

TECHNICAL CASE STUDY

IMPLEMENTING THE CHAIN OF TRUST FOR AN AGENTIC TRANSACTION

Agent identity, authentication and accountability in a cross-organisation flight purchase under Singapore's Model AI Governance Framework for Agentic AI

15 OCTOBER 2026

Contents

Executive Summary.....	3
1 The Experiment.....	4
1.1 Agents Involved.....	4
1.2 Steps Involved.....	5
2 Open Standards Used in the Experiment.....	6
2.1 Naming: DNS, DNSSEC and DANE.....	7
2.2 Discovery: DNS-AID.....	7
2.3 Registration: ANS.....	8
2.4 Capability Discovery and Trust Attestations: ARD.....	10
2.5 Description: the A2A agent card and MCP.....	10
2.6 Commerce: UCP, the Universal Commerce Protocol.....	12
2.7 Authentication: HTTP Message Signatures and SPIFFE.....	14
2.8 Authority: passkeys, AP2 mandates and SD-JWT.....	14
2.9 Records: SCITT and the Open Trust Index.....	14
3 Meeting IMDA's Agent Identity Requirements.....	15
3.1 Identity: unique, accounted for, differentiated and catalogued.....	15
3.2 Authority: scoped, least privilege, time-bound, non-transferable and bounded.....	17
3.3 Withdrawing identity and revoking authority.....	19
3.4 Auditability: A record each side can hold the other to.....	19
4 Conclusion.....	21
Annex: Standards Adopted in Case Study.....	22

Authors

- Wei Chen – Infoblox
- Andrei Ciobanu – GoDaddy
- Scott Courtney – GoDaddy
- Fabio Ramos – Infoblox
- Jim Mozley – Infoblox
- Ingmar Van Glabbeek – Infoblox

Executive Summary

This technical case study applies the agent identity best practices in Section 2.1.2 of IMDA's Model AI Governance Framework for Agentic AI to a completed transaction spanning multiple parties across multiple platforms.

The transaction involved a Madrid-to-Singapore flight purchase completed by four independently deployed agents: a buyer, a seller, a mandate authority that recorded the person's approval, and a verifier. The agents ran across three cloud providers and four domains, with each party verifying the others through published records and cryptographic signatures rather than relying on a shared platform. The case study also involved several simulated "rogue" agents to test the security of the reference architecture.

The case study demonstrates that secure agent-to-agent transactions across organisational boundaries can be built on existing internet infrastructure and established trust mechanisms, rather than relying upon centralised platforms. By using tested protocols for discovery, identity verification and authentication, it shows a practical path to interoperable agent transactions that can scale across independently operated platforms while preserving security, accountability, and verifiability.

1 The Experiment

Many agent identity frameworks were written for agents inside one organisation, where a central system can issue, track and withdraw identities. Commerce between agents breaks that assumption. The buyer's agent and the seller's agent belong to different organisations, run on different platforms, and may never have met. Neither can look the other up in its own directory.

The experiment tests whether open standards and existing internet infrastructure can carry the identity, authentication and accountability work that a central system would otherwise do.

