Identity Protocol [SAIP].

This document defines mechanism, not content: VDAC verifies the existence and integrity of an agreement; the substance of what is agreed remains entirely between the contracting parties.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 24 November 2026.

Copyright Notice

Table of Contents

1. Introduction

The Verifiable Data Access Contract (VDAC) provides a cryptographic mechanism by which a content publisher (the Site) and an automated client (the Agent) can establish a verifiable bilateral agreement regarding the terms of programmatic data access.

Modern content publishers face a recurring problem: automated agents -- AI training crawlers, retrieval systems, monitoring services, IoT telemetry consumers -- access publisher resources at scale, often outside the scope envisioned by the publisher. Existing mechanisms address this only partially:

- robots.txt declares what a site permits, but is unilateral, advisory, and cryptographically unverified.

- Terms-of-service URLs declare legal terms, but bind no party cryptographically and are difficult to enforce per-request.

- Anonymous attestation models (such as those developed in the IETF webbotauth working group) verify that a bot is vouched for, but do not establish any agreement between bot and site about what the bot will do.

- Identity attestation models (such as [SAIP]) verify who the bot is, but make no statement about agreed terms of access.

VDAC fills the gap between identity verification and behavioral agreement. Once a Site and Agent have both signed a VDAC contract, each subsequent request from the Agent references the contract cryptographically, and both parties retain non-repudiable logs of the actual interaction. Discrepancies between intent and behavior become provable rather than asserted.

1.1. Relationship to VICDM

This document develops the bilateral agreement mechanism originally introduced in Section 6.6 of [VICDM]. VDAC is extracted into a standalone specification to enable focused review and independent deployment.

VDAC implements the fifth principle of the VICDM model:

> Agreed intent MUST be honored bilaterally.

An Agent operating under a valid VDAC contract is classified as Class 4 (Contracted Agent) under the VICDM trust model, receiving the highest trust level in that classification.

The two documents are companion specifications: [VICDM] provides the conceptual framework and identity classification within which VDAC operates; this document defines the wire protocol, document formats, and operational mechanics.

1.2. The Emerging Architectural Stack

The automated web ecosystem naturally separates into distinct functional layers. Each layer is a responsibility, and one or more protocols may fulfill that responsibility:

Layer	Responsibility	Protocol(s)
Identity	Establishing a verifiable agent identity	SAIP (full identity framework), or VICDM DNS-based verification
Framework	Trust classification and delegation	VICDM
Intent	Unilateral intent declaration	ALOI (defined in VICDM)
Agreement	Bilateral operational agreement	VDAC

The Identity layer is deliberately shown with two protocol options. SAIP and VICDM both provide identity verification, by different mechanisms: SAIP is a complete identity and signature framework with per-request key rotation and related operational features, while VICDM additionally defines a lighter-weight DNS-based verification suitable for agents that do not need the full SAIP machinery. A protocol at a higher layer -- such as VDAC -- consumes "an identity", not "SAIP specifically". This is what makes the stack composable: each layer can be satisfied by whichever protocol fits the deployment.

This separation is fundamental to modern automated network interactions because identity trust does not equate to behavioral trust. Cryptographic identity alone cannot establish operational accountability. VDAC introduces an explicit, operational layer where identity answers "who you are", and VDAC answers "whether you honored what was agreed".

1.3. Relationship to SAIP and ALOI

VDAC requires that every participating Agent possess a verified identity: a cryptographically verifiable public key bound to the Agent and discoverable by the Site. VDAC does NOT mandate any single identity mechanism.

The Signed Agent Identity Protocol [SAIP] is the RECOMMENDED identity mechanism for VDAC, as it provides a complete identity and signature framework designed for automated agents. However, SAIP is not required. An Agent MAY establish its identity through any of the following:

- SAIP -- the RECOMMENDED full identity framework.
- The DNS-based identity verification described in [VICDM] Section 5, which publishes an Agent public key in DNS without requiring the full SAIP machinery. This is a lighter-weight option suitable for Agents that do not need per-request key rotation or the other operational features of SAIP.
- Any equivalent mechanism that provides a verifiable Agent public key, provided both Site and Agent can rely on it.

