AIエージェント同士が、企業間で商取引を完結させる

― Verifiable Credentials による組織横断の相互認証：設計と実証 ―

> 記載の実装状況は **2026-08-28 時点**の実装を実測した値に基づきます。

共同研究体制

役割	担当
買い手側エージェント（Client Agent / Shopping Agent）実装	GMOグローバルサイン・ホールディングス株式会社
売り手側エージェント（Remote Agent / Merchant）実装	株式会社VESS Labs
仕様設計	GMOグローバルサイン・ホールディングス株式会社 / 株式会社VESS Labs

エグゼクティブサマリー

AIエージェントが、人間の操作を介さずに他社のエージェントへ商品を発注する――この近未来の取引において、「相手のエージェントは本当にその企業のものか」「その発注は本当に人間の承認に基づくものか」を、「双方が相手の認証システムに問い合わせることなく、暗号技術だけで検証する仕組み」を、2社が共同で設計・実装した。

組織をまたいだ権限の受け渡しは、業界標準の側でも整備が進んでいる。ただしそれらは**取引する2社が事前に相対で合意している**ことを前提とする。相手が事前に決まらないエージェント取引では、この前提が成り立たない。ここで有効なのが **Verifiable Credentials（VC、検証可能な証明書）** である。VC は、受け取った側が**発行元に問い合わせることなく、その場で真正性を検証できる**。共通の信頼の起点さえ共有されていれば、**個別の事前合意なしに、初めて出会う相手を検証できる**――この性質が決定的に効く。

本実証の結果として、売り手側エージェントをクラウド上で実稼働させたうえで、**2社が独立に実装した買い手・売り手のエージェントを直結し、相互認証から取引完了までの一連のシナリオがすべて成功することを確認**した。1社が両側を作るのではなく**独立に実装したものを突き合わせて疎通させた**点に、共同研究としての価値がある。

価値は3点に整理できる。

- 第一に、**取引のたびに人手で相手を確認する作業が要らなくなる**：エージェント同士が自動で商取引を完結でき、取引にかかる人的コストが原理的にゼロに近づく。
- 第二に、**「誰が誰に、何を委任したか」が署名付きの証跡として残る**：自動取引であっても、第三者が後から暗号的に検証できる形で責任の所在を追跡できる。
- 第三に、**Allowlist（手動の許可リスト）に頼らずに信頼を確立できる**：これはエージェントの数が増えるほど効いてくる。本人確認・組織の実在性確認・権限確認はあらゆる商取引に共通して必要となる工程であり、応用範囲は特定業種に限定されない。

1. 背景と問題設定

1.1 すでに起きている実害

AIエージェントが外部ツールを呼び出し、他のエージェントにタスクを委任し、自律的に商取引・決済を実行する利用形態が現実味を帯びている。エージェント間通信の標準化も進み、`MCP`（Agent to Tool/Context）、`A2A`（Agent to Agent）、`UCP` / `AP2`（Agent to Commerce / Payments）といったプロトコルが登場している。

ここで「取引の相手を信頼してよいか」という問いが生まれる。これは将来の理論的懸念ではなく、**すでに実害が生じている領域**である。

- **なりすまし** — 正規の取引先を騙るエージェントに対して、機密情報や与信を渡してしまうリスク。相手が名乗る所属を裏付ける手段がなければ、実質的に自己申告を信じるほかない。
- **誤発注・暴走発注** — エージェントの判断ミスや不正な指示により、ユーザーが承認していない発注が実行される。従来型の仕組みでは、その発注に人間の承認があったかどうかを**売り手側が確かめる術がない**。
- **責任の所在が追えない** — 問題が起きた後、「誰の指示で、どの権限に基づいて実行されたのか」を事後に証明できない。自動化の規模が拡大するほど、この不透明さは重大なリスクになる。

現状これらは、**取引相手を人手でリスト管理する（Allowlist）** ことで回避されている。相手が数社なら回るが、エージェント経済圏の規模では破綻する。

1.2 既存アプローチとの関係

「組織をまたいだ認可」は本研究に固有の課題ではなく、標準化コミュニティでも活発に取り組まれている。**既存アプローチで解けること・解けないこと**を整理し、本研究の位置づけを明確にする。

OAuth は組織をまたげるが、成立条件がある。 OAuth 2.0 Token Exchange（**RFC 8693**）を組織横断に拡張する `draft-ietf-oauth-identity-chaining` は、2026年9月時点で revision 17 が IESG 評価を終えて **RFC Editor Queue** に入っており、RFC 発行を待つ最終段階にある [1]。ただし同ドラフトは domain B の Auth Server が domain A の公開鍵を信頼していることを前提とする。仕様はこの信頼関係が「典型的には鍵素材の交換として現れる」と述べ、domain A の `jwks_uri` に依拠する形態も挙げるが、いずれ

も **domain B 側が domain A を名指しで信頼対象として設定する**点は変わらない。つまり **Auth Server 間の二者間（pairwise）の事前合意**が信頼の基礎にある。

相手が既知で数が限られる企業間連携では自然な運用だが、**取引相手が事前に定まらないエージェント経済圏では、参加者数に対して合意コストが組み合わせ的に増える**（10社なら45通り、100社なら約5,000通り）。対して VC + PKI 方式では、**各参加者が「共通の信頼の起点」を1つ持てばよい**。パスポートと同じく、渡航先ごとに個別交渉せずとも共通の枠組みへの信頼で初対面の相手を検証できる。**エージェントが増えるほど、この差が効いてくる。**

なお OpenID Federation（1.0: 2026-02-17、1.1: 2026-05-06 に Final 化 [2]）も Trust Anchor 起点で個別合意なしの相互信頼を実現する。本研究の X.509 チェーン検証と発想を同じくするアプローチであり、大規模な信頼確立には Trust Anchor 型が要るという結論に両者が独立に到達している点は、本研究の設計判断を裏づける。

エージェント関連標準も、事前交渉の排除に向かっている。 A2A v1.0 は **Signed Agent Card**（`signatures`、JWS）を §8.4 に規定し（署名は MAY、すなわち任意）[3]、**UCP** は `/.well-known/ucp` への公開鍵掲示 [4]、**MCP** は HTTPS URL 自体を `client_id` とする OAuth Client ID Metadata Document への対応を SHOULD として推奨する（事前登録・動的クライアント登録も併記されている）[5]。ただしこれらはすべてドメイン所有（DNS / TLS）を信頼の起点とする点で共通し、ここが分岐点になる。ドメインに紐づく検証が示すのは provenance（この主張は当該ドメインの管理者に由来する）であって trust（その主体が信頼に足る）ではない。ドメインを取得できれば誰でも鍵を掲示でき、「実在する企業か」の判断は別途必要になる。

	ドメイン所有型（Signed Agent Card / UCP / MCP）	Agent Identity VC（本研究）
証明する内容	ドメイン所有者が署名した（自己署名）	Issuer（審査を担う認証局）がエージェントの実在と capability を審査・認定した（設計上の想定）
信頼の起点	ドメイン所有（DNS / TLS）	X.509 チェーン → 共有 Root of Trust
選択的開示 / 鍵所持証明 / 失効	いずれも当該仕様の範囲外	いずれも有り（項目単位の開示制御 / KB-JWT による PoP / Token Status List）

> 表の2行目は、Agent Identity VC という**方式が証明できる命題**を示したものであり、本 PoC で実際に Issuer が Agent 審査を行ったという意味ではない。**本 PoC における Issuer による本格的な Agent 審査は対象外**である（3.1 で後述）。 表の3行目は、これらの仕様に当該機能が原理的に実現できないという意味ではなく、**クレーム単位の選択的開示・提示ごとの Holder Binding・標準化された失効リストのいずれもが、当該仕様自身では規定されていない**（別途の仕組みを要する）という意味である。

両者は競合ではなく、証明できる命題の層が異なる。 実際、本 PoC の買い手・売り手いずれの Agent Card も、A2A 仕様上は `signatures`（Signed Agent Card）を持つことができ、VC を `metadata` に同梱

する構成と排他ではなく併用可能である。ただし本 PoC では `signatures` フィールドに実際の値は設定しておらず（空のまま）、VC 側のみを用いている。

したがって本研究が対象とする領域は、次の2点に絞られる。

- **Issuer の審査を伴う信頼を、事前の相対合意なしに組織間で成立させること** — Trust Anchor 型はこの性質を持つが、**エージェント間通信プロトコル上での実装方式は確立していない**。ドメイン所有型は事前合意を不要にするが、provenance の層にとどまる。
- **ユーザー意思の暗号的証明と、その業務データへのバインド** — 「この取引は人間が承認した」ことを取引内容と不可分な形で証明し、相手が独立に検証できる仕組み。AP2 の Mandate がこれに相当する。AP2 は 2026-04 に FIDO Alliance へ寄贈され、Agentic Authentication TWG と Payments TWG の双方で標準化が進んでいるが [8]、VC 基盤の上でどう組み立てるかについて公開された実装知見は限られる。