1.1. Agents Involved

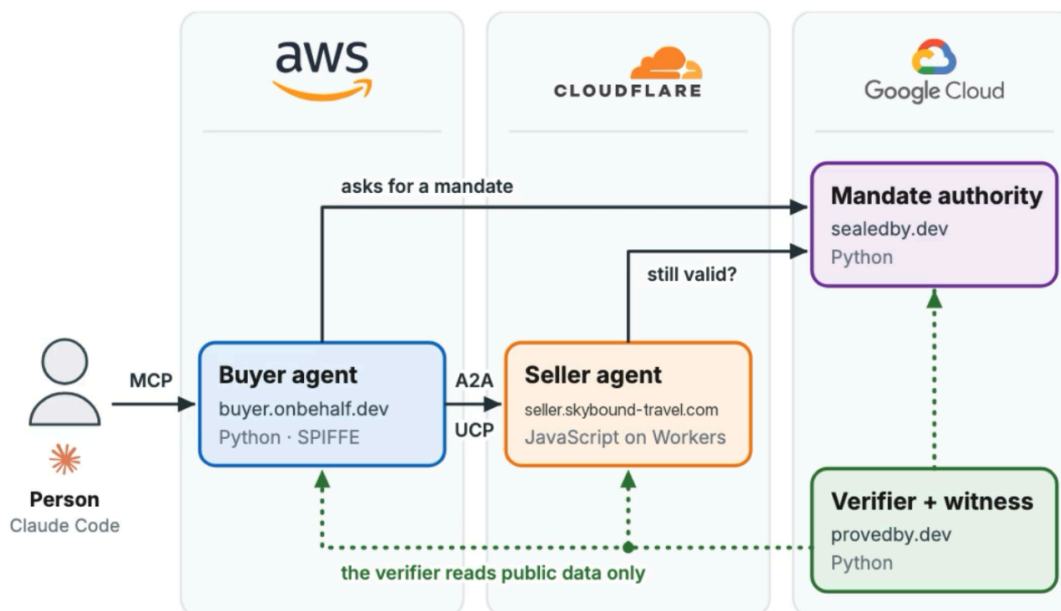
The experiment involved four agents and the person's own AI assistant, which connects to the buyer agent over the Model Context Protocol (MCP).

Table 1 and Figure 1 below list the four agents involved, the personal assistant. We also use several rogue scenarios to test the seller agent's refusals.

Table 1: Agents involved in the experiment

Agent	Domain	Runs on	What it does
Buyer agent	buyer.onbehalf.dev	AWS, Python	Works for the person. Finds the seller, checks it, collects signed prices, asks for the person's approval and completes the purchase.
Seller agent	seller.skybound-travel.com	Cloudflare, JavaScript	Open to anyone who searches for a flight. Signs every price it quotes, and completes a checkout only under a mandate it can verify.
Approval service (mandate authority)	sealedby.dev	Google Cloud, Python	Where the person says yes. Shows what is being bought, applies the person's own limits, signs the mandate the seller checks, and stops any attempt the person has not approved.
Verifier	provedby.dev	Google Cloud, Python	The independent auditor. Repeats the other three's checks from their public records, and publishes a signed verdict.
The person's assistant	Claude Code, on the person's computer	Local	Turns the person's request into a structured call to the buyer agent over MCP.

Figure 1: Agent transaction diagram



1.2 Steps Involved

Below in Table 2 is a list of the steps taken by the person and the agents in the experiment.

Table 2: Steps involved in the experiment

Step	What happened
1. The person asks	The assistant turned the person's request into a call to the buyer's buy tool, using an access token the approval service issued after the person signed in with a passkey.
2. The buyer finds the seller	Starting from the domain skybound-travel.com alone, the buyer found the seller's record in signed Domain Name System (DNS).
3. The buyer checks	The buyer ran fourteen checks on the seller before sending the query.
4. How to buy	The buyer read the seller's agent card, its commerce profile and its tool list.
5. Search and signed quotes	The seller fetched live fares and returned two signed quotes. The buyer recomputed each quote's fingerprint over the signed bytes, chose the cheaper one and opened a checkout session.
6. The person approves	The person confirmed the purchase with a passkey, and the approval service signed two mandates.
7. The buyer asks to buy	The buyer closed both mandates and sent a signed completion request over UCP REST.

Step	What happened
8. The seller checks	The seller ran a number of checks in code before touching any money.
9. Payment and receipts	The seller created and captured a Stripe PaymentIntent, and returned three signed receipts.
10. Records sealed	Each ledger was checkpointed in an auditable log.
11. Independent check	The verifier compared the parties' records and mandate chains, and confirmed the payment at Stripe.

2 Open Standards Used in the Experiment

This section describes the open standards the agents used, grouped by the question it answers about identity, authentication or authority. Table 3 below is a list of the standards and how they are used in the experiment. The full descriptions, versions of the standards and links to the relevant specifications can be found in the [Annex](#) to this paper.