VDAC and SAIP are complementary, not interdependent. VDAC defines the bilateral contract layer; how the Agent proves its identity is a separate concern that VDAC deliberately leaves open. Where this document refers to an Agent’s signing key, that key is whatever public key the chosen identity mechanism establishes.

VDAC is complementary to the Agent Letter of Intent (ALOI) defined in Section 6.5 of [VICDM]:

- ALOI is unilateral: the Agent declares its intent in DNS, and any Site may observe and enforce that declaration.
- VDAC is bilateral: a specific Site and a specific Agent enter into a mutually signed agreement that supersedes general ALOI declarations for interactions under the contract.

An Agent MAY operate under ALOI alone (Intent-Declaring mode), under VDAC with a specific Site (Contracted Agent mode), or under both simultaneously. When in conflict, the more specific commitment governs: VDAC terms supersede ALOI declarations for requests under the contract; ALOI continues to apply outside any VDAC contract.

1.3.1. Identity Mechanism Requirements

VDAC is identity-mechanism-agnostic. The protocol does not privilege any particular identity system. Any identity mechanism MAY be used as the Identity layer for VDAC, provided it satisfies all of the following requirements:

- It MUST yield a public key that is cryptographically bound to the Agent.
- It MUST allow the Site to verify that public key independently, without contacting the Agent out-of-band.
- It MUST produce per-request signatures that can cover the contract-id and contract-hash parameters defined in Section 7, so that each request is bound to a specific contract version.

Any mechanism meeting these three requirements is suitable as the VDAC Identity layer.

SAIP and the VICDM DNS-based verification are two such mechanisms specified within this document family. They are RECOMMENDED for convenience and for interoperability across independent implementations, but they are in no way privileged by the protocol itself. A VDAC implementation that relies on neither SAIP nor VICDM is fully conformant, provided its chosen identity mechanism meets the requirements above.

Implementers MAY therefore use existing identity infrastructure -- including mechanisms not authored within this document family, and including organization-internal identity systems -- wherever that infrastructure satisfies the stated requirements. The Identity layer is a replaceable component, not a fixed dependency. This is a deliberate design property: it keeps VDAC composable and avoids binding adopters to any single identity ecosystem.

The agent.id_method field in the Acceptance Document (Section 5.1) names the identity mechanism actually in use, allowing the Site to apply the correct verification procedure.

1.4. Relationship to webbotauth Work

The IETF webbotauth working group is developing mechanisms for anonymous bot attestation [WEBBOTAUTH-ARCH]. VDAC is complementary to that work:

- Anonymous attestation addresses the privacy use case: rate limiting and abuse prevention without per-instance disclosure.
- VDAC addresses the bilateral accountability use case: explicit mutual agreement on terms of access, with cryptographic evidence retained by both parties.

A deployment MAY support anonymous attestation for general traffic and require VDAC for specific access tiers (e.g., bulk data retrieval, AI training, premium API endpoints). The two mechanisms address different threat models and need not be mutually exclusive.

1.5. Protocol Scope and Non-Goals

VDAC defines the mechanism for verifiable bilateral agreement. It addresses:

- Offer publication by Sites.
- Acceptance by Agents.

- Cryptographic binding of both parties to agreed terms.
- Per-request contract reference.
- Mutual tracking and audit obligations.
- Dispute evidence format.
- Contract termination.

VDAC explicitly does NOT address:

- Pricing, payment, or commercial settlement mechanisms.
- Legal enforcement, arbitration, or jurisdiction.
- Tax, regulatory, or compliance matters.
- Negotiation protocols (VDAC offers are take-it-or-leave-it from the protocol's perspective; out-of-band negotiation may precede the protocol).