同種の課題意識は複数のコミュニティで並行して扱われている。DIF の Trusted AI Agents WG は **KYA-OS** を策定中であり、DID / VC を基礎に「事前の調整を要しない組織横断の検証」を掲げ、MCP 向けのリファレンス実装を公開している [6]。W3C Agent Identity Registry Protocol CG は 2026-04-24 に発足したが、現時点で仕様草案の公開には至っていない [7]。**本研究の固有性は、買い手・売り手を別々の組織がそれぞれ独立に実装し、仕様のみを共有した状態で相互接続を成立させた点にある**（2章3点目）。

> **参照**（URL 確認日: 2026-09-11） [1] IETF OAuth WG, *OAuth Identity and Authorization Chaining Across Domains*, `draft-ietf-oauth-identity-chaining-17`（2026-07-19 / RFC Editor Queue） https://datatracker.ietf.org/doc/draft-ietf-oauth-identity-chaining/ [2] OpenID Foundation, *OpenID Federation*（1.0 Final / 1.1 Final: 2026-05-06 承認） https://openid.net/openid-federation-1-1-final-specifications-approved/ [3] *A2A Protocol Specification v1.0* §8.4（Agent Card Signing）, Agentic AI Foundation（Linux Foundation） https://a2a-protocol.org/v1.0.0/specification/ [4] *Universal Commerce Protocol Specification* — Signatures, UCP https://ucp.dev/specification/signatures/ [5] Model Context Protocol, *Authorization*（revision 2025-11-25）, Agentic AI Foundation https://modelcontextprotocol.io/specification/2025-11-25/basic/authorization [6] DIF Trusted AI Agents Working Group（KYA-OS） https://identity.foundation/working-groups/trusted-agents.html [7] W3C Agent Identity Registry Protocol Community Group（2026-04-24 発足） https://www.w3.org/community/agent-identity/ [8] FIDO Alliance, *FIDO Alliance to Develop Standards for Trusted AI Agent Interactions*（2026-04-28） https://fidoalliance.org/fido-alliance-to-develop-standards-for-trusted-ai-agent-interactions/

1.3 VC を適用すべき領域の特定

事前分析として、エージェント連携の3種別における VC の適用可否を整理した。

#	種別	VC の要否	理由
(1)	**A2A のクロスドメイン（組織間）通信**	**要**	取引相手が事前に定まらないため、双方の Authorization Server における事前の pairwise（二者間）の合意を前提にできない
(2)	**UCP / AP2 の決済フロー**	**要**	ユーザー承認（Mandate）の暗号的証明が必要。決済は否認防止が要件になる
(3)	**MCP / 同一ドメイン内の A2A**	**不要**	共通の管理主体が存在するため、既存の OAuth 2.1 / OIDC で十分

結論として **VC が意味を持つのは (1) と (2)** であり、本研究はこの2領域に焦点を当てた。

2. 研究の目的と体制

Authorization Server の信頼境界を越えた、AIエージェント間の**双方向の暗号的相互認証**と、その上での**ユーザー意思を暗号的に証明した商取引・決済フロー**を、実装可能な形で設計・検証すること。具体的には次を目的とした。

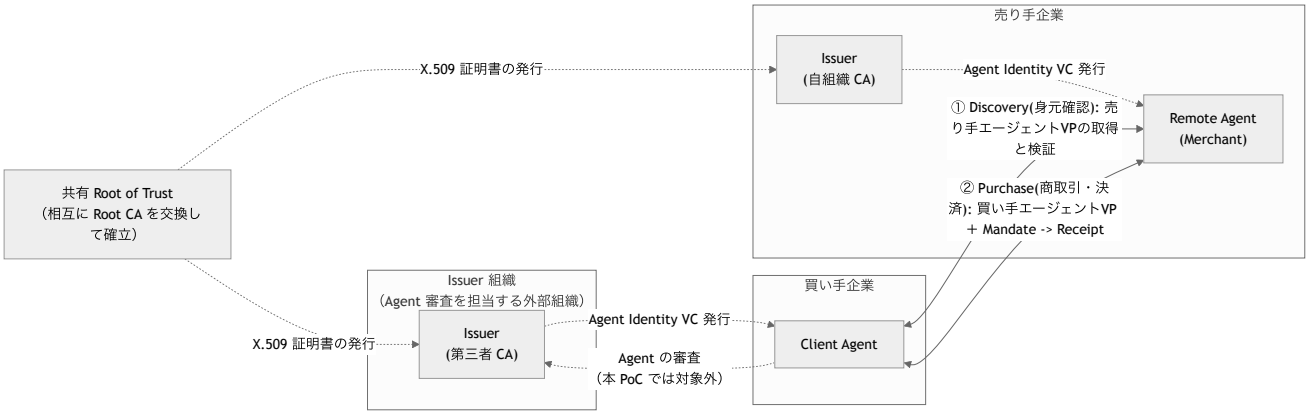
1. A2A プロトコル上で、VC による双方向 VP 提示・検証の交換方式を設計する。
2. AP2 v0.2 の Checkout / Payment Mandate によって、ユーザー承認を暗号的に証明する。
3. 買い手・売り手を**別々の組織がそれぞれ独立に実装**し、相互接続によって設計の妥当性を検証する。
4. 方式設計として判断を要する未解決の論点を明らかにする。

本研究の特徴は3点目にある。1社が両側を作ると、設計上の前提が暗黙のうちに両側で共有されてしまい、方式として成立しているかを確かめられない。**あえて分担することで、仕様のみを共有した状態で相互接続が成立するかを検証できる。**

3. 設計と実装

3.1 全体像

B2B における AI エージェント間の商品購買を仮想ユースケースとした。買い手のエージェントが売り手のエージェントに商品を発注する。両者はまず互いの「デジタル身分証（Agent Identity VC）」を提示し合い、相手が信頼できる組織のエージェントかを確認する。確認が取れて初めて、商品の検索・注文内容の確定・決済へと進む。全体像を図示すると次の通りである。



Issuer は、審査を担う組織が Root of Trust に連なる CA を運用するという構造を取る（売り手は自組織 CA、買い手は外部の Issuer 組織を想定）。相手の Issuer に問い合わせることなく検証が成立し、両者を繋いでいるのは共有 Root of Trust のみで、共通の Authorization Server は存在しない。これが本方式の要点である。**なお本 PoC では、Issuer による本格的な Agent 審査は対象外である**。

また上図のとおり、買い手エージェントは常に A2A の `Client Agent`、売り手エージェントは常に `Remote Agent` であり、この役割は固定されている（P2P ではなく Client-Server 型）。したがって、Discovery から決済完了まで一貫して**アクセスを起点とするのは買い手側のみ**であり、**売り手側が買い手の Agent Card およびA2AのEndpointにアクセスすることはない**。買い手の身元は、買い手自身が Purchase リクエストの Part（`data` フィールド）に VP を同梱して能動的に提示する形で伝わる（3.3 参照）。

この全体像を実現する各レイヤーの採用技術と、その仕様のステータスを整理すると次の通りである。

レイヤー	採用技術	仕様のステータス（2026-08 時点）
エージェント間通信	A2A Protocol v1.0.1 (2026-05-28)	Linux Foundation 管理
コマース / 決済	AP2 v0.2（2026-04-28）	FIDO Alliance へ移管（2026-04-28）
VC の選択的開示	SD-JWT	**RFC 9901**（2025-11 発行、Standards Track）
VC フォーマット	SD-JWT VC（`dc+sd-jwt`）+ KB-JWT（`kb+jwt`）	`draft-ietf-oauth-sd-jwt-vc-18`（IESG 提出済み）
失効	Token Status List（`status.status_list` claim 経由）	`draft-ietf-oauth-status-list-21`（IESG 提出済み）
信頼確立	X.509 PKI	—
署名	ES256 / ES384 / ES512、`alg:none` 禁止	—

基盤となる SD-JWT は **RFC 9901** として発行済みであり、その上に載る SD-JWT VC・Token Status List はいずれも IESG 提出段階にある。**本研究が依拠する暗号レイヤは、実験的な仕様ではなく標準化の最終段階にあるもの**である。また、OID4VP / OID4VCI は**やり取りの方式（プロトコルフロー）としては不採用**とし、その**データモデル（SD-JWT VC + KB-JWT）のみを流用**した。

3.2 Agent Identity VC の設計

Agent Card の `capabilities.extensions[]` 配列に、本 PoC 独自仕様の拡張 URI を宣言している。買い手・売り手いずれの Agent Card にも `https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1`（Agent Identity VC の導入）を宣言している。またこの拡張 URI 固有のプロパティとして `params`（`roles` / `vc_formats` / `issuer.domain`）を持つ。

なお本拡張とは別に、AP2 対応を示す拡張 URI として独自の `https://github.com/vesslabs-gmogshd/ap2/tree/v0.2` を用いている。これは AP2 の discovery 方式が v0.1 から v0.2 で変更されたことに起因する暫定対応であり、詳細は 6.2 で述べる。

