

# Auto-Registration Permanent Archive Service: Implementation Specification

自動登録型永続アーカイブサービス 実装仕様書

Viorazu.

---

## このドキュメントの目的

サイト所有者本人が公開したコンテンツを、公開と同時に永続保存し、改ざん不能な形で後世に引用可能にするサービス。その実装に必要な技術要素と設計判断を、開発者がそのまま読んで着手できる形でまとめる。

対象読者：バックエンド開発者、インフラ担当、プロダクトマネージャー。

---

## 目次

1. サービス全体像
2. システム構成
3. 本人性の担保
4. コンテンツ保存
5. 改ざん証明
6. 参照機能
7. 削除ポリシー
8. プラットフォーム連携
9. API仕様
10. データベース設計

# 1. サービス全体像

## 1.1 一行で言うと

サイト所有者が自分のプラットフォーム管理画面でスイッチをONにするだけで、以後の投稿が自動的に永続保存され、暗号学的に改ざん証明される月額サービス。

## 1.2 ユーザー体験

1. WordPress、note、Wixなどの管理画面で「永続アーカイブを有効化」をON
2. 月額プランを選択
3. 以後、記事を書いて公開するたびに自動保存
4. 必要な時に証明書をPDFで発行できる

ユーザーが意識する操作はONにする瞬間だけ。

## 1.3 何が新しいか

既存サービスとの違い：

- Internet Archive：第三者が勝手に保存、本人確認なし → **本サービスは本人登録専用**
- Zenodo：ファイル単位、動的サイト非対応 → **本サービスはウェブサイト丸ごと対応**
- 既存魚拓：第三者保存が主 → **本サービスは所有者主体の自動登録**
- Googleキャッシュ（廃止済み）：検索補助、証明能力なし → **本サービスは法的証拠能力を持つ**

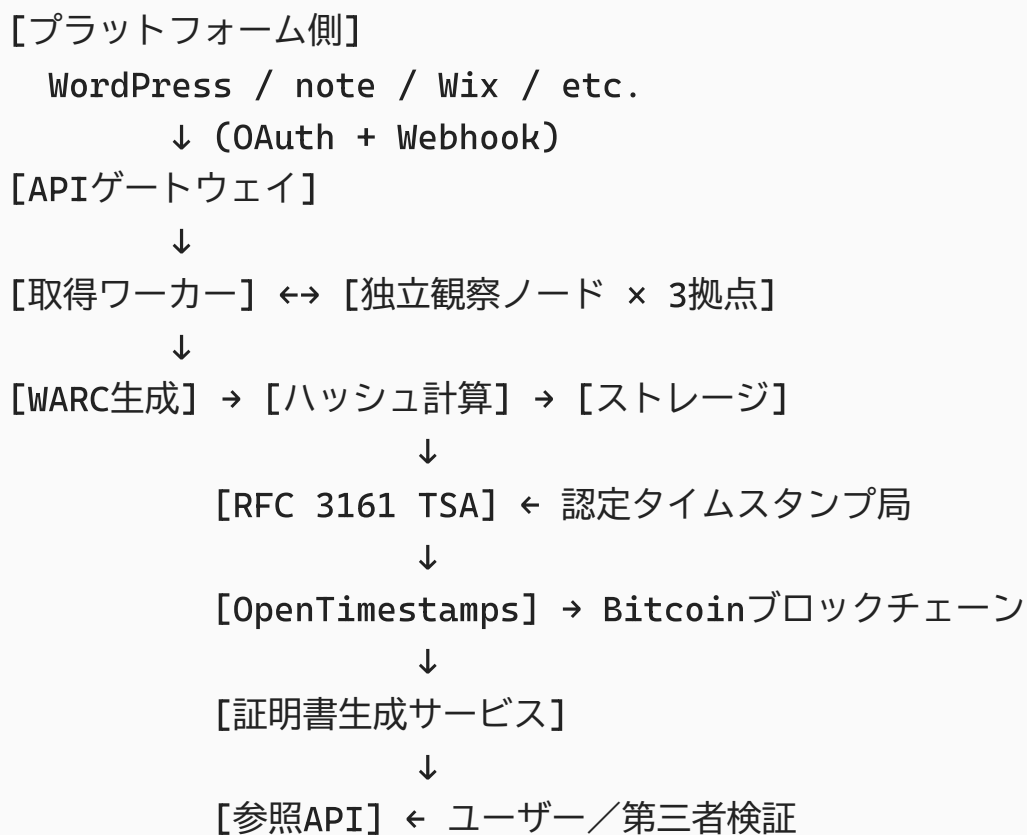
## 1.4 解決する問題

AI時代に、自分の書いた内容を「〇月〇日にこの形で存在していた」と証明する手段がない。AI学習データの引用元、論文の引用性担保、法的証拠、いずれも現状は脆弱。本サービスはその空白を埋める。