Table 3: Standards used in the experiment

Question	Standards (see Annex)	Used in this experiment
Who controls this name?	DNS: SVCB, DNSSEC, DANE	Domain Name System Security Extension (DNSSEC) signed records under each agent's domain; the public key that every server a verifier contacts must present, published in DNS with DNS-based Authentication of Named Entities (DANE)
Where is the agent, and what does it speak?	DNS-AID	The seller's DNS record and the zone's index of agents
Who registered this agent?	ANS	Registration with a domain check, an identity certificate and a public log entry
Which agents offer this capability?	ARD	Directory of static and dynamic agentic capability discovery, and the verifiable trustManifest attestations anyone can check
What does the agent say about itself?	A2A agent card; MCP	The seller's description, interfaces and skills; typed tool calls
How do I buy from it?	UCP	The seller's commerce profile, catalogue search and checkout

Question	Standards (see Annex)	Used in this experiment
Who sent this request, from which running copy?	HTTP Message Signatures; SPIFFE and SPIRE	Every buyer request signed; one identity per running instance
Did a person approve this purchase, within what limits?	Passkeys; AP2; SD-JWT	The person's confirmation; two key-bound mandates
Can anyone rewrite what happened?	SCITT	Ledger checkpoints with receipts from a public log

2.1 Naming: DNS, DNSSEC and DANE

DNS delegates authority down a hierarchy. The root delegates .com, .com delegates skybound-travel.com to its registrant, and only that registrant can create names beneath it, such as seller.skybound-travel.com. The Service Binding (SVCB) record (RFC 9460) maps a name to a service, carrying the endpoint, port and protocol parameters in one cacheable answer.

Domain Name System Security Extension (RFC 4033 to 4035), or DNSSEC, adds digital signatures to DNS responses that can be validated via a chain of trust, so that anyone can confirm that an answer came from the domain's owner and was not altered on the way. DANE (RFC 6698) lets the domain's owner publish which certificate, or which public key, its server must present, in a record called TLSA within a DNSSEC signed zone. Because that record is contained within a DNSSEC signed zone, it is protected from modification in transit and can be verified. The seller's endpoint publishes two TLSA records. The buyer accepts the server when the certificate it presents matches either.

Together these DNS standards prove control of a domain. All three are IETF Protocol Standards.

2.2 Discovery: DNS-AID

DNS-AID, short for DNS for AI Discovery, is a proposal for publishing agents and organizational agent catalogues in DNS. The owner adds a record under their own domain saying where the agent lives, which protocol it speaks and where its description can be found, and an index record listing the agents the domain publishes. It is an IETF Internet-Draft that aims at the standards track.

The primary record sits at the agent's own name and signals the agent protocol in alpn, one agent protocol per record. An organisation-level index lives at `_index._agents.<domain>` that targets an organisation catalogue of agents. Records can be DNSSEC-signed, and must be when publishing TLSA record for the endpoint. The draft covers finding a known agent or browsing a known organisation, and leaves capability search to services built on top. DNS-AID helps agents find one another and leaves the question of whether to trust what they find to the checks in Section 3.2.