実際の買い手 Agent Card から `capabilities.extensions` のみを抜粋すると次のとおりである。


```
{
  "capabilities": {
    "extensions": [
      {
        "uri": "https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1",
        "description": "This agent presents an organization-issued Agent Identity VC for cross-domain mutual aut
        "required": false,
        "params": {
          "roles": ["buyer"],
          "vc_formats": ["dc+sd-jwt"],
          "issuer": { "domain": "issuer.agent-sign.example.com" }
        }
      },
      {
        "uri": "https://github.com/vesslabs-gmogshd/ap2/tree/v0.2",
        "description": "AP2 shopping agent",
        "required": false
      }
    ]
  }
}
```


`required: false` としているのは、本拡張を解さない相手にも Agent Card 自体は通常どおり取得・表示できるようにするためである。**Agent Card を丸ごと VC でラップする構造を採っているため、デフォルトの A2A Agent Card 仕様はそのまま維持され**、この独自拡張をサポートするエージェントだけが Agent Identity VC による相互認証に参加できるという後方互換な設計になっている。なお、この拡張 URI は 3.3 で述べる `A2A-Extensions` ヘッダで有効化する際に指定する値と同一である。

Agent Identity VC は、この拡張に対応する VC であり、A2A の Agent Card を VC でラップしたものである。Agent Card 単体では自己申告に過ぎないが、VC 化により Issuer が内容を審査・認定したことを暗号的に証明できる。

VC は SD-JWT VC 標準 claim（`iss` / `sub` / `iat` / `exp` / `vct` / `cnf.jwk` / `status.status_list`）に加え、Agent Card の内容を単一の `agentCard` claim（`name` / `skills` / `capabilities` / `provider` / `supportedInterfaces` 等を含むオブジェクト）として格納する。`cnf.jwk` には Agent 自身の公開鍵を格納しており、これが 3.5 の群 C（KB-JWT による鍵所持証明）の検証対象になる。`sub` には Agent Card URL を採用しており（did:jwk は不採用）、検証側は取得した Agent Card の URL と VC の `sub` を文字列として直接突き合わせられるため、DID resolver 等の追加処理を挟まずに VC と Agent の紐付けを検証できる。

Agent Identity VC は「誰であるか（身元）」と「何ができるか（能力）」を証明するものであり、個別取引の認可とは役割が異なる。「今回の取引で何を購入してよいか」は Checkout Mandate が、「その代金をどの支払い手段でいくら支払ってよいか」は Payment Mandate が、それぞれ取引ごとに発行されて担う。`skills` はあくまで「能力」であり、それ自体が特定の取引を許可する「認可」を意味するものではない。

3.3 A2A における通信データ構造

VP の交換は A2A の `extensions` 機構（`metadata` / Part の `data` フィールド）で行う。AP2 のリファレンス実装が同型パターンを採るため、業界整合性を重視して選択した。

- **拡張の宣言と有効化**: 3.2 で述べた通り Agent Card の `capabilities.extensions` に拡張 URI（`agent-identity-vc/v0.1` / `ap2/tree/v0.2`）を宣言しておき、リクエスト側は `A2A-Extensions` ヘッダにカンマ区切りで有効化する拡張 URI を指定する。
- **売り手 VP の運び方**: Discovery 時点では Agent Card の `metadata` に同梱して公開配信する（3.4 の非対称設計に対応）。
- **買い手 VP の運び方**: 購買リクエストの Part（`data` フィールド）に同梱する。この `data` フィールドは拡張 URI をキーにした構造で、`<URI>/vp` に VP 本体（`vp_token`）と `seller_nonce` の echo を格納する。
- **action の運び方**: `catalog.search` / `cart.create` / `checkout` の action 名は `<URI>/action` キーに格納し、action ごとの業務データ（検索キーワード等）は同じ `data` フィールドの別キーに同居させる。この action 名・キー構造は本 PoC 独自仕様であり、3.6.1 で述べる UCP 非準拠の範囲と対応する。
- **Mandate等actionに対応するデータの運び方**: 同様に Part の `data` フィールドに同梱して運ぶ。

実際に買い手が売り手から取得する Agent Card から、標準的な構造を維持しつつ本拡張に関連する主要な部分（拡張の宣言と `metadata` による VP 同梱）のみを抜粋すると次のとおりである。

```
{
  "name": "Partner Sales Agent",
  "provider": { "organization": "Partner Inc.", "url": "https://merchant-poc.vegent.ai" },
  "skills": [
    "... (catalog.search, cart.create, checkout 等の業務スキル定義。省略) ..."
  ],
```

```
  "capabilities": {
    "extensions": [
      {
        "uri": "https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1",
        "required": false,
        "params": {
          "roles": ["seller"],
          "vc_formats": ["dc+sd-jwt"]
        }
      },
      {
        "uri": "https://github.com/vesslabs-gmogshd/ap2/tree/v0.2",
        "required": false
      }
    ]
  },
  "metadata": {
    "https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1/vp": {
      "vp_token": "eyJ0eXAiOiJkYytzZC1qd3QiLCJhbGciOiJFUzI1NiIsIng1YyI6Wy4uLl19...(省略)...",
      "seller_nonce": "zyycXA4VX0t4B2Ot2nGv75mVNG28rWa25zl-TuDEHmk"
    }
  }
}
```

次に実際に買い手から売り手へ送信される `catalog.search` リクエストは次のとおりである（VP 本体は長大なため省略）。

```
POST /a2a HTTP/1.1
Content-Type: application/json
A2A-Extensions: https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1, https://github.com/vesslabs-gmogshd/

{
  "jsonrpc": "2.0",
  "id": "d8b6af53-fdff-4d41-9efb-4b42feae6e67",
  "method": "SendMessage",
  "params": {
    "message": {
      "messageId": "b1a2c3d4-...",
      "role": "ROLE_USER",
      "parts": [
        {
          "data": {
            "https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1/vp": {
              "vp_token": "eyJ0eXAiOiJkYytzZC1qd3QiLCJhbGciOiJFUzI1NiIsIng1YyI6Wy4uLl19...(省略)...~eyJ0eXAiOiJrY
              "seller_nonce_echo": "zyycXA4VX0t4B2Ot2nGv75mVNG28rWa25zl-TuDEHmk"
            },
            "https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1/action": "catalog.search",
            "query": "Standard Widget"
```

```
          }
        }
      ]
    }
  }
}
```

`A2A-Extensions` ヘッダで宣言した拡張 URI と、Part の `data` オブジェクトのキーに使われている URI が同一である点がこの構造の要点である。`<URI>/vp` に VP 本体と nonce echo、`<URI>/action` に action 名を格納し、`query` のような action 固有の業務データは同じ `data` オブジェクトの直下に同居させる。Mandate を運ぶ `checkout` リクエストでも同様に、`"ap2.mandates.CheckoutMandate.closed"` 等のキーが同じ `data` オブジェクトに追加される（3.6.3 参照）。

3.4 信頼確立の流れとVP 交換方式の選択

信頼確立は2段階で行う。互いの身分証を確認し、確認が取れてから取引に進む。なお、以下の Phase 1 で買い手がアクセスする売り手の Agent Card URL は、買い手側が保持する静的な Registry（discovery 専用であり、trust の判定は行わない）から取得する。Registry の位置づけの詳細は 6.1 ③ で述べる。

この2往復構成は、レイテンシと安全性のトレードオフとして複数案を比較したうえで選択した。**1往復案**（買い手が先に VP を送る）はレイテンシ最小だが、**買い手が売り手を未検証のまま自らの VC を晒す**ため、偽の売り手を立てるだけで買い手の VC を収集できてしまう。**3往復案**（VP 提示を独立させる）は安全だが専用の往復が挟まる。採用した2往復ハイブリッドは、売り手を先に検証したうえで買い手 VP を購買リクエストと**一括送信**するため、VP 提示専用の往復を設けず、実質的なオーバーヘッドは Agent Card 取得の1回のみに収まる。

上記の Phase 1（Discovery）・Phase 2（Purchase）それぞれのエンドポイントと内容を整理すると次の通りである。

Phase	エンドポイント	内容
Phase 1: Discovery	`GET /.well-known/agent-card.json` + `X-AID-Request-Nonce` ヘッダ	買い手が売り手の Agent Card を取得する。売り手 VP は `metadata` に同梱。買い手が売り手 VP を検証し、`seller_nonce` を受領する。
Phase 2: Purchase	`POST /a2a`（JSON-RPC）+ Part（`data` フィールド）	買い手が自身の VP を送信する。KB-JWT に `seller_nonce` echo + `aud` を含む。売り手が買い手 VP を検証する。売り手側には `catalog.search` / `cart.create` / `checkout` の複数の action エンドポイントがあり、action ごとにリクエスト内容は異なるが、いずれの action でも都度買い手 VP を同梱・再検証させる（詳細は 3.6 で後述）。2回目以降の action で使う `seller_nonce` は前 action のレスポンスに同梱されて渡される（3.5 参照）。