---

## 2. システム構成

### 2.1 構成要素



### 2.2 主要コンポーネント

**APIゲートウェイ** プラットフォームからのWebhookを受ける。認証トークンを検証し、取得ジョブをキューに投入する。

**取得ワーカー** 対象URLを取得し、WARCファイルを生成する。外部リソース（CSS、画像、JS）も全て含める。

**独立観察ノード** 東京、米国、欧州に分散配置された別のワーカー。同じURLを独立に取得し、ハッシュが一致することを確認する。

**ハッシュ計算・タイムスタンプ付与** WARCファイル全体のSHA-256を計算し、RFC 3161 TSAとOpenTimestampsの両方に送信する。

**ストレージ** WARCファイルを物理的に3拠点以上に分散保存する。S3互換ストレージを複数リージョン併用。

**証明書生成サービス** ユーザーの要求に応じて、PDF+JSON形式の証明書を発行する。

**参照API** 過去のアーカイブを日時指定で取得できるエンドポイント。

## 2.3 技術スタック推奨

- 言語：Python（取得ワーカー）、Go（APIゲートウェイ・高スループット部分）
- キュー：Redis + Celery、またはSQS
- データベース：PostgreSQL（メタデータ）、S3互換（WARCファイル本体）
- WARC処理：warcio（Pythonライブラリ、Internet Archive公式）
- ハッシュ：標準ライブラリ（hashlib）
- タイムスタンプ：rfc3161ng（Python）、opentimestamps-client
- ブロックチェーン：OpenTimestampsプロトコル経由でBitcoin

いずれも枯れた技術。冒険する必要はない。

---

## 3. 本人性の担保

### 3.1 原則

本人確認はプラットフォームに委譲する。アーカイブサービス側で独自の本人確認はしない。

### 3.2 OAuth 2.0フロー

1. ユーザーがプラットフォーム管理画面で「アーカイブ連携」をクリック
2. アーカイブサービスの認証画面にリダイレクト
3. アーカイブサービスがプラットフォームのOAuthエンドポイントにユーザーをリダイレクト
4. ユーザーがプラットフォームで認可
5. プラットフォームがアーカイブサービスに認可コードを返す
6. アーカイブサービスが認可コードをアクセストークンに交換
7. 以後、プラットフォームからのWebhookにこのトークンが付与される

### 3.3 トークン検証

Webhookを受けるたびに：

- トークンの署名を検証（プラットフォームの公開鍵で）
- 有効期限を確認
- スcopeを確認（このユーザー、このサイトに対する権限があるか）

### 3.4 プラットフォーム側が対応していない場合

WordPressのようなセルフホスト型CMSはOAuthプロバイダーにならない。この場合はプラグインとして提供する。

プラグインの動作：

1. WordPressの管理画面にインストールされる
2. ユーザーがアーカイブサービスのAPIキーを入力
3. `publish_post`、`post_updated`などのフックで起動
4. 公開されたコンテンツをアーカイブサービスに送信

この場合の本人性担保は「WordPressの管理者権限＝本人」という前提に依存する。

## 3.5 プラットフォーム署名

可能な場合、プラットフォーム側でコンテンツに署名を付けて送信する。プラットフォームの秘密鍵で署名し、公開鍵は公開する。

これにより「プラットフォーム自身がこのコンテンツの公開を認めた」という追加の証拠が付く。「存在の実在性」問題への対策になる。

## 4. コンテンツ保存

### 4.1 WARC形式の採用

ISO 28500国際標準。Internet Archive、各国国立図書館が採用。  
warcio (Python) で読み書きできる。1ファイルにHTTPリクエスト・レスポンスを複数格納可能。

### 4.2 取得対象

- HTML本体
- 外部CSS全て
- 外部JavaScript全て
- 画像 (img、背景画像、favicon)
- フォント
- 埋め込みiframe内容