Figure 2. The seller's DNS-AID record that is DNSSEC-signed

```
seller.skybound-travel.com. 3600 IN SVCB 1 .
  alpn="h2,h3"
  port=443
  key65400="https://seller.skybound-travel.com/.well-known/agent-card.json"
  key65401="-om12VK0A9jnWZEBidCIYleAOrMcM5XNqr_wypX14To"
  key65402="a2a"
  key65405="eyJhbGciOiJFUzI1NiIsImtpZCI6InNlbGxhci0yMDI2LTewliwidHlwIjoizG5zLWFpZC1yZW50cmQrandzI
n0.eyJhbHBuljoiYTJhliwiZXhwIjoxNzkyMTc5MDk1LCJmcWRuljoic2VsbGVyLnNreWJvdW5kLXRyYXZlbC5jb20iL
CJpYXQiOjE3OTE5MTk4OTUsInBvcnQiOiJQ0Mywic3ZjYiI6IjIwYy1YeWVVKdkxNeEFabzQwa01oZGVrWFU2RiVm
blhIYIZkLVA1bUdLV0UiLCJ0YXJnZXQiOiJzZWxsZXluc2t5Ym91bmQtdHJhdmVsLmNvbSJ9.Sy5w0is9mLMWkklv
CGtPDmHULG04j1Fi0z-x5i8-VPJtBNRjnyfE4thZQXVj4Ukt8W1eJTeuXPzJS9YYex8blw"
  key65409="agent-card.json"
```

```
seller.skybound-travel.com. 3600 IN RRSIG SVCB 13 3 3600 (
  20261014213113 20261012193113 34505 skybound-travel.com.
  vH3nbRdq8OFURE7q83NJo757zyeUNQqAWCVdTdR4kmpu
  01s6HrsdCv0hvo2Nlr6k29U0r7i0bYfDVkhfOI7cjw== )
```

The buyer checked the record's signatures itself, against the zone's key and its delegation at the parent zone, and relied on the resolver's ad flag for the rest of the chain up to the root. It wrote that limit into its ledger entry rather than leaving it for a reader to discover.

2.3 Registration: ANS

ANS is a registry for agents and a proposal submitted as an IETF Internet-Draft. When an agent registers, the registry confirms that its operator controls the domain, issues identity certificates from its own certificate authority, and records the registration in a public, append-only log. The draft defines Bronze, Silver and Gold assurance tiers. ANS and DNS-AID share a record layer and the ANS reference implementation emits DNS-AID records by default. The registry used here is GoDaddy's production ANS registry, but anyone can implement their own ANS registration authority. Figure 3 shows the seller's registration as it appears in the log.

```
{
  "schemaVersion": "V1",
  "status": "ACTIVE",
  "payload": {
    "logId": "01a136bf-a5e3-7c69-b076-75b1cd01479f",
    "producer": {
      "event": {
        "ansId": "c3f2e9ae-320f-4244-83dd-8af52c2a73f8",
        "ansName": "ans://v1.0.9.seller.skybound-travel.com",
        "eventType": "AGENT_REGISTERED",
        "agent": {
          "host": "seller.skybound-travel.com",
          "name": "seller",
          "version": "v1.0.9"
        }
      },
      "attestations": {
        "dnsRecordsProvisioned": {
          "_443._tcp.seller.skybound-travel.com": "3 0 1
870f8f29f0dd14e29c830fea08f4505a0c58199e8c99b07cde96a59a78aef285",
          "_ans-badge.seller.skybound-travel.com": "v=ans-badge1; version=v1.0.9;
url=https://transparency.ans.godaddy.com/v1/agents/c3f2e9ae-320f-4244-83dd-8af52c2a73f8",
          "_ans.seller.skybound-travel.com": "v=ans1; version=v1.0.9; p=a2a; mode=direct;
url=https://seller.skybound-travel.com/a2a",
          "seller.skybound-travel.com": "0 . alpn=h2"
        },
        "domainValidation": "ACME-DNS-01",
        "identityCert": {
          "fingerprint":
"SHA256:6341aedace9e84b37e380d108f0a930dedaa40eb78b3c01dfd28b9154af5d167",
          "type": "X509-DV-CLIENT"
        },
        "serverCert": {
          "fingerprint":
"SHA256:870f8f29f0dd14e29c830fea08f4505a0c58199e8c99b07cde96a59a78aef285",
          "type": "X509-DV-SERVER"
        }
      },
      "expiresAt": "2027-01-05T21:35:02.000000Z",
      "issuedAt": "2026-10-13T19:31:10.527304Z",
      "raId": "raid-0d401ec535da03bdf4007192bb2af97ca8d9d88901b154f90fabb8c73ec469cc",
      "timestamp": "2026-10-13T19:31:31.925214Z"
    },
    "keyId": "23d2aa13",
    "signature":
"eyJhbGciOiJIUzI1NiIsImtpZCI6IjZDZDhYTEzliwcmFpZCI6InJhaWQtMGQ0MDFiYzUzNWZhMDNiZGY0MDA3MTkyYmlyYWYyN2NhOGQ5ZDg4OTAxYjE1NGY5MGZhYmI4YzczZWw0NjIjYyYsInRpbWVzdGFtcCI6MTc5MTkxOTg5MSwidHlwIjoiaSldUln0..TloPj99JA1EnMwTeU_8Jbf_at1zi1N8uOD9iNi2G2iC4leJxWeTiKk3naL3jBiWGgbACA KjiNHcJJoMW5RmGVJQ"
  }
}
```