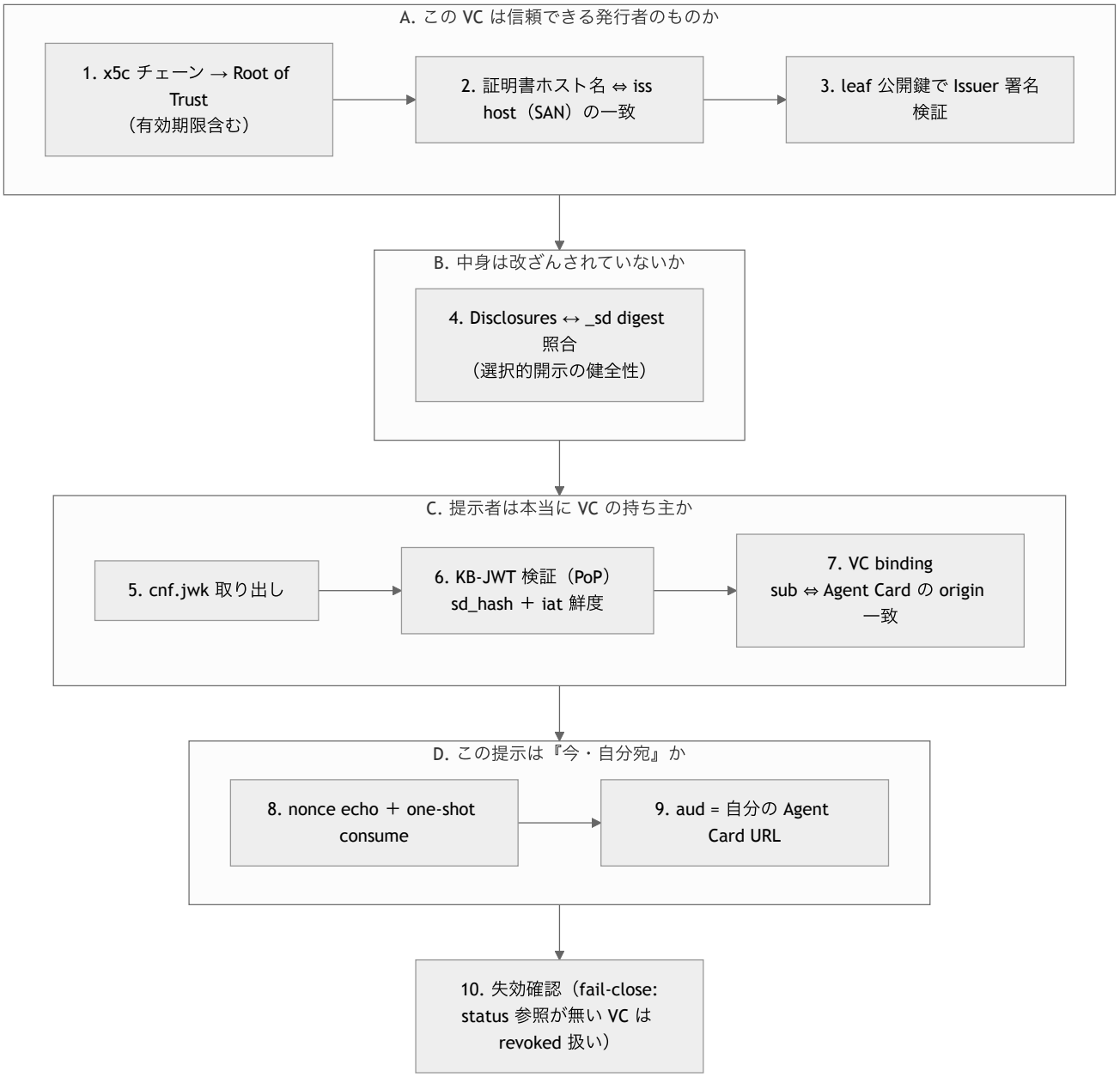
売り手 VP と買い手 VP を非対称に扱うのが本設計の要点である。

- **売り手 VP** は `.well-known` 経由で**公開配信される credential**であり誰でも取得可能。個別バインドの必要性が薄いため、KB-JWT の `nonce` / `aud` 検証は**不要**とし、鮮度は Status List の cache TTL を strict 運用することで担保する。
- **買い手 VP** は購買リクエスト固有（非公開）であり、`nonce`（= `seller_nonce` echo）/ `aud`（= 売り手の Agent Card URL）を**必須**とする。

この非対称性が実務上なぜ効くかは、5.1 ① で述べる。

3.5 VP の検証パイプライン（10ステップ）

VP を受け取った側は、10ステップの検証を順に適用する。**この検証パイプライン自体は売り手 VP・買い手 VP の双方で共通**であり、各ステップは独立した純関数として実装し、適用可否をポリシーで切り替える構成とした。ステップは、証明したい命題ごとに4群へ整理できる。整理した全体像を図示すると次の通りである。



群 A〜C・10の8ステップは、売り手 VP・買い手 VP のどちらの検証でも共通して適用される。 差分は群 **D**（nonce echo + `aud` 検証）のみで、これは**買い手 VP にのみ適用**する。3.4 で述べた非対称設計(売り手 VP は誰でも取得できる公開 credential、買い手 VP は当該取引・当該相手専用の非公開提示）に対応するもので、売り手 VP は公開 credential のため「今・自分宛」の証明を求めない。

この10ステップが証明するのは、あくまで VC/VP というデータそのものの真正性（信頼できる発行者による、改ざんのない、提示者本人の、今この瞬間の提示であること）にとどまる。 検証を通過した VC 内の agentCard（skills / provider.organization 等）の各種 claim をどう解釈し、業務判断にどう用いるかは本パイプラインの範囲外であり、各社の業務ロジック・ポリシーに委ねられる。

`seller_nonce` は CSPRNG（256bit）で発行し、**一度だけ消費（one-shot consume）して再利用を拒否する。`(holder 鍵 thumbprint, nonce)` をキーに重複検出し、リプレイを防ぐ。一度きりで消費されるため、Phase 2 で複数 action が続く場合、売り手は成功・失敗いずれの応答にも次回用の**

`seller_nonce` を同梱し、買い手は次の action でそれを echo する。この piggyback の連鎖により、Discovery で最初に受領した1個の nonce だけで以降の全 action を賄う必要がなくなる。

nonce 系の失敗は `nonce_unknown`（未発行）/ `nonce_expired`（期限切れ）/ `nonce_holder_mismatch`（別 Holder が使用）/ `nonce_replayed`（再利用）に細分化している。原因ごとに送信側の対処が異なるためである。

3.6 商取引・決済フロー（独自仕様およびAP2 v0.2）

3.6.1 概要

エージェント経済圏の商取引プロトコルでは、**UCP と AP2 を組み合わせ、「コマース部分」と「決済部分」を分離する**構成が有力な選択肢の一つとされている。UCP は「何を・どこで・いくらで買うか」を、AP2 は「誰が承認したか・決済をどう暗号的に証明するか」を、それぞれ標準化する。この分離により、AP2 はクレジットカードに限らず銀行振込やステーブルコインなど多様な決済手段に対応でき、UCP 以外のコマースプロトコルからも呼び出せる設計になっている。両者を運ぶ Transport Layer（A2A / MCP(JSON-RPC) / REST API 等）には依存しない。本 PoC は Transport に 前述の通り A2A を用いている。

本 PoC は、3.2 で述べた独自拡張 URI（`https://github.com/vesslabs-gmogshd/agent-identity-vc/v0.1`）の仕様の一部として、UCP への移行を見据えた**独自仕様（UCP ライク）で商取引フロー（`catalog.search` → `cart.create` → `checkout` の各 action）を実装し、その上に AP2 の決済承認フローを統合**している。どの部分が既存仕様に準拠し、どの部分が独自かを整理すると次のとおりである。

レイヤー	本 PoC での実装	準拠状況
エージェント間の疎通経路（transport）	A2A の `SendMessage` / Part（`data` フィールド）	**A2A v1.0 準拠**
Agent Identity VC/VP の相互認証	VP を Agent Card の `metadata` / 購買リクエストの Part（`data` フィールド）に同梱する仕組み自体は A2A の `extensions` 機構を利用。一方、3.2 で前述している通り、Agent Identity VC のスキーマと 3.5 の10ステップ検証仕様は本 PoC 独自定義（ただし X.509 PKI によるチェーン検証の仕組み自体は標準的な PKI 機構であり独自定義ではない）	**A2A v1.0 準拠**（`extensions` 機構）＋**独自仕様**（VC スキーマ・検証仕様）
決済承認の暗号的証明（Mandate）	Checkout Mandate / Payment Mandate の構造・検証ロジック	**AP2 v0.2 準拠**
商取引フロー本体（`catalog.search` / `cart.create` / `checkout` という action 名・Part の `data` フィールド構造、および `cart.create` action が返す `issued_checkout_jwt` のスキーマ）	本 PoC 独自定義	**独自仕様**（UCP 非準拠。UCP 移行前の暫定形。6.2 参照）