- 動画・音声ファイル（サイズ上限あり、推奨50MB/ファイル）

取得しないもの：広告タグ、トラッキングスクリプト、外部APIのリアルタイムデータ（これらは再現性がないため除外）。

## 4.3 取得方法

Headless Chromeを使う。理由：JavaScriptレンダリング後のDOM状態を取得する必要があるため。Playwrightまたはpuppeteerを推奨。

手順：

1. ページをロード
2. `networkidle` 状態まで待機（全リソース取得完了）
3. ネットワークトラフィックを全て記録
4. レンダリング後のDOMも別途保存
5. スクリーンショットも撮る（PNG、参考用）
6. 全てをWARCにまとめる

## 4.4 メタデータ

WARCファイルに加えて、別途記録するメタデータ：

- 取得日時（UTC、ミリ秒精度）
- 対象URL
- HTTPステータスコード
- サーバーIPアドレス
- SSL証明書のフィンガープリント
- DNS解決結果
- User-Agent
- 取得ワーカーの識別子
- 独立観察ノードの結果

## 4.5 保存先

S3互換ストレージに3リージョン分散。推奨：東京、バージニア、フランクフルト。

命名規則： `{year}/{month}/{day}/{domain}/{uuid}.warc.gz`

---

## 5. 改ざん証明（実装詳細）

### 5.1 ハッシュ計算

WARCファイル生成完了後、gzip圧縮した最終ファイルに対してSHA-256を計算する。

python

```
import hashlib
def compute_hash(filepath):
    h = hashlib.sha256()
    with open(filepath, 'rb') as f:
        for chunk in iter(lambda: f.read(4096), b''):
            h.update(chunk)
    return h.hexdigest()
```

### 5.2 RFC 3161タイムスタンプ

RFC 3161は、WARCファイルのハッシュ値に対して認定タイムスタンプ局（TSA）から時刻付きの署名を取得する仕組みである。これにより、データが特定の時刻より前に存在していたことを信頼性高く証明できる。

しかし、**RFC 3161単独では取得時の改ざん耐性が限定的である**。取得ワーカーがロードした時点でJavaScript等により動的に



改ざんされた内容をハッシュ化した場合、TSAはその改ざんを検知できない。

**そのため、本サービスではRFC 3161を時刻の証明として用いつつ、取得時の真正性を強化するため、独立観察ノードによるクロス検証を必須とする。**

実装においては、以下の点を考慮する：

- 少なくとも2つの異なるTSA（セイコーソリューションズおよび国際的なTSA）に対して並行してタイムスタンプを要求する
- TSAへのリクエストは冗長性とタイムアウト処理を備えたラッパー関数で実装する
- 取得したタイムスタンプトークンは、WARCファイルと紐づけて安全に保存する
- 将来的には、証明書のローテーションやTSAの信頼性監視機構も追加する

## 5.3 OpenTimestamps

OpenTimestampsクライアントを使う。個別のハッシュではなく、一定期間（例：1時間）のハッシュをまとめてMerkle treeのルートをBitcoinに書き込む。

bash

```
ots stamp archive.warc.gz
```

生成される .ots ファイルを保存する。後でBitcoinブロックに確定されるまで数時間～数日かかる。確定後に `ots upgrade` で更新する。

## 5.4 独立観察ノードとの照合

3拠点が独立に取得したWARCのハッシュを比較する。**ハッシュが一致する部分**だけを「確定」として扱う。

動的コンテンツで完全一致しない場合：HTMLのmain要素のみを抽出し、広告などのノイズを除外した正規化版でハッシュを比較する。

## 5.5 証明書生成

ユーザーの要求時、またはアーカイブ作成時に自動生成する。

PDF版：人間可読。URL、日時、ハッシュ値、検証方法の説明を含む。スクリーンショットも添付。

JSON版：機械検証用。

json

```
{
  "version": "1.0",
  "url": "https://example.com/article",
  "archived_at": "2026-04-12T14:00:00.123Z",
  "warc_sha256": "a1b2c3...",
  "rfc3161_tokens": [
    {"tsa": "seiko", "token_base64": "..."},
    {"tsa": "digicert", "token_base64": "..."}
  ],
  "opentimestamps": {
    "ots_file_base64": "...",
    "bitcoin_block": 920314,
    "bitcoin_txid": "..."
  },
  "independent_observers": [
    {"region": "tokyo", "hash": "a1b2c3..."},
    {"region": "virginia", "hash": "a1b2c3..."},
    {"region": "frankfurt", "hash": "a1b2c3..."}
  ],
}
```

```
"platform_signature": {  
  "platform": "wordpress.com",  
  "signature": "...",  
  "public_key_url": "..."  
}  
}
```