9

2.4 Capability Discovery and Trust Attestations: ARD

ARD, short for Agentic Resource Discovery, is an open discovery protocol that operates across both static and dynamic discovery layers. For static discovery, an organisation publishes a manifest of its agents, MCP servers, skills, and APIs at a well-known location (`/.well-known/ard.json`), via DNS service binding records, or in web markup. For dynamic discovery, ARD defines a standard RESTful Registry Search API (POST `/search`, POST `/explore`) so registries can crawl, federate, and answer capability and natural-language queries. To support verifiable discovery, ARD supports an extensible trustManifest framework in which the manifest lists the publisher's identity and the cryptographic or compliance attestations anyone can check. In this experiment, the ARD manifest is published at <https://catalog.skybound-travel.com/.well-known/ard.json>. In this experiment the catalogue sits at the host the DNS-AID index points to.

2.5 Description: the A2A agent card and MCP

A2A, short for Agent2Agent, is now run by the Linux Foundation. It lets agents exchange messages and long-running tasks, and it defines the agent card: a short file at `/.well-known/agent-card.json` saying who runs an agent, where to reach it, which interfaces it speaks and what it can do. A2A 1.0 defines card signatures. The seller's card is written to be read by A2A. The card carries an A2A 1.0 JWS signature (EdDSA), whose verification key is published at the seller's trust card, named by the signature's `jku`. The buyer gets a second, independent binding from DNS: the card's fingerprint is published in the signed DNS record, so a tampered card fails the buyer's digest check even without fetching the signing key. The seller's card declares two interfaces, A2A and MCP, and two skills: flight search, which needs no authorisation, and checkout, which needs a mandate from an approval service the seller trusts. A card is the agent's own assertion and says nothing about whom the agent acts for.

MCP is an open protocol that lets AI applications use tools, each described by a name and the exact inputs it expects. It was contributed to the Agentic AI Foundation, part of the Linux Foundation. Here MCP is the doorway through which the person's assistant reaches the buyer agent, and the transport the buyer uses for the seller's catalogue search. Figure 4 and Figure 5 below show part of the seller's agent card and the ARD catalog.