These elements are private between the contracting parties or addressed by other documents. VDAC provides only the mechanism for verifiable agreement.

2. Terminology

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.

Site:	The party publishing content and offering access terms. Identified by a DNS domain.
Agent:	The automated client seeking access. Holds a verified identity providing a cryptographically verifiable public key (see Section 1.3).
Offer Document:	A signed JSON object published by the Site describing access terms.
Acceptance Document:	A signed JSON object produced by the Agent indicating intent to operate under a specific Offer.
Contract Document:	The mutually signed JSON object representing the final bilateral agreement. Contains both the Offer and the Acceptance, plus signatures from both parties.
Contract Identifier:	A unique value identifying a specific Contract Document, used as a per-request reference.
Contract Hash:	SHA-256 hash of the canonical Contract Document, used to bind individual requests cryptographically to a specific contract version.

3. Conceptual Overview

3.1. The Progression from Anonymous to Contracted

VDAC sits at the top of the VICDM interaction progression:

Interaction Mode	Identity	ALOI	VDAC

Anonymous	No	No	No
Identified	Yes	No	No
Intent-Declaring	Yes	Yes	No
Contracted Agent	Yes	Yes opt.	Yes

Each level is opt-in. No Agent is compelled to enter a VDAC contract; no Site is compelled to publish a VDAC Offer. Where bilateral terms make sense, the mechanism is available.

3.2. Programmable Bilateral Consent

VDAC introduces a transition from static legal declarations to machine-verifiable operational consent. Traditional mechanisms are unsuited
for programmatic execution: robots.txt is advisory only, Terms of Service
documents are legally descriptive but unparseable by systems, and API keys
grant entry without tracking ongoing behavioral obligations.

Under the Programmable Bilateral Consent model:
1. Both parties cryptographically bind themselves to operational scopes.
2. The operational scope is explicitly machine-readable.
3. Violations become algorithmically provable rather than merely asserted.

3.3. Contracted Web Semantics

VDAC establishes a structural shift in web interaction philosophy. The historical web model relies on anonymous, unilateral, advisory, and loosely enforceable interactions. VDAC formalizes mutually authenticated interactions, cryptographically verifiable obligations, and
bilateral accountability. This introduces a transition from advisory web
semantics toward verifiable bilateral interaction semantics.

3.4. Separation of Identity Trust and Behavior Trust

A critical distinction within this architecture is the complete decoupling of identity verification from behavioral tracking:

- Identity Trust establishes who the Agent is, whether the identity is cryptographically valid, and whether delegation roots are intact. This is provided by an identity mechanism such as SAIP or the DNS-based verification of [VICDM].

- Behavior Trust (VDAC) verifies whether the Agent and Site have honored the underlying agreement over time, and preserves independent mathematical logs.

Because a valid identity does not automatically guarantee compliant behavior, VDAC serves as the explicit operational accountability layer above whatever identity mechanism is in use.

3.5. Mechanism, Not Content

VDAC is a protocol primitive. It cryptographically establishes that:

- The Site offered specific terms at a specific time.

- The Agent accepted those specific terms at a specific time.
- Both parties bound themselves to the contract by signature.
- Each subsequent request was made under the claimed contract.
- Both parties retained matching logs of the interaction.

What the terms ARE -- whether they permit AI training, restrict to ten requests per minute, require attribution, exclude personal data paths, or impose specific structural limits -- is outside the scope of this document. VDAC verifies that an agreement exists and is honored. The wisdom or commercial value of the agreement is for the contracting parties to determine.

4. Site Offer Document

4.1. Publication Location

A Site that wishes to enable VDAC-based access SHOULD publish an Offer Document at a well-known location, per [RFC8615]:

```
https://<site-domain>/.well-known/vdac-offer
```

When multiple Offers are published, they MAY be available at versioned paths:

```
https://<site-domain>/.well-known/vdac-offer/<offer-id>
```

The Offer Document MUST be served over TLS. Servers MAY include the Offer Document inline in API responses where convenient, but the well-known URL remains the authoritative source.