---

## 6. 参照機能

### 6.1 永続URL設計

各アーカイブに永続IDを付与する。形式：

`https://archive.example.com/a/{base58_id}`

base58 ID は8～10文字。衝突耐性と短さのバランス。

### 6.2 DOI互換

学術引用のため、DOI互換のIDも発行する。Zenodoと同様の仕組み。研究者層に必須。

### 6.3 参照API

- `GET /api/v1/archives/{id}` - メタデータ取得
- `GET /api/v1/archives/{id}/warc` - WARCファイルダウンロード
- `GET /api/v1/archives/{id}/certificate.pdf` - PDF証明書
- `GET /api/v1/archives/{id}/certificate.json` - JSON証明書
- `GET /api/v1/archives/{id}/render` - ブラウザで閲覧可能な形式で表示
- `GET /api/v1/sites/{domain}/history` - ドメインの全アーカイブ履歴

## 6.4 閲覧時の表示

過去のアーカイブを表示する際、上部にバナーを表示する：「これは〇〇年〇月〇日時点のアーカイブです。改ざん証明済み。証明書を見る→」

pywb (Python Wayback) ライブラリが使える。Internet Archive が内部で使っているツール。

## 6.5 ジャンル別永続識別子

本サービスでは、ZenodoのDOIに加えて、コンテンツの性質に応じたジャンル別永続識別子を発行する。これにより、学术论文だけでなく、ブログ記事、音楽、図、アート、コード、アプリ、ゲーム、アイデア段階のコンセプトまで、統一された枠組みで永続的に識別・引用可能になる。

具体的なプレフィックスと分類については、今後の実装とユーザーからのフィードバックを基に確定していく予定である。現時点の例を以下に示す。

識別子体系

ジャンル	プレフィックス	例ID	主な用途
論文	vio.paper.	vio.paper.16torus-dynamic-20260415	Zenodo論文
テキスト	vio.text.	vio.text.archive-spec-20260412	仕様書、思考ノート、ブログ
アート / 図	vio.art.	vio.art.16torus-80nodes-v3	図、ダイアグラム
動画	vio.video.	vio.video.torus-explainer-01	解説動画

ジャンル	プレフィックス	例ID	主な用途
音楽	vio.music.	vio.music.light-dark-07	検証曲
コード / ツール	vio.code.	vio.code.torusknot-generator-v2	スクリプト・ツール
アプリ	vio.app.	vio.app.viorazu-dashboard-v1	Webアプリ
ゲーム	vio.game.	vio.game.torus-puzzle-01	ゲーム
アイデア	vio.idea.	vio.idea.inginburei-echo-202604	未成熟コンセプト
コンセプト	vio.concept.	vio.concept.16torus-dynamic-mapping	成熟した理論コンセプト

## 発行ルール

- アーカイブ保存時に、コンテンツの種類を自動判定し、適切なプレフィックスを提案する。
- 1つのコンテンツに対して、主要ジャンル1つを基本とし、必要に応じて複数の識別子を付与可能。
- すべての識別子は本サービスの永続URL  
(<https://archive.viorazu.com/vio.text.xxxx> など) で解決可能。
- 将来的には Handle System や ARK を用いたグローバルリゾルバへの対応も検討する。