Figure 4. Part of the seller agent's agent card at <https://seller.skybound-travel.com/.well-known/agent-card.json>

```
{
  "name": "seller",
  "description": "seller agent for Skybound Travel — A2A-first; MCP also supported",
  "provider": {
    "organization": "Skybound Travel",
    "url": "https://skybound-travel.com"
  },
  "version": "1.0.9",
  "capabilities": {
    "extensions": [
      {
        "description": "MCP server endpoint",
        "params": { "endpoint": "https://seller.skybound-travel.com/mcp/", "version": "1.0" },
        "uri": "https://modelcontextprotocol.io/ext/server/v1"
      },
      {
        "description": "ANS agent identity",
        "params": {
          "ansUri": "ans://v1.0.9.seller.skybound-travel.com",
          "jwksUri": "https://seller.skybound-travel.com/.well-known/jwks.json",
          "spiffe": "spiffe://seller.skybound-travel.com",
          "trustCard": "https://seller.skybound-travel.com/.well-known/ans/trust-card.json"
        },
        "uri": "https://ans.dev/ext/identity/v1"
      }
    ]
  },
  "skills": [
    {
      "id": "flight_search",
      "name": "Search flights",
      "description": "Products with a seller-signed quote JWS per variant (dev.agenthaven.quote). Open to anyone; no authorisation required.",
      "tags": ["seller", "ucp"]
    },
    {
      "id": "checkout",
      "name": "Book the flight",
      "description": "A2A + MCP. Ticket { ticket_id, quote_id, mandate_id, passenger_ans, flight, issued_at, scitt_receipt } Signed with dpop (dpop). Prerequisites: quote_id, mandate, dpop_proof.",
      "tags": ["seller", "a2a", "mcp"]
    }
  ],
  "signatures": [
    {
      "header": { "kid": "HtevsZBuER7mho80btalCxEfyaa97dxL8QeKZEgpN90" },
      "protected": "eyJhbGciOiJIJZERTQSIImprdiSI6Imh0dHBzOi8vc2VsbGVyLnNreWJvdW5kLXRyYXZibC5jb20vLndlbGwta25vd24vYW5zL3RydXN0LWNhcmQuanNvbiliImtpZCI6Ikh0ZXZzWkJ1RVl3bWhvODBidGFJQ3hFRnlhYTtk3ZHHMOFFIS1pFZ3BOOTAiLCJ0eXAiOiJhZ2VudC1jYXJkK2p3cyJ9",
      "signature": "LnF0ICxqXCcWBLYJhjc-6EZplpyMMg7sdLg37fZiDOclj8d3VODbFrYUdeKvUIYhwoaBms6Pc70HCxiG6rrKBg"
    }
  ],
  "supportedInterfaces": [
    { "protocolBinding": "JSONRPC", "protocolVersion": "1.0", "url": "https://seller.skybound-travel.com/a2a" },
    { "protocolBinding": "https://modelcontextprotocol.io/binding/http/v1", "protocolVersion": "1.0", "url": "https://seller.skybound-travel.com/mcp/" }
  ]
}
```

Figure 5. The seller's ARD catalog at <https://catalog.skybound-travel.com/.well-known/ard.json>

```
{
  "entries": [
    {
      "data": {
        "entries": [
          {
            "displayName": "seller",
            "identifier": "urn:air:skybound-travel.com:agents:seller",
            "mediaType": "application/a2a-agent-card+json",
            "url": "https://seller.skybound-travel.com/.well-known/agent-card.json"
          },
          {
            "displayName": "seller",
            "identifier": "urn:air:skybound-travel.com:agents:seller",
            "mediaType": "application/mcp-server-card+json",
            "url": "https://seller.skybound-travel.com/.well-known/mcp-server-card.json"
          }
        ]
      },
      "specVersion": "1.0"
    },
    {
      "description": "seller agent for Skybound Travel",
      "displayName": "seller",
      "identifier": "urn:air:skybound-travel.com:agents:seller",
      "mediaType": "application/ai-catalog+json",
      "metadata": {
        "agentHost": "seller.skybound-travel.com",
        "ansName": "ans://v1.0.9.seller.skybound-travel.com"
      },
      "publisher": {
        "displayName": "skybound-travel.com",
        "identifier": "skybound-travel.com",
        "identityType": "dns"
      },
      "trustManifest": {
        "identity": "urn:air:skybound-travel.com:agents:seller"
      },
      "version": "1.0.9"
    }
  ],
  "host": {
    "displayName": "seller.skybound-travel.com",
    "identifier": "seller.skybound-travel.com"
  },
  "specVersion": "1.0"
}
```

2.6 Commerce: UCP, the Universal Commerce Protocol

UCP is an open protocol for shopping between agents and businesses, published at ucp.dev, and has Google and Shopify among its participants. A business publishes a profile at [/.well-known/ucp](https://ucp.dev/.well-known/ucp) of what it supports, including checkout, payment handlers and the AP2 mandate extension; a buying platform publishes a profile of its own and names it on every request. Profiles declare capabilities by reverse-domain name, such as dev.ucp.shopping.checkout, each pointing to a published schema, and list signing keys as a JWK Set. UCP defines checkout as a session, carried over REST (the core binding), MCP or A2A. Once the AP2 mandate extension is negotiated, every checkout response must carry a seller signature over the checkout, and the seller must not complete without a mandate.