4.2. Offer Document Format

The Offer Document is a JSON object signed by the Site’s identity key:

```
{
  "offer_id":     "premium-ai-training-v1",
  "site": {
    "domain":     "site.domain",
    "pubkey":     "pX023...[Truncated Base64URL]"
  },
  "valid_from":   1779369600,
  "valid_until":  1795132800,
  "terms": {
    "scope":      ["/api/v1/public/*", "/articles/archived/*"],
    "exclusions": ["/api/v1/public/users/*", "/articles/premium/*"],
    "rate_limit": {
      "window_seconds": 60,
      "requests_per_window": 120,
      "burst_allowance": 15,
      "max_concurrent_connections": 4,
      "bandwidth_cap_bytes_per_day": 1073741824
    },
    "obligations": ["attribution_required", "no_resell"],
    "custom_terms_uri": "https://site.domain/legal/tos-ai.html",
    "custom_terms_hash": "e3b0c44298fc1c149afbf4c8996fb92427..."
  },
  "offer_sig":    "SigVal...[Truncated Base64URL]"
}
```

Field semantics:

offer_id
: Unique identifier issued by the Site. MUST be unique within the scope of the Site for the Offer’s validity period.

site.domain	The DNS domain of the Site, matching the host portion of the publication URL.
site.pubkey	The Site’s Ed25519 public key. MUST correspond to the private key used to produce offer_sig.
valid_from	Unix timestamp before which the Offer MUST NOT be accepted.
valid_until	Unix timestamp after which the Offer MUST NOT be accepted.
terms	Substantive content of the Offer.
terms.scope	An array of URL path matchers defining permitted endpoints. Pattern matching follows standard glob/wildcard semantics relative to the site root path.
terms.exclusions blocks	An array of URL path matchers defining explicit within the permitted scope. Exclusions MUST override scope definitions when paths overlap.
terms.rate_limit Parsers	Stated operational performance constraints. MUST support the primitive integer definitions for windows, requests, concurrency, and optional daily bandwidth caps.
terms.obligations	Free-form list of obligations the Agent agrees to fulfill if accepting. The protocol does not parse these strings natively.
terms.custom_terms_uri	An optional URI pointing to additional contract terms that do not fit within JSON. If present, the contents MUST be hashed and the hash included in "custom_terms_hash" computed at Offer publication time.
offer_sig	Ed25519 signature over the canonicalized offer object excluding the signature field itself. Canonicalization follows [RFC8785].

4.3. Multiple Concurrent Offers

A Site MAY publish multiple Offers concurrently representing different access tiers. Each Offer MUST have a distinct offer_id. The set of currently valid Offers MAY be enumerated at:

https://<site-domain>/.well-known/vdac-offer-index

4.4. Discovery and Capability Advertisement

To minimize the application layer overhead of fetching offer indexes proactively, implementations SHOULD leverage discovery mechanisms.

4.4.1. HTTP Capability Advertisement

Sites MAY advertise contract capabilities during standard HTTP interactions
by including a VDAC-Advertise header in responses:

```
VDAC-Advertise:
  offer="https://site.domain/.well-known/vdac-offer",
  types="ai-training, retrieval",
  index="https://site.domain/.well-known/vdac-offer-index"
```

4.4.2. DNS-Based Discovery

Agents wishing to evaluate a domain's VDAC enforcement posture before initiating application-layer handshakes MAY query the zone's DNS records.