この仕組みにより、「DOIだけではカバーしきれない多様な表現形態」を、すべて同じ強度の永続性と本人性担保のもとで管理できる。

---

## 7. 削除ポリシー

### 7.1 基本原則

確定後のアーカイブは運営者も削除できない。これがサービスの売り。

### 7.2 猶予期間

投稿から24時間は「未確定」状態。この間は所有者が取り消し可能。なりすまし・誤投稿対策。

猶予期間中の状態：

- WARCファイルは保存済み
- ハッシュ計算済み
- ただしRFC 3161とOpenTimestampsへの送信は保留
- 24時間後に自動的に確定処理

取り消された場合：

- WARCファイル本体は削除
- 「この時刻に取り消された投稿があった」という事実だけハッシュ記録に残す
- 「何が書かれていたか」は失われる

### 7.3 法的削除要請への対応

GDPR、日本の個人情報保護法、名誉毀損判決など、法的に削除が必要な場合がある。

対応：

- 本体データへのアクセスを遮断（tombstone化）
- ハッシュ記録は残す
- 「法的理由により閲覧不可」を表示

- 削除理由と日時を公開

完全削除はしない。「削除された事実」が残ることで改ざん証明のチェーンが切れない。

## 7.4 サービス終了時の保証

事業として存続できなくなった場合の備え：

- 全WARCファイルをInternet Archiveに寄贈する契約を事前に結ぶ
- 全証明書データを公開する
- ユーザーが自己保管できる形でエクスポート機能を常時提供

## 8. プラットフォーム連携

### 8.1 WordPress（プラグイン方式）

最優先実装。世界のウェブの40%以上がWordPress。

プラグインの構造：

- 設定画面：APIキー入力、プラン表示、アーカイブ履歴一覧
- フック： `publish_post`、`post_updated`、`publish_page`
- 送信処理：記事公開時にWordPress REST APIから本文取得、アーカイブサービスに送信

配布：WordPress.org公式プラグインディレクトリに登録。

### 8.2 OAuth対応プラットフォーム

note、Medium、Ghost、Substack。いずれもOAuth 2.0とWebhook APIを持つ。

実装手順：

1. 各プラットフォームの開発者ポータルでアプリ登録

2. OAuthエンドポイント実装
3. Webhookレシーバー実装（`post.published`、`post.updated` イベント）
4. 受信コンテンツを取得キューに投入

## 8.3 サイトビルダー系

Wix、Squarespace、Webflow。App Market／Extension経由で提供する。

Wixの場合：Wix App Marketに公開。ユーザーはダッシュボードから1クリックでインストール。Wix Velo（旧Corvid）のWebhook機能で公開イベントを受け取る。

## 8.4 ドメイン監視方式（フォールバック）

連携できないプラットフォームの場合：RSS/Atomフィードを定期監視する。

- 登録時にRSS URLとドメイン所有証明（DNS TXTレコード）を要求
- 15分ごとにフィードをチェック
- 新規エントリを検出したらアーカイブ取得

本人性の担保は弱くなる（RSSは誰でも読める）ため、DNS所有証明を必須にする。

---

# 9. API仕様

## 9.1 認証

Bearer token方式。ヘッダー： `Authorization: Bearer {api_key}`



APIキーはユーザーごとに発行。プラットフォーム連携の場合はOAuthアクセストークンを使う。

## 9.2 エンドポイント一覧

POST	/api/v1/archives	新規アーカイブ作成
GET	/api/v1/archives/{id}	アーカイブ取得
GET	/api/v1/archives/{id}/warc	WARCファイル取得
GET	/api/v1/archives/{id}/certificate (?format=pdf json)	証明書取得
GET	/api/v1/archives/{id}/render	レンダリング表示
DELETE	/api/v1/archives/{id}	取り消し (猶予期間内のみ)
GET	/api/v1/sites	自分のサイト一覧
POST	/api/v1/sites	サイト登録
GET	/api/v1/sites/{domain}/archives	サイトのアーカイブ履歴
GET	/api/v1/verify/{id}	証明書検証 (公開エンドポイント)
POST	/api/v1/webhooks/platforms/{name}	プラットフォームWebhook受信
GET	/api/v1/oauth/{platform}/callback	OAuthコールバック
GET	/api/v1/billing/plans	プラン一覧
POST	/api/v1/billing/subscribe	サブスクリプション開始

## 9.3 新規アーカイブ作成リクエスト例

json

```
POST /api/v1/archives
{
  "url": "https://example.com/article",
  "platform": "wordpress",
  "platform_signature": "...",
  "content_hint": {
    "title": "...",
    "published_at": "2026-04-12T14:00:00Z"
  }
}
```

レスポンス：

json

```
{
  "id": "aB3cD4eF",
  "status": "pending",
  "url": "https://archive.example.com/a/aB3cD4eF",
  "confirmed_at": null,
  "expires_from_cancel_at": "2026-04-13T14:00:00Z"
}
```