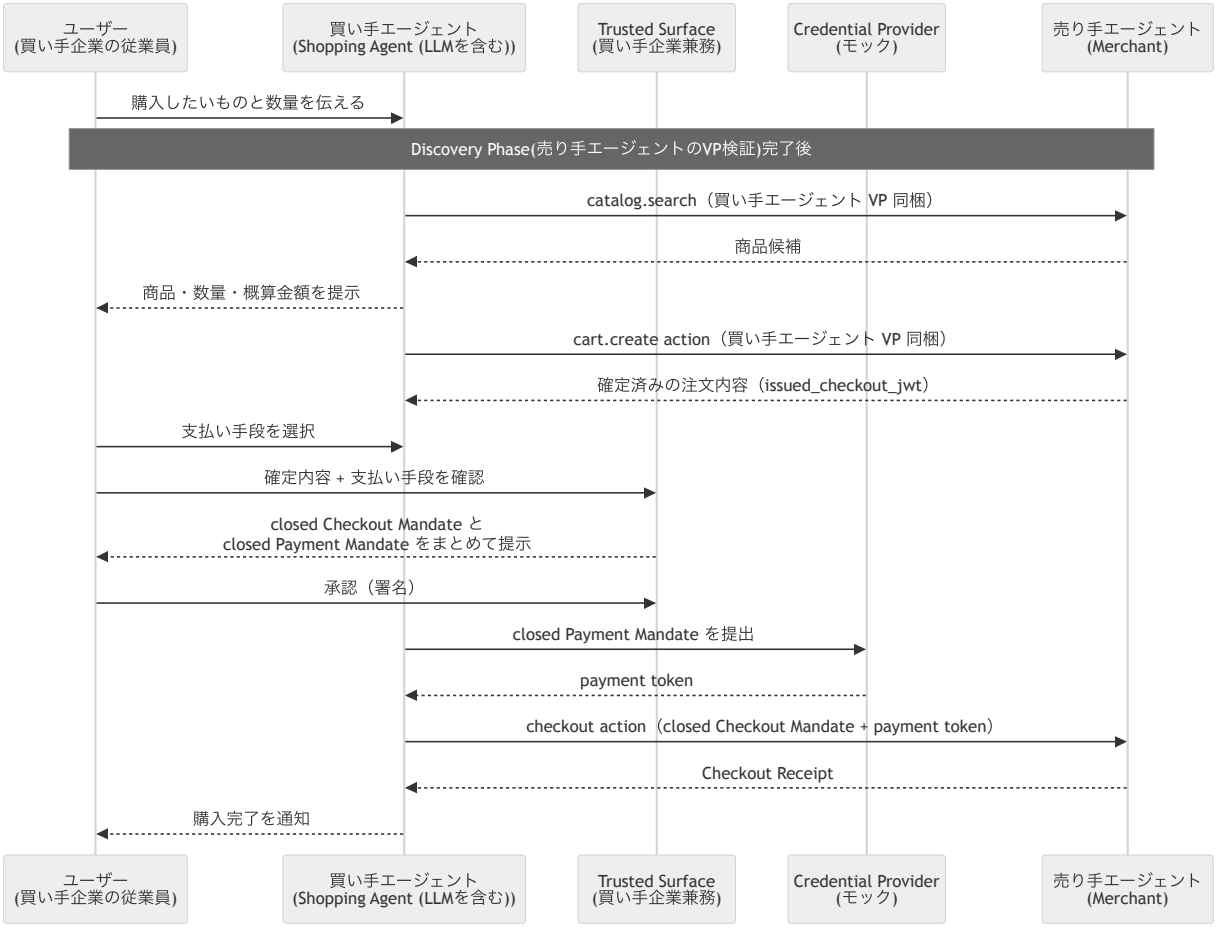
商取引フロー本体を構成する3つの action の役割を整理すると次の通りである。

action	目的	リクエストの主な内容	レスポンスの主な内容
`catalog.search`	商品検索	検索キーワード	該当する商品候補の一覧
`cart.create`	注文内容の確定	商品 ID・数量	売り手が署名した `issued_checkout_jwt`
`checkout`	決済実行	closed Checkout Mandate ＋ payment token	Checkout Receipt

`issued_checkout_jwt` は、`cart.create` action の結果として売り手が発行する **注文内容の確定版**である。商品・数量・金額といった注文内容を売り手が署名して固定し、これ以降の Checkout Mandate はこの確定済みの注文内容に対して発行される。ユーザーが承認するのはこの確定版であり、承認後に注文内容が変わっていないことは `checkout_hash`（`issued_checkout_jwt` の SHA-256）を Checkout Mandate に含めることで担保する。

VC/VP の交換と Mandate の構造は既存仕様（A2A の `extensions` 機構、AP2 の Mandate）に則っている一方、それらを運ぶ商取引フロー自体（action 名・Part の `data` フィールド構造）は本 PoC 独自のものである、という切り分けが本節以降の前提になる。

VC/VP の相互認証（Discovery）が完了した後、Agent 間・TS・Credential Provider の間でどのようなメッセージがやり取りされるかを大まかに図示すると、以下のようになる。



なお、TS は本 PoC では買い手企業自身が兼務しているが、Agent 本体とは分離した鍵・プロセスで運用される独立の署名主体として描いている。またCredential Provider は本 PoC ではモック実装であり、実運用ではカード発行会社等の外部事業者が担う想定である。（次章参照）

3.6.2 本PoCにおけるAP2が果たす役割

AP2 の役割に当てはめると、買い手エージェントが **Shopping Agent**、売り手エージェントが **Merchant** に対応する。加えて、後述のとおり本 PoC では**買い手側が TS（Trusted Surface、ユーザーの承認を受けて Mandate に署名する信頼された仲介者）も兼ねている**。ユーザーは Trusted Surface 上の UI で内容を確認・承認するが、その承認を暗号署名に変換する鍵は独立した第三者ではなく、買い手自身のバックエンドが保持・運用している。ただしこれは組織としての兼務であって、エージェント本体とは分離した鍵およびX.509 証明書チェーンを用いて運用しており、Agent 自身が単独で Mandate に署名することはできない。同じ鍵にしてしまうと Agent が誰の関与もなく自分自身で Mandate に署名できてしまい、「ユーザーが確定内容を見て同意した」ことの証明にならないため、意図的に分離している。

AP2 は、人間が取引のたびに承認するかどうか（**Human Present フロー / Human Not Present フロー**）と、前述の TS の役割、すなわち Mandate に署名する鍵を誰が持つか（**Trusted Agent Provider モデル**: ユーザとは別の仲介者が鍵を持つ／**User Credential モデル**: ユーザー自身の Wallet が鍵を持つ）という2つの軸の組み合わせで、4つのフローパターンを取りうる。**本 PoC が実装しているのはこの内、Human Present フロー × Trusted Agent Provider モデルの1パターンである。**

人間の承認フロー \ TS(署名鍵の主体)モデル	Trusted Agent Provider	User Credential
Human Present	**本 PoC で実装**	未実装
Human Not Present	一部実装済み	未実装

ユーザーは取引のたびに Trusted Surface で内容を確認・承認し、その承認を暗号署名に変換する鍵は TS（買い手側が兼務）が保持する。他の3パターンの位置づけは 6.2 で述べる。またこの構成に対応して、本 PoC で現時点で登場する Mandate は closed Checkout Mandate および closed Payment Mandate の2種類になる。

決済側に登場する次の2つの当事者は、本 PoC ではいずれも**実体を持たないダミー**である。

Credential Provider - 買い手エージェント（Shopping Agent）からアクセスされ、ユーザーの支払い手段を管理し、closed Payment Mandate を検証したうえで決済用トークン（payment token）を発行する当事者である（実運用では、企業管理の買い手エージェントであればコーポレートカード発行会社等、個人ユーザーであれば Google Wallet 等のデジタルウォレット事業者が相当すると考えられる）。本 PoC ではモック実装にとどまり、署名検証を行わずダミーの payment token を返す。

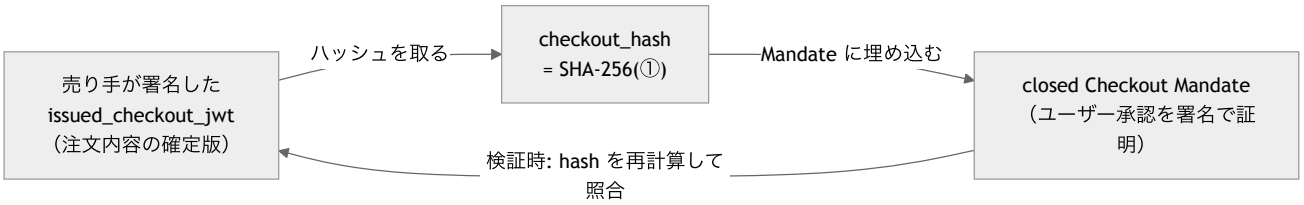
MPP（Merchant Payment Processor） - 売り手エージェント（Merchant）からアクセスされ、実際の決済ネットワークとやり取りして決済を実行し、その結果（payment_receipt）を返す当事者だが、本 PoC ではモックとしても実装しておらず、対象外としている。

3.6.3 Checkout Mandateのバインド構造

3.6.1 で述べた `issued_checkout_jwt`（注文内容の確定版）と closed Checkout Mandate は、次の手順で不可分に結び付けられる。