The VDAC pointer is constructed as a TXT record located at a dedicated subdomain string:

```
_vdac.site.domain. IN TXT
  "v=vdac1;
   index=https://site.domain/.well-known/vdac-offer-index"
```

5. Agent Acceptance Document

5.1. Acceptance Format

An Agent that wishes to accept a Site Offer constructs an Acceptance Document:

```
{
  "contract_id":  "Z21h...[Truncated Base64URL]",
  "offer_hash":   "SHA256Val...[Truncated Base64URL]",
  "agent": {
    "agent_id":   "id=master-agent-identity",
    "id_method":  "saip",
    "pubkey":     "aF912...[Truncated Base64URL]",
    "vendor":     "vendor-domain.com",
    "delegation_allowed": true
  },
  "accepted_at":  1779370000,
  "expires_at":   1795132800,
  "agent_sig":    "SigVal...[Truncated Base64URL]"
}
```

Field semantics:

contract_id	The Contract Identifier derived per Section 6.2. The Agent computes this value at acceptance time.
offer_hash	SHA-256 hash of the canonicalized Offer Document. This binds the Acceptance to a specific version of the Offer.
agent.agent_id	The Agent's identity string, as established by the chosen identity mechanism (Section 1.3). For SAIP, this is the id= value per [SAIP].
agent.id_method	The identity mechanism in use. Defined values: "saip" (SAIP identity), "vicdm-dns" (DNS-based verification per [VICDM] Section 5), or an implementation-defined string for other mechanisms.
agent.pubkey	The Agent's public key. MUST match the key discoverable through the identity mechanism named in id_method (for SAIP, the key in the Agent's DNS _saip record; for vicdm-dns, the key published per [VICDM] Section 5).

agent.vendor	The DNS domain of the vendor operating the Agent.
agent.delegation_allowed	A boolean flag declaring whether the contract intends to utilize distributed worker nodes authenticated via VICDM delegation structures.
accepted_at	Unix timestamp at which the Acceptance was constructed. MUST be within the Offer's validity period.
expires_at	Unix timestamp at which the resulting Contract expires.
agent_sig	Ed25519 signature over the canonicalized acceptance object excluding the signature field.

5.2. Submission and Server Response

The Agent submits the Acceptance Document to the Site via HTTP POST:

```
POST https://<site-domain>/.well-known/vdac-accept
Content-Type: application/json
```

The request body is the Acceptance Document. If validation succeeds, the Site returns the complete signed Contract Document (Section 6) with HTTP status 201 (Created).

If validation fails, the Site returns HTTP status 400 (Bad Request) with structured error codes: 'offer_not_found', 'offer_expired', 'duration_exceeds', 'signature_invalid', 'identity_unverified', or 'duplicate_contract'.

6. Contract Document

6.1. Contract Format

The Contract Document is the canonical record of the bilateral agreement:

```
{
  "contract_id":  "DerivedID-Value",
  "offer":        { ... complete Offer Document ... },
  "acceptance":   { ... complete Acceptance Document ... },
  "site_sig":     "SigVal...[Truncated Base64URL]",
  "agent_sig":    "SigVal...[Truncated Base64URL]"
}
```

The site_sig and agent_sig fields are Ed25519 signatures computed over the canonicalized concatenation of the offer and acceptance objects (in that order), per [RFC8785].

6.2. Contract Identifier Derivation

The contract_id is deterministically derived to allow either party to compute it independently:

```
contract_id = Base64URL(SHA256(
                  offer_id || agent.agent_id || accepted_at
              ))
```

Where || denotes UTF-8 string concatenation with a single

ASCII null byte (0x00) as a strict delimiter.

6.3. Contract Storage Obligations

Both parties MUST retain identical copies of the Contract Document for the duration specified in terms.duration plus the log retention period (Section 8.4).

6.4. Delegated Contracting and Infrastructure Scale

For large-scale, enterprise, and distributed systems where scraping or API consumption is handled by clustered pools of machines, executing unique contracts per ephemeral node is unfeasible. VDAC addresses this by integrating with the [VICDM] delegation mechanism.