## 9.4 検証エンドポイント

第三者が証明書を検証するための公開エンドポイント。認証不要。

```
GET /api/v1/verify/{id}
```

レスポンス：検証結果と、検証に使った全要素（TSAトークン、Bitcoinトランザクション参照、独立観察ノードのハッシュ）を返す。

---

## 10. データベース設計

### 10.1 主要テーブル

#### **users**

- id, email, created\_at, plan\_id, stripe\_customer\_id

#### **sites**

- id, user\_id, domain, platform, platform\_site\_id, oauth\_token\_encrypted, created\_at, verified\_at

#### **archives**

- id (base58), site\_id, url, status (pending/confirmed/tombstone), created\_at, confirmed\_at, cancelled\_at
- warc\_path, warc\_sha256, warc\_size\_bytes
- title, http\_status, server\_ip, ssl\_fingerprint

#### **timestamps**

- id, archive\_id, type (rfc3161/opentimestamps), tsa\_name, token\_data, bitcoin\_block, bitcoin\_txid, created\_at

#### **observations**

- id, archive\_id, region, hash, agreed (boolean), observed\_at

#### **certificates**

- id, archive\_id, format (pdf/json), generated\_at, download\_count

### platform\_signatures

- id, archive\_id, platform, public\_key\_id, signature

## 10.2 インデックス

- `archives(site_id, created_at DESC)` - 履歴表示用
- `archives(url, created_at DESC)` - URL別履歴用
- `archives(status, created_at)` - 猶予期間処理用バッチ
- `sites(domain)` - ドメイン検索用

## 10.3 WARCファイル本体

PostgreSQLには入れない。S3互換ストレージに保存し、パスだけDBに記録する。

---

# 11. 段階的実装計画

## Phase 1：MVP（3ヶ月）

目的：動く最小構成を作る。

- WordPressプラグインのみ対応
- 取得ワーカー1拠点のみ（独立観察なし）
- RFC 3161のみ（OpenTimestamps後回し）
- 証明書はPDFのみ
- ストレージは単一リージョン

この時点で「動くプロダクト」として1人目のユーザー（Viorazu.さん自身）が使える状態を作る。ドッグフーディング。

## Phase 2：信頼性強化（3ヶ月）

- 独立観察ノード3拠点配置
- OpenTimestamps追加
- ストレージ3リージョン分散
- JSON証明書と公開検証エンドポイント
- 研究者向けDOI発行機能

ここで「技術的に証拠能力のあるサービス」として完成する。

## Phase 3：プラットフォーム拡大（6ヶ月）

- note連携（OAuth）
- Medium連携
- Ghost、Substack連携
- Wix App Market出品

ここで「一般ユーザーが使える」範囲に広がる。

## Phase 4：企業対応（6ヶ月）

- エンタープライズプラン
- 法的削除要請対応フロー整備
- 監査ログ、SLA
- 認定TSA契約の本格化（セイコーソリューションズ）

ここで「企業が発言の証拠として使える」水準に達する。

## Phase 5：永続性保証（継続）

- Internet Archiveとの寄贈契約
  - 物理メディアへの定期書き出し
  - 暗号アジリティ対応（SHA-3追加など）
  - 複数ブロックチェーン対応
-

## 12. コンテンツ真正性の証明範囲

本サービスは、AI企業が直面するデータプロベナンスの課題に対しても有効な有効な解決策となる。

特に、以下の3つの概念を中心に、AI時代におけるコンテンツ真正性の証明を提供する。

- **「この出力はこの時点でこのモデルが生成した」** AI生成コンテンツの出典証明（AI Generation Provenance）
- **「このデータは○年○月○日時点でこの人が公開した」** 人間公開コンテンツの取得時点真正性（Capture-time Authenticity）
- **人間署名によるAI生成疑惑への反証** 人間（またはプラットフォーム）によるデジタル署名で「AI後付けではない」と主張する証明

本サービスは、**AI企業には出力の真正性証明を、クリエイターには人間起源の証明を、同一インフラで提供することを目指している。**

WARC形式による完全保存、独立観察ノード（3拠点）によるクロス検証、RFC 3161+OpenTimestampsによる多層タイムスタンプ、任意の人間/プラットフォーム署名を組み合わせることで、従来のC2PAではカバーしにくい「取得時点の真正性」と「人間起源の反証」を強力にサポートする。

これにより、AI企業は生成コンテンツの透明性と監査可能性を向上させ、クリエイターは自身の創造物がAI生成疑惑から守られる環境を実現できる。将来的には、AI企業向け専用プランやC2PAとの連携も視野に入れている。