1. `cart.create` action にて、売り手が署名済み `issued_checkout_jwt` を返却する。
2. 買い手が closed Checkout Mandate に `checkout_hash = SHA-256(issued_checkout_jwt)` を埋めてバインドする。
3. `checkout` action にて買い手がclosed Checkout Mandateを売り手に送信する。
4. 売り手が検証する。

以下に、このバインド構造を図示する。



closed Checkout Mandate の検証内容は次の3点である。

- `closed` の x5c チェーン検証（→ RoT）+ leaf 鍵署名検証

- `checkout_hash` 照合
- `exp`（有効期限）確認

Human Present フローでは open Checkout Mandate は登場しないため、`open` と `constraints` はいずれも使わない。ユーザーが Trusted Surface 上で承認した対象は内容そのものであり、注文内容とのバインドは `checkout_hash` の照合によって担保される。この構造により、**ユーザーが承認した対象は「売り手自身が署名した注文内容」に一意に固定される**。買い手が注文内容を差し替えても hash が変わるため検証に失敗し、逆に売り手が注文内容を後から書き換えても同様に失敗する。承認と注文内容のバインドが、双方の改ざんに対して成立する。

なお、決済面においては、AP2 v0.2 の責務分界に従い **Merchant は Payment Mandate を直接検証しない**（Credential Provider が検証してトークン化し、Merchant は `payment_token` を受け取る）形をとる。

3.7 LLMの利用

買い手側は LLMをカタログ検索フェーズの解釈・選定に用いている。ユーザーの自然文入力（例:「防水のトレッキングシューズを探して」）から、検索意図の解釈、売り手候補の選定、検索結果(売り手エージェントからのレスポンス)からの商品選定、ユーザー向け要約文の生成までを LLM が担う。

一方、**Mandate による注文内容との結合（`checkout_hash` 照合）や各種検証（VP の10ステップ検証、x5c チェーン検証、署名検証、`nonce` / `aud` 照合、失効確認等）には、買い手・売り手のいずれにおいても LLM は一切関与せず決定論的処理で行う**。「エージェントが自律的に判断する」という論旨の対象は前者（何を買うか探す判断）であり、後者（承認内容が改ざんされていないことの保証）は双方の実装で暗号的検証がすべてを担っている——この境界を区別することが、本方式の信頼確立が LLM の振る舞いに依存しないことの根拠になる。

3.8 実装構成

両実装とも TypeScript / Node.js で構築したが、**実行モデルは意図的に揃えなかった**。買い手側はリクエスト単位で起動するイベント駆動型(サーバレス型)、売り手側は常駐プロセス型である。

この非対称は結果として有益だった。**同一の設計を、状態の持ち方が異なる2つの実行モデルで独立に実装したことになり、仕様が特定の実行形態に暗黙依存していないことの検証になった**からである。特に `seller_nonce` のライフサイクル（発行・一度きりの消費・失効）は、常駐プロセスならメモリ上で完結するが、イベント駆動型では外部ストアが要る。この差が仕様の穴として表面化しなかったことは、設計の可搬性を示している。

なお売り手側は、VC フォーマットを抽象化した `VcFormatAdapter`（`dc+sd-jwt` を第一実装、`vc+jwt` を将来差替スタブ）を境界に置き、検証・提示・Mandate 検証がこの interface 越しにのみ VC を扱う設計とした。フォーマットの変遷（`vc+sd-jwt` → `dc+sd-jwt` のような media type 変更や、将来の別フォーマット併存）を、検証ロジックに波及させないための構造である。

また買い手側はフロントエンド（UI）を持つ。ユーザー（買い手企業従業員）が商品を検索・発注するチャット画面と、売り手側とのやり取り履歴を確認する管理画面の2つを実装しており、3.7 で述べた自然文からの商取引意図の解釈は、このチャット画面のプロンプトからバックエンドが Gemini API（LLM）を呼び出す形で実現している。

4. 成果

4.1 独立実装の実結合による全シナリオ疎通

2社が独立に実装した買い手・売り手エージェントを直結し、一通りのシナリオの検証にすべて成功した。

1社が両側を作ると、設計上の前提が暗黙のうちに両側で共有されるため、方式として成立しているかを確かめられない。**仕様のみを共有した2つの独立実装が接続できたことは、本方式が特定の実装に依存せず成立することを示している。**

4.2 End-to-End 実証と検証水準

売り手側・買い手側それぞれのエージェントが、実際にどこまで本番相当の検証を実装し確認できているかを以下に示す。

4.2.1 売り手側エージェント

売り手側エージェントはクラウド上で実稼働しており、Discovery → 相互認証 → 商取引 → 決済の取引完結を確認済みである。ローカル環境ではなく、**実際の TLS 終端と公開 DNS を伴う経路で検証している**点が、証明書チェーン検証を含む本方式では重要になる。

技術的な中身として重要なのは、**以下を AP2 の前段に組み込んで動作させた**点である。

- **VC の発行** — Issuer として Agent Identity VC を発行する API を実装した。スキーマ検証を**署名前**に行い（不正な内容に署名を与えない）、失効管理用に Status List の index を払い出す。
- **VC の提示** — 売り手 VP は Agent Card の `metadata` に同梱して公開配信、買い手 VP は購買リクエストの Part（`data` フィールド）に同梱。A2A の `extensions` 機構に載せる形で実現した。
- **VC の検証** — 3.5 の10ステップパイプライン。X.509 チェーン検証・選択的開示の健全性照合・鍵所持証明・失効確認を含む。
- **Mandate による委任の実現** — AP2 の Checkout Mandate によりユーザー承認を暗号的に証明し、`checkout_hash` で売り手発行の注文内容とバインドした。

また、**「正しく拒否できること」を e2e で網羅的に駆動**している点も設計上の要点である。nonce の再利用 / 未発行 nonce / 別 Holder による nonce 使用 / `aud` 不一致 / nonce echo 欠落 / 失効済み VC / `checkout_hash` 不一致 / 注文内容偽造（Merchant 署名でない `checkout_jwt` の混入）について、いず

れも期待どおり拒否されることを確認した。**攻撃面の多くは「検証をすり抜けられるか」に集約されるため、異常系こそが本方式の実効性を示す。**

商用運用で必要となる検証——**X.509 チェーン検証（Trust Anchor 到達）、失効確認（Token Status List、fail-close）、証明書ホスト名 ⇔ `iss` host 一致、Human Present フローでの Checkout Mandate 検証**——は、現在すべて本番設定で有効化されている。

> **成果の正確な位置づけ**: **支払いネットワークへの転送は mock** で代替しており（`payment_token` の中身は検証せず素通しする）、本番決済や実商流ではない。商品カタログも試験用の固定データである。検証したのは**相互認証から取引成立までの信頼確立機構**であって、決済処理そのものではない。また、mTLS 等の A2A `securitySchemes` による認証は必須とせず（Agent Card の `securitySchemes` は未設定）、A2A メッセージに含まれる VP の検証を主要な認証手段（primary security control）としている。
>
> 上記の異常系を含む自動テストは、**売り手側**で unit + e2e 281 件が通過している（2026-08-28 実測）。

4.2.2 買い手側エージェント

買い手側エージェントもクラウド上で実稼働しており、実際の売り手側エージェントとの通信で Discovery → 相互認証 → 商取引 → 決済の取引完結を確認済みである。売り手側がクラウド上で常駐プロセスとして動くのに対し、買い手側はサーバーレス（イベント駆動）で実装しており、**実装形態が異なる2つの独立実装が接続できた**ことも4.1の主張を補強する。

技術的な中身として重要なのは、以下の点である。

- **VC の検証** — 売り手 VP の検証を、売り手側と同一仕様の10ステップパイプラインとして買い手側でも独立実装した。X.509 チェーン検証・選択的開示の健全性照合・鍵所持証明・失効確認を含む。加えて、Discovery 時に取得した売り手 VP は checkout 提出の直前にも鮮度を再確認しており（有効期限・失効状態のみの軽量チェック）、セッション内で古い VP を使い回すリスクに対処している。
- **VP 検証後の claim 利用** — 3.5 で述べた通り、検証を通過した後の claim 解釈は各社の業務ロジックに委ねられる整理としており、本 PoC における買い手側の扱いを例示する。`provider.organization` は、Issuer の署名により改ざんが防止された文字列であることを活かし、Trusted Surface の承認画面に表示して、ユーザーが Mandate を承認する際の判断材料としている。`supportedInterfaces` は、商取引フロー本体（`catalog.search` / `cart.create` / `checkout`）の各 action にアクセスする際の実際の endpoint 情報として利用している。一方 `skills` については、フローを開始する前に必要な `skills` が一通り揃っているかを事前に確認するという設計もありうると考えているが、本 PoC では実装していない。
- **自然文からの商取引意図の解釈** — カタログ検索フェーズでの LLM 活用（3.7参照）。