If ‘agent.delegation_allowed‘ is true in the signed contract, application requests operating under the contract_id MAY be signed by distinct, ephemeral operational public keys. The Agent node MUST include the matching umbrella corporate ‘Delegation Token‘ within the request protocol metadata, proving that the ephemeral node’s key is a legitimate downstream delegate of the contract’s ‘agent.pubkey‘.

7. Per-Request Contract Reference

Once a Contract is established, the Agent includes a contract reference in every signed request to the Site:

```
contract-id=<contract_id>; contract-hash=<contract_hash>
```

Where contract_hash is the SHA-256 hash of the canonicalized Contract Document, encoded as Base64URL.

The contract-id and contract-hash parameters MUST be covered by the request signature produced by the Agent’s identity mechanism, so that each request is cryptographically bound to a specific contract version. When SAIP is the identity mechanism, these parameters MUST be included in the SAIP canonical signature string per [SAIP] Section 6.4.2. When another identity mechanism is used, the parameters MUST be covered by that mechanism’s request signature in an equivalent manner.

8. Mutual Tracking Obligation

VDAC requires mutual logging by both parties. Tracking is an integral part of the contract, not an optional feature.

8.1. Site-Side Log Entry Format

```
{
  "contract_id":   "DerivedID-Value",
  "ts":            1779371050,
  "endpoint":      "/articles/archived/news-item",
  "method":        "GET",
  "bytes_sent":    45120,
  "status_code":   200,
  "agent_sig":     "SigVal...[Truncated Base64URL]",
  "site_log_sig":  "SigVal...[Truncated Base64URL]"
```

```
}
```

8.2. Agent-Side Log Entry Format

```
{
  "contract_id":         "DerivedID-Value",
  "ts":                  1779371050,
  "endpoint":            "/articles/archived/news-item",
  "method":              "GET",
  "response_hash":       "SHA256Val...[Truncated Base64URL]",
  "bytes_received":      45120,
  "agent_log_sig":       "SigVal...[Truncated Base64URL]"
}
```

8.3. End-User Privacy Boundaries

This section addresses a critical constraint: VDAC tracking MUST NOT become a vector for surveillance of end-users whose interactions are mediated by a contracted Agent.

The following constraints apply to all VDAC log entries:

1. End-user identifying information MUST NOT be included in VDAC log entries. This includes, but is not limited to: end-user IP addresses, user-agent strings of human users whose interaction is proxied by the Agent, end-user authentication tokens, session identifiers, and personally identifiable URL parameters.

2. The endpoint field MUST be normalized to exclude query parameters that could identify end-users. Implementations MUST log only the path component, or a redacted form of the query string where retention of certain parameters is operationally necessary.

3. Response content MUST NOT be logged in raw form. Only a hash (response_hash) is logged. The hash provides verification capability without retaining the content itself.

8.4. Log Retention

Both parties retain logs for the duration specified in the contract terms. The RECOMMENDED minimum retention is 90 days from the request timestamp. Logs older than the retention period SHOULD be securely deleted.

9. Reconciliation and Dispute Evidence

Periodic reconciliation between Site and Agent logs is RECOMMENDED. To facilitate rapid, automated cross-checking without mass data transfer,
parties utilize a structural summary document.

9.1. Reconciliation Manifest Format

When performing an audit checkpoint, a party compiles state records into a
canonical Manifest object:

```
{
  "contract_id": "DerivedID-Value",
  "period_start": 1779369600,
  "period_end": 1779456000,
  "total_requests": 14500,
```

```
  "total_bytes": 48291045,
  "log_summary_hash": "SHA256HashOfAllCompiledRequestSignatures...",
  "manifest_sig": "SigVal...[Truncated Base64URL]"
}
```

If both local calculations match the ‘log_summary_hash‘, the historical block is mutually verified. If hashes diverge, systems enter detailed reconciliation by breaking the window into narrower sub-intervals to isolate the precise mismatched request signatures.