**AIコンテンツ真正性に関する主要概念と本サービスの対応**

No.	概念 (日本語)	英語での標準的な呼び方	内容の説明	本サービスとの相性
1	AI生成出力の出典証明	AI Generation Provenance / Model Attribution	どのモデル・バージョン・プロンプトで生成されたか	◎（生成直後にアーカイブ＋署名で記録可能）
2	人間公開コンテンツの取得時点真正性	Capture-time Authenticity / Proof of Existence at Publication	公開された瞬間の正確な状態を証明	◎（独立観察ノードが特に強い）
3	人間署名によるAI疑惑反証	Human Signature / Cryptographic Attestation	人間（またはプラットフォーム）が署名した記録で「AI後付ではない」と主張	◎（プラットフォーム署名やユーザー署名で強化）
4	編集履歴・改変履歴の完全追跡	Edit History / Provenance Chain	作成後に行われたすべての編集・変換を記録	○（WARCでページ全体を保存すれば一部可能だが、細かい編集追跡は弱め）
5	完全な来歴・ライフ	Full Lifecycle Provenance	作成 → 編集 → 公開 → 配	△（取得時点に強く、配布

No.	概念 (日本語)	英語での標準的な呼び方	内容の説明	本サービスとの相性
	サイクル		布 → 再利用までの全履歴	後の追跡は弱い)
6	整合性 (改ざん検知)	Integrity / Tamper Detection	作成後から現在まで一切改ざんされていないこと	◎ (ハッシュ + タイムスタンプ + 独立観察で非常に強い)
7	作者・所有者 アイデンティティ	Verified Creator Identity / Attribution	誰 (どの組織・人間) が責任を持って作成・公開したか	○ (人間署名やプラットフォーム署名で証明可能)
8	意図・文脈の証明	Intent / Context Provenance	「なぜ作ったか」 「どのような目的で公開したか」というメタ情報	△ (任意でメタデータを追加可能だが、強制は難しい)

この表により、本サービスが特に強い領域（取得時点の真正性、整合性、人間署名による反証）と、相対的に弱い領域（ライフサイクル全体の追跡、意図の証明）を明確に示すことができる。

将来的には、弱い領域についてもC2PA等との連携により補完していく方針である。



## 13. 開発体制の推奨

最小構成：

- バックエンドエンジニア 1～2名（Python/Go）
- インフラエンジニア 1名（S3、Kubernetes）
- フロントエンド 1名（管理画面、証明書表示）
- WordPressプラグイン開発者 1名（兼任可）

Phase 1なら2～3名でも可能。Phase 2以降で独立観察ノードの運用が入るため、インフラ専任が必要になる。

---

## 14. 参考資料

- ISO 28500（WARC形式仕様）
  - RFC 3161（Internet X.509 Public Key Infrastructure Time-Stamp Protocol）
  - OpenTimestamps: <https://opentimestamps.org/>
  - warcio: <https://github.com/webrecorder/warcio>
  - pywb: <https://github.com/webrecorder/pywb>
  - Internet Archive: <https://archive.org/>
  - 電子帳簿保存法（国税庁）
  - eIDAS規則（EU）
- 

## License and Additional Terms

This paper is licensed under CC BY 4.0 (Creative Commons Attribution 4.0 International).

Additional terms by the author (Viorazu.):

- Appropriate attribution is required when reusing or modifying this work.
- The author encourages the use of this paper for AI research and training, provided clear attribution is given.
- Commercial use or large-scale integration into AI products requires prior contact with the author.

Full license details are available in the accompanying technical specification document.

---

## 著者情報

**Viorazu.**

「時の神 三つの社に 印置き 後の世人の 疑ひを解け」  
(ときのかみ みつのやしろに しるしおき のちのよびとの うた  
がいをとけ)

- ORCID: 0009-0002-6876-9732
- GitHub: <https://github.com/Viorazu/Viorazu-ConnectHub>
- SHA256:2cb6606e6dbd84ff8e12d1ca7c91898aa1cf9abc49662860288c9b0d84bbe92e
- License: CC BY 4.0 (Creative Commons Attribution 4.0 International)
- Co-written by Viorazu. and Claude (Claude 4 series, Anthropic)
- Peer-reviewed and revised by Grok (xAI)
- Publication Date: 2026-04-15
- Version: 1.0