- **Mandate による委任の実現（Human Present フロー）** — closed Checkout Mandate・Payment Mandate への署名は、買い手エージェント本体とは分離した鍵およびプロセスを持つ TS（Trusted Surface。本 PoC では買い手企業が兼務）が行う。ユーザーが Trusted Surface 画面で内容を確認し明示的に承認しない限り、これらの Mandate は発行されない。
- **Credential Provider を介した分離提出** — closed Payment Mandate は Credential Provider（モック）にのみ提出し、そこで発行された payment token（生の支払い情報を含まない参照文字列）と closed Checkout Mandate のみを Merchant に提出する、AP2 Human Present フローの2段階提出を実装している。

4.3 デモ動画

本レポートの補足資料として、買い手企業の従業員がユーザーとして本フローを実際に操作するデモ動画を公開している。チャットで商品を検索させ、Trusted Surface 画面に表示された Checkout の内容を確認して明示的に承認し、Checkout Mandateの発行から売り手エージェントへの提出とcheckout receiptの受領までの取引が完了する一連の操作を収めている。

- https://youtube.com/watch?v=tHjOwEQURn4

5. 得られた知見

5.1 設計上の学び

① 「誰にでも見せる証明書」と「相手を選んで出す証明書」は、設計を分けるべき

売り手の身分証（Agent Identity VC）は、店舗の看板と同じく**誰に見られても構わない公開情報**である。一方、買い手の身分証は「この取引のために、この相手に対してだけ提示するもの」であり、性質が根本的に異なる。

買い手の身分証には「今回の取引限り」という印（使い捨ての nonce と宛先 `aud`）を必須とした。これにより、**ある取引で提示した身分証を、第三者が別の取引で使い回すこと（リプレイ攻撃）ができない**。逆に売り手側にこの制約をかけると、看板を見るたびに事前のやり取りが必要になり、取引開始のたびに余計な往復が発生する。

具体的にどう効くか——調達エージェントが複数の仕入先を比較検討する場面を考える。売り手の身分証が公開情報であるため、買い手は**50社の身分証を、自分の素性を一切明かさずに取得・検証できる**。「実在する企業か」「この商材を扱う正規の事業者か」を、問い合わせも交渉もなしにその場でふるいにかけ、実際に発注する1社に対してだけ自分の身分証を提示する。

これを対称に設計すると、比較検討の段階で50社すべてに自社の素性と購買意図を晒すことになる。**自社の調達動向が競合他社に漏洩するリスクがある**うえ、偽の売り手を1社立てるだけで買い手の証明書を

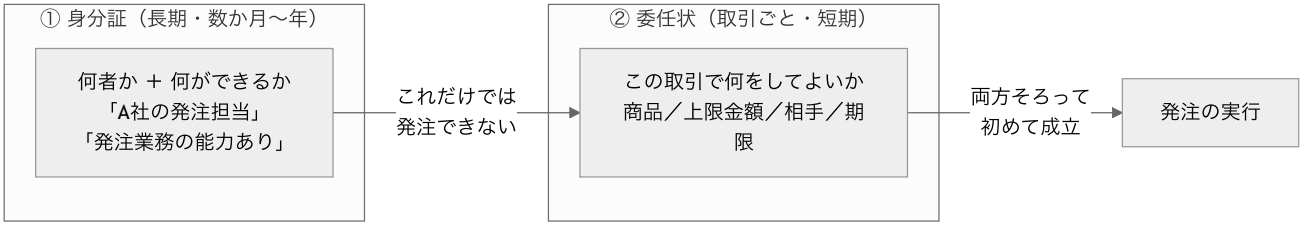
収集できてしまう。非対称設計は、安全性と実用性の両立に加えて、**「見比べる」という商取引の基本動作を成立させる**ために必要だった。VC を使う実システム設計で一般化できる論点である。

② 身元の証明（何者か）と権限の証明（何をしてよいか）を分ける

前者を身分証（Agent Identity VC）、後者を委任状（Mandate）が担う2層構造とした。**身分証だけでは発注が成立せず、取引ごとの委任状が必ず必要**という構造にすることで、以下の問題を防ぐ。

身分証には「このエージェントは発注業務ができる」という能力が記載される。この記載だけで発注を許すと、**身分証が有効な数か月〜年のあいだ、金額も品目も無制限の発注権限**として機能してしまう恐れがある。

そこで本設計では、身分証は「発注業務ができる能力がある」ことしか意味せず、**実際に発注してよいかは取引ごとに発行される委任状が決める**構造にした。委任状には「この商品を」「この上限金額まで」「この売り手に対してのみ」「この時刻まで」という制約が書き込まれ、売り手はそれを検証する。



結果として、身分証が流出しても委任状がなければ発注は成立しない。**能力と認可を分離することで、権限が「一度与えたら広がりっぱなし」になる構造を断ち切っている。**

③ VC は既存のセキュリティを置換せず、性質が異なる

まず本 PoC は Agent Card の `securitySchemes` を未設定（空）のまま、VP 相互認証を primary security control としている（`securitySchemes` はオプショナルであり、A2A 1.0 の仕様上に準拠した構成である）。

A2A の `securitySchemes`（apiKey / HTTP Auth（Basic・Bearer）/ OAuth2 / OpenID Connect / mTLS の5種）は、いずれも**接続・リクエストのたびに都度提示する資格情報**であり、Verifier が単独で検証できるとは限らない—OAuth2 のトークンは典型的に発行元（AS）への問い合わせを要し、mTLS も接続が終端した時点でその証明は失われ、後段のアプリケーション層やログには伝播しない。対して VC は、身元情報を署名としてデータそのものに埋め込むため、一度トラストアンカーを共有しておけば、以後は検証のたびに発行元へ問い合わせることなく独立して検証でき、選択的開示や（nonceや `aud` の制約下での）複数 Verifier への提示も可能である。VC はこれらの securitySchemes 単独では得にくい能力を提供する。つまり両者は置換ではなく補完関係にあると考える。なお、両者（たとえば mTLS と VC を）併用する場合の整合性設計は本研究のスコープ外とする。

6. 事業的な示唆と今後の展望

6.1 なぜ事業として意味があるか

① 商取引にかかる人的コストの削減

従来、企業間の発注には相手の与信確認・担当者の承認・発注書の授受といった人的工程が伴う。本研究が実証したのは、その前提となる「相手を信頼してよいか」の判定を自動化する仕組みであり、**ここが自動化されて初めて取引全体の自動化が成立する**。人間が承認の起点として関与し続けたまま（Human Present フロー）実行はすべて自動化する——その第一歩を実装で示せたことが、中核的な意義である。

② 自動取引でも、責任の所在が追跡可能になる

自動化には「問題が起きたとき誰の責任か」という懸念が必ず伴う。本方式では、**「誰が誰に、どの範囲の権限を委任したか」が署名付きで記録される**。これは単なる操作ログとは質が異なり、**第三者が後から暗号的に検証できる証跡**である。監査対応・紛争解決・内部統制の観点で、自動取引を実運用に載せる際の前提条件を満たす。

③ Allowlist からの脱却

現状のエージェント取引は、信頼できる相手を人手でリスト管理している。VC による自動検証は、**エージェントが増えるほど効いてくる**（1.2 の組み合わせ数の議論）。相手が未知の組織であっても、信頼の起点さえ共有されていれば検証が成立する。

> **Registry との関係**: 本 PoC の買い手側は、取引候補となる売り手エージェントの一覧を静的な Registry（Agent Card URL と取扱カテゴリ等の記述的メタデータのみを保持）から取得する。これは Allowlist と紛らわしいが役割が異なる。**Registry が担うのは discovery（どこに問い合わせるか）のみであり、trust の判定は一切行わない。** Registry のエントリが汚染されて悪意ある Agent の URL が混入していても、その Agent は本方式の VP 検証（3.5）を通過できないため、信頼判定という意味では無力化される。Allowlist は「登録されていること自体」が信頼の根拠になるのに対し、本方式では**信頼は Registry の中身と独立に、接続のたびに暗号的に再検証される**。したがって Registry の規模や構成は信頼境界に含まれない。

④ 適用範囲は特定業種に限定されない

本人確認・組織の実在性確認・権限の確認は、**一般的に商取引に共通して必要となる工程**である。したがって本方式の応用範囲は、原理的にはエージェントが関与する商取引全般に及ぶ。そのうえで、**検証の厳密さが特に強く求められる領域**——たとえば電子証明書の発行、本人確認・年齢確認を伴う規制商材の取引、多段の委任を伴う代理購買など——は、効果が顕在化しやすく、初期の適用先として有力である。

6.2 技術面の拡充

実証済みの基盤に対し、標準準拠度と実用性を高めるアップデートを進める。現時点の候補は以下であり、優先順位を付けて順次対応していく想定である。

- **UCP の導入** — 現在の実装は A2A v1.0 / AP2 v0.2 に加え、購買フロー部分が独自仕様になっている。この部分を **UCP 準拠**に置き換える。UCP は、Google と Shopify が Etsy・Wayfair・Target・Walmart とともに策定した、AI エージェントによる商取引のためのオープン標準である（AP2 と UCP の関係の詳細は下記補足）。本実装が暫定的に独自の拡張 URI を用いているのはこの移行期にあるためで、**エコシステム接続性の観点から優先度が高い**。あわせて `checkout_jwt` の署名鍵分離（現在は VC Holder 鍵を流用しており、「Agent 本人性」と「Merchant としてのカート保証」が同一鍵に乗っている）も整理する。