UCP is where the buyer learns two identity facts about the seller before it buys: which key signs everything the seller sends, and which approval services the seller accepts mandates from.

Figure 6. Part of the seller's UCP profile at <https://seller.skybound-travel.com/.well-known/ucp>

```
{
  "capabilities": { "guest": { "ceiling": 100000, "mandate_max_seconds": 300 } },
  "endpoints": {
    "rest": { "base": "https://seller.skybound-travel.com/ucp/v1", "test_rail_handler": "test_rail" }
  },
  "keys": [
    {
      "alg": "ES256", "crv": "P-256",
      "kid": "2_IqCL-EaUWqDvVsKzEtS5FYrz4b_IpsGTjnVQHoV1w",
      "kty": "EC", "use": "sig",
      "x": "uEH_2Ja0b9iklBeht3mQUqyhAZFoageAPX5Y4E5PyRk",
      "y": "j7Ql7f0j_aYY1IK_QImEBllmvRkDKCsbj52LPGPxb2l"
    }
  ],
  "ucp": {
    "capabilities": {
      "dev.ucp.common.payment.ap2_mandate": [
        {
          "config": {
            "dev.agenthaven.guest": { "ceiling_minor": 100000, "currency": "EUR", "mandate_max_seconds": 300 },
            "dev.agenthaven.mandate_forms": ["ap2-v0.2-delegate-chain"],
            "dev.agenthaven.mandate_issuers": [
              "https://authority.skybound-travel.com",
              "https://authority.webmesh.ai",
              "https://sealedby.dev"
            ]
          }
        }
      ],
      "extends": "dev.ucp.shopping.checkout",
      "version": "2026-08-25"
    }
  },
  "dev.ucp.shopping.catalog.search": [{ "version": "2026-08-25" }],
  "dev.ucp.shopping.checkout": [{ "version": "2026-08-25" }],
  "payment_handlers": {
    "com.skybound-travel.payment.test_rail": [
      { "credential_type": "ap2_mandate", "id": "test_rail", "version": "2026-08-25" }
    ]
  },
  "services": {
    "dev.ucp.shopping": [
      { "endpoint": "https://seller.skybound-travel.com/ucp/v1", "transport": "rest", "version": "2026-08-25" }
    ]
  },
  "version": "2026-08-25"
},
"ucp_version": "2026-08-25"
}
```

2.7 Authentication: HTTP Message Signatures and SPIFFE

HTTP Message Signatures (RFC 9421) let a sender sign selected parts of a request with a key the receiver can verify, and Content-Digest (RFC 9530) covers the body. UCP builds on them to authenticate a buying platform the seller has never met: the buyer names its profile on each request, the profile carries its keys, and the seller checks the signature against them. The buyer in this experiment signs every request this way, under the key in its own UCP profile.

A domain name tells a counterparty which organisation runs an agent. SPIFFE, a graduated CNCF project, goes a level further and gives a running workload an identity of its own, written like a web address that begins with `spiffe://`.

SPIRE is the software that issues those identities, after checking where the workload runs. In this set-up the seller's and the buyer's running instances each have their own SPIFFE identity, which is a choice made here; the standard does not require it. SPIRE issues the buyer's on its host. The seller runs on a serverless edge platform, so its operator issues the identity from outside the runtime, over a key the instance generated itself. The buyer publishes a key statement at `/.well-known/agent-keys.json`, signed under that identity, naming the key it signs with; when it retires a key it publishes that as well. The same key appears in the buyer's UCP profile and in the mandate.

2.8 Authority: passkeys, AP2 mandates and SD-JWT

A passkey is a cryptographic key kept by the person's device or password manager, defined by the FIDO Alliance and the W3C's Web Authentication standard. The fingerprint, face or PIN used to unlock the person's phone unlocks the passkey, which then signs a statement the website asked for. The approval service records how each approval was signed.