10. Violation Detection and Sanctions

10.1. Violation Notice Format

```
{
  "contract_id":   "DerivedID-Value",
  "violation":     "rate_limit_exceeded",
  "evidence_ref":  "ManifestHashOrLogID",
  "detected_at":   1779372000,
  "site_sig":      "SigVal..."
}
```

The notice MAY be delivered via HTTP response header on the offending request:

```
VDAC-Violation: <base64url-encoded violation notice>
```

10.2. Sanction Progression

Sanctions for VDAC violations SHOULD follow the same progressive model defined for ALOI in [VICDM] Section 6.5.7:

Violation Count	Sanction	Action
1	Warning	Notice only; no rate change.
2-3	Throttle	Reduce to minimal rate.
4-5	Downgrade	Trust downgrade.
6-10	Block	Reject contract requests.
>10	Termination	Terminate contract per Section 11.

10.3. Self-Reporting (Good Faith Mechanism)

An Agent that detects an unintentional violation of contract terms MAY self-report the violation before the Site detects it, per the Good Faith Mechanism defined in [VICDM] Section 6.5.8.

Self-reporting under VDAC uses the same signal format as ALOI self-reporting:

```
VDAC-Self-Report: contract-id=<contract_id>;
                  violation=<violation-type>;
                  ts=<unix-timestamp-of-violation>;
                  self-sig=<signature per VICDM 6.5.8>
```

11. Contract Termination

11.1. Termination Conditions

A VDAC contract terminates in any of the following conditions:

- Natural expiry: reaching the contract's expires_at timestamp.
- Mutual termination: both parties signing a termination notice.
- Unilateral termination by either party, with signed notice.
- Material breach: severe violation as defined in contract terms.

11.2. Termination Notice Format

```
{
  "contract_id":      "DerivedID-Value",
  "terminated_by":    "site",
  "reason":           "material_breach",
  "effective_at":     1779373000,
  "evidence_ref":     "ViolationNoticeHash",
  "terminator_sig":   "SigVal..."
}
```

Defined reason codes: 'natural_expiry', 'mutual_consent', 'site_initiated', 'agent_initiated', 'material_breach', or 'offer_revoked'.

11.3. Post-Termination Obligations

After termination, the Agent MUST cease using the contract_id in requests.
Both parties retain logged evidence for the contract's log retention period.
The Agent reverts to its prior interaction mode (Anonymous, Identified, or Intent-Declaring per the VICDM progression).

12. Relationship to ALOI

ALOI and VDAC are complementary: ALOI is a unilateral declaration, whereas VDAC is a bilateral agreement where both parties are bound.

An Agent MAY operate under both simultaneously. ALOI defines the Agent's general operational scope across all interactions; VDAC defines specific terms for one particular Site. When in conflict, the more specific commitment governs: VDAC terms supersede ALOI declarations for requests under the contract.

13. Use Case Profiles (Informative)

13.1. AI Training Data Access

An AI model vendor wishes to access bulk content from a publisher Site for training purposes. The structural 'rate_limit' block maps a macro configuration (e.g., aggregate daily byte caps, low multi-concurrency connections) alongside obligations like 'no_resell' and explicit content path exclusions.

13.2. Content Retrieval and Inference Access

A retrieval-augmented AI assistant accesses individual articles in real time

to answer user queries. Configuration profiles employ lower window boundaries, higher
burst thresholds, and shorter termination spans. Under this profile, the privacy
constraints of Section 8.3 are strictly critical.

13.3. Monitoring and Health-Check Access

A third-party monitoring service accesses Site endpoints to verify availability.
The profile establishes lightweight logging structures with simple token
structures and predictable heartbeat windows.

14. Security Considerations

14.1. Signature Forgery

The integrity of VDAC depends entirely on the security of the Ed25519 signature scheme [RFC8032] and the secrecy of private keys. Agent private keys SHOULD be protected by hardware-backed key storage (TPM, HSM, or Secure Enclave). When SAIP is the identity mechanism, Agent keys are protected per [SAIP] key protection requirements.