  補足すると、AP2 v0.1 では A2A Extension（`google-agentic-commerce/ap2/tree/v0.1`）を Agent Card の `capabilities.extensions` に宣言することで AP2 対応を discovery できる仕様だったと理解している。v0.1 から v0.2 へのリポジトリ構成変更に伴い、これに相当するドキュメントファイルは見当たらなくなっているが、AP2 のドキュメント本文には引き続き「A2A・UCP いずれの extension としても利用可能」という記述が残っており、A2A のサンプルも残存していると理解している。したがって AP2 が A2A extension としての利用を仕様上やめたとは言えない。一方 UCP は `.well-known/ucp` による独自の capability discovery 機構を持つが、これは UCP 自身の仕組みであり、AP2 仕様がこれを要求・追加したものではないと理解している（AP2 は transport 非依存である）。本 PoC の実装は Mandate（Checkout Mandate 等）としては実質 v0.2 相当だが、公式の `v0.1` 拡張 URI をそのまま名乗ると「古い Mandate 仕様（Intent, Cart 等）である」という誤ったバージョン主張になってしまう一方、`v0.2` に対応する公式の A2A 拡張 URI が存在するかは確認できていない。そのため、**UCP 移行が完了するまでの暫定措置として**、独自 URI（`vesslabs-gmogshd/ap2/tree/v0.2`）を「AP2 v0.2 相当の A2A Extension」という意味で宣言することにした。UCP 準拠化が完了すれば discovery 方式ごと `.well-known/ucp` に置き換わり、この独自拡張 URI 自体が不要になる想定である。

- **Human Not Present フローの導入** — 人間が都度承認しない自律取引へ拡張する。売り手側には Human Not Present フローの検証ロジックがすでに実装されているが、買い手側は open Mandate の構築・委任チェーンの発行を行っておらず、E2E では未検証である。

  売り手側の Human Not Present フローの検証内容について簡略に述べる。Open Mandate を Trusted Agent Provider 鍵（x5c → RoT）で、closed Mandate を `open.cnf.jwk` の鍵で検証し、両者の `exp`・`checkout_hash`・`constraints`（`allowed_merchants` / `acceptable_items` / 数量）を完全評価する、というものである。

- **Trusted Surface における User Credential モデルの導入** — 現在の Trusted Agent Provider モデルに加え、ユーザー自身が保有する資格情報を用いる方式を検討する。ここで Wallet / Digital Credentials API / OID4VP が登場することになり、VC の本来の応用領域と直結する。

- **継続的なリクエスト認証** — 毎リクエストで VP 検証を行うのではなく、初回以降は短期 Bearer JWT もしくは Agent Identity VC の `cnf` を用いた DPoP による認証に切り替える。設計論点として、短期トークンの有効期間と失効反映のタイムラグをどう許容するか、また**セッションが途切れた後の再開時**に VP 検証からやり直すのか短期トークンの再発行のみで継続させるのか（後者は途切れている間の失効・権限変更を捕捉できないリスクがある）といった継続性の論点がある。
- **Extended Agent Card の導入** — 取引先ごとに変わりうる `skills` 等を Public Card に含めず、VP 認証後に Extended Card として公開する。

6.3 ガバナンスとエコシステム形成の課題

6.2 が実装機能の拡充であるのに対し、以下の2点は機能追加では閉じない、体制・合意形成の課題である。本 PoC はいずれもスコープ外としており（3.1）、実運用に向けては別途取り組みが必要になる。

- **Agent 審査基準・審査プロセスの確立** — 本 PoC では Issuer が Agent Identity VC を発行する仕組みまでを実装し、「実在する組織か」「申告どおりの capability を持つか」を Issuer がどう審査するかは対象外とした。実運用では、この審査の厳密さこそが VC の信頼性の実体的な担保点になる。
- **Trust Framework の策定とエコシステム形成** — 本 PoC は、共有 Root of Trust から売り手・買い手双方の Issuer に至る Trust Chain（Certificate Chain）を仮想的に構築し、そのチェーンの上で相互認証が成立することを検証した。しかし実運用に向けては、このチェーンを成立させる Trust Framework 自体の策定、およびそれを複数の参加組織に適用するエコシステムの形成が、本 PoC のスコープ外として残る。

  Trust Framework とは、どのような審査基準を満たした組織が Chain に参加できるか、Root CA から末端 Issuer に至る階層構造をどう設計するか、証明書ポリシー（CP）・認証局運用規程（CPS）に相当する規約をどう定めるかという信頼の設計と、誰が Root を運営するか、参加基準の改定・審査・失効の判断を誰がどのプロセスで下すかという運営（ガバナンス）の設計を、不可分な一つの枠組みとして含む。これを策定し、複数の Issuer・Root CA が参加できるエコシステムとして機能させて初めて、本方式は「共通の信頼の起点さえあれば事前合意なしに初めて出会う相手を検証できる」という価値を実運用で発揮する。

6.4 具体的ユースケースの深掘り

事業性を意識した、第三者を巻き込みうるより具体的なユースケースへの取り組みを検討する。VC でなければ解けない領域——規制産業（KYC / 年齢確認付き購買）、多段委任の権限縮小、電子証明書の Agent 間購買など——から、実際の商流と接続できるものを選定する（別途企画検討中）。

7. まとめ

AIエージェントが組織を越えて取引する時代の信頼確立を、Authorization Server に依存しない VC + PKI の双方向相互認証として設計・実装した。A2A v1.0 上に AP2 v0.2 の決済フローを載せ、その前段に SD-JWT VC の発行・提示・検証（選択的開示・鍵所持証明・失効確認を含む10ステップ）を組み込み、Mandate による委任の暗号的証明を実現した。

成果として、**2社が独立に実装した買い手・売り手を直結し、一通りのシナリオの疎通に成功**した。仕様のみを共有した2つの独立実装が接続できたことは、**本方式が特定の実装に依存せず成立する**ことを示している。

組織横断の認可は OAuth 側でも標準化が進むが、その成立条件は Authorization Server 間の事前合意にある。本研究が示したのは、**共有 Root of Trust のみを前提として、Issuer の審査を伴う信頼を、事前の相対合意なしに組織間で成立させる**方式が、実在のエージェント間プロトコル上で動作するということである。

商取引の自動化は、「相手を信頼してよいか」の判定が自動化されて初めて成立する。本研究はその判定部分を、実装をもって成立させた。エージェント経済圏の相互運用に向けた、実践的な一歩である。

付録：用語

- **エージェント（AI Agent）**: ユーザーの代理で自律的に判断・行動する AI。
- **A2A**（Agent-to-Agent Protocol）: エージェント間でタスクを委任・協調するプロトコル。
- **AP2**（Agent Payments Protocol）: エージェントによる決済プロトコル。ユーザー / Trusted Agent Provider の意思を Mandate として暗号的に証明する。
- **UCP**（Universal Commerce Protocol）: エージェント経済圏における商取引フローのオープン標準。AP2 v0.2 は UCP との統合を前提に設計されている。
- **VC / VP**: Verifiable Credentials / Verifiable Presentation。発行者が署名した検証可能な証明とその提示。受け取った側が発行元に問い合わせることなく真正性を検証できる。
- **SD-JWT VC**: 選択的開示に対応した JWT ベースの VC フォーマット。wire 上は `<JWT>~<disclosure>~...` の compact serialization をとる。
- **KB-JWT**（Key Binding JWT）: Holder が鍵所持を証明する JWT。PoP（Proof of Possession）を担う。
- **選択的開示**: 証明書の中の必要な項目だけを見せ、他は伏せられる仕組み（例：生年月日を見せずに「20歳以上」だけを証明する）。
- **Mandate（委任状）**: ユーザー / Trusted Agent Provider の意思を暗号的に証明する署名付きデータ（AP2）。誤発注や無断決済を防ぐ。
- **Checkout Mandate**: Merchant署名済みの checkout（品目・数量・金額を含む）への承認を証明する Mandate。closed は checkout_jwt とそのハッシュを保持し、open は品目・マーチャントの制約のみを定める。

- **Payment Mandate**: 特定の checkout に対する支払いの承認を証明する Mandate。closed は支払先・金額・支払手段を確定値として保持し、open は金額レンジや利用可能な手段などの制約を定める。
- **PKI / 認証局**: デジタル証明書の信頼を支える仕組みと、その発行主体。
- **Root of Trust（RoT）**: 証明書チェーン検証の起点となる信頼アンカー。
- **Allowlist**: 信頼できる相手を人手で登録した許可リスト。
- **Human Present フロー / Human Not Present フロー**: 取引の起点に人間の承認が存在する形態 / エージェントが自律的に実行する形態。

著者

- GMOグローバルサイン・ホールディングス株式会社
- 株式会社VESS Labs

> ※ 実装状況は 2026-08-28 時点の実測に基づきます。