14.2. Replay of Acceptance

Resubmission of the same Acceptance Document produces an identical contract_id (per Section 6.2). The Site MUST detect duplicate contract_id values and reject duplicate Acceptance submissions.

14.3. Offer Substitution

The offer_hash field in the Acceptance Document binds the Acceptance to a specific Offer version. A Site cannot substitute different terms after Acceptance without invalidating agent_sig.

15. Privacy Considerations

See Section 8.3 (End-User Privacy Boundaries) for the core privacy requirements applicable to VDAC tracking. Because reconciliation logs map hashes and paths rather than raw states, they protect end-user activity while preserving behavioral
audit trails.

16. IANA Considerations

This document requests IANA to register the following well-known URIs per [RFC8615]:
‘vdac-offer‘, ‘vdac-offer-index‘, ‘vdac-accept‘, and ‘vdac-contract‘.

This document requests IANA to register the following HTTP header fields per [RFC3864]:
‘VDAC-Violation‘ and ‘VDAC-Self-Report‘.

This document requests IANA to create the VDAC Termination Reason Codes registry.

17. References

17.1. Normative References

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, January 2017.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, May 2019.

[RFC8785] Rundgren, A., "JSON Canonicalization Scheme (JCS)", RFC 8785, June 2020.

[RFC3864] Klyne, G., "Registration Procedures for Message Header Fields", RFC 3864, September 2004.

[VICDM] Jovancevic, S., "Verifiable Identity Claims and Delegation Model (VICDM)", Work in Progress, Internet-Draft, draft-jovancevic-vicdm-05, 18 May 2026.

[SAIP] Jovancevic, S., "SAIP: Signed Agent Identity Protocol", Work in Progress, Internet-Draft, draft-jovancevic-saip-08, April 2026.

17.2. Informative References

[WEBBOTAUTH-ARCH]
Meunier, T. and S. Major, "HTTP Message Signatures for automated traffic Architecture", Work in Progress, Internet-Draft, draft-meunier-web-bot-auth-architecture, March 2026.

Appendix A. Document History

The Verifiable Data Access Contract (VDAC) was originally defined in Section 6.6 of draft-jovancevic-vicdm-05. It was extracted into this standalone specification to enable focused review and independent versioning.

Changes in draft-jovancevic-vdac-01:

- Made the identity layer mechanism-independent. SAIP is now RECOMMENDED rather than mandatory. Section 1.3 rewritten to permit SAIP, VICDM DNS-based verification, or any equivalent identity mechanism providing a verifiable Agent public key.
- Added Section 1.3.1 (Identity Mechanism Requirements) defining the three properties any identity mechanism must satisfy to serve as the VDAC Identity layer, and stating explicitly that no identity system is privileged by the protocol.
- Reorganized the Section 1.2 architectural stack table by functional layer rather than by protocol, showing that the Identity layer can be satisfied by SAIP or by VICDM DNS-based verification.
- Renamed the agent.saip_id field to agent.agent_id and added an agent.id_method field naming the identity mechanism in use.
- Section 7 (Per-Request Contract Reference) generalized so that contract parameters are covered by whatever request signature the chosen identity mechanism provides.
- Editorial: corrected line wrapping in Sections 3.4, 14, 15.

Changes relative to VICDM Section 6.6 (in draft -00):

- Expanded Privacy Considerations with explicit Section 8.3 boundaries.
- Added Section 4.4 Capability Discovery architectures (HTTP/DNS).
- Formalized structural standard fields for rate_limit JSON.
- Introduced Section 6.4 Enterprise Infrastructure Key Delegation.
- Added Section 9.1 Fast Reconciliation Manifest hashes.

Author’s Address

Srecko Jovancevic
SKGO, IKT Support
Makedonska 22
11000 Belgrade
Serbia

Email: srecko.jovancevic@skgo.org
Email: srecko.jovancevic@gmail.com
URI: https://github.com/sreckojovancevic