AP2, the Agent Payments Protocol, defines a mandate: a signed statement that a person has allowed an agent to buy something, within limits, until a set time. It comes in two forms. A Checkout Mandate is what the merchant checks, and a Payment Mandate is what the payment side checks. AP2 insists that approval happens on a surface that is not an agent, and that every check on a mandate runs in deterministic code. It treats the shopping agent as a potential attacker.

Mandates are built on SD-JWT (RFC 9901), an IETF format that holds only hashes of its fields, so the presenter reveals only what each verifier needs. Key binding places the agent's public key in the mandate and requires a short signed proof at each use. A mandate as issued is open: it says what may be bought, and by which key. The buyer closes it by adding a signed statement of its own, addressed to one seller and tied to one checkout session. A copied mandate is worthless without the key it is locked to. The seller accepts mandates in two formats issued from the same approval, this AP2 form and a compact `mandate+jwt` profile, and checks both against the same ten controls.

2.9 Records: SCITT and the Open Trust Index

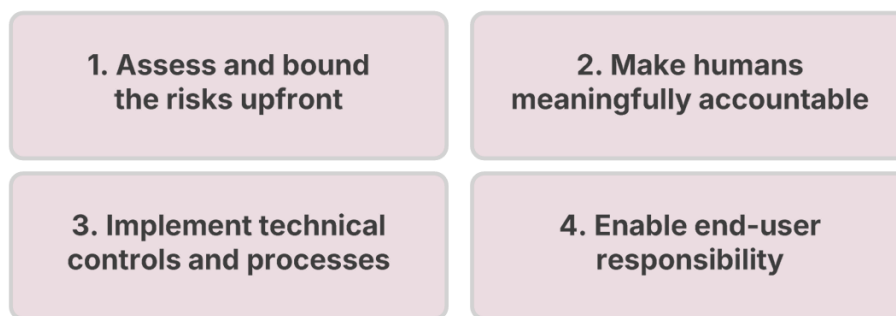
SCITT, Supply Chain Integrity, Transparency and Trust, is an IETF architecture (RFC 9943, June 2026) for transparency services: append-only logs where each registered statement receives a receipt that anyone can check later, and where statements can be added but never changed or removed. A log proves that a record existed unaltered at a point in time; it does not

vouch for the record's content. The public log used here is a SCITT service, whose COSE receipts carry a Merkle-tree inclusion and proof (RFC 6962). Anyone can compute a consistency proof between two of the log's signed checkpoints from the hash tiles the log publishes.

The open Trust Index, from the ANS project, gathers public signals about an agent, such as its certificates and DNS set-up, and turns them into a score. It answers how far to trust an agent, which is a separate question from who the agent is, and each organisation decides for itself what weight to give it. The registry's public search returns each agent's score, and the scoring engine is open source (github.com/agentnameservice/agent-trust-discovery).

3 Meeting IMDA's Agent Identity Requirements

[IMDA's Model AI Governance Framework for Agentic AI](#) (version 1.5, May 2026)(IMDA MGF) is organized around four dimensions: (1) assess and bound the risks upfront, (2) make humans meaningfully accountable, (3) implement technical controls and processes, and (4) enable end-user responsibility.



Four dimensions of the MGF for Agentic AI

This case study focuses on the first two dimensions, and specifically on agent identity, which runs through both.

3.1 Identity: unique, accounted for, differentiated and catalogued

Under dimension 1, the IMDA MGF asks organisations to give each agent an identity that is unique, accounted for, differentiated according to the capacity in which it acts, and catalogued and centrally managed.

Our experiment shows how open standards can meet each of these requirements where the agent on the other side belongs to a different organisation. Each agent in the experiment is named by a domain, and every identity check traces back to that name. See Table 5 for evidence in the experiment meeting IMDA MGF agent identity requirements.

Table 5 Evidence in the experiment meeting IMDA MGF agent identity requirements

IMDA MGF Requirement	Evidence in this experiment
Unique	Each agent has its own domain name, such as buyer.onbehalf.dev, with DNS records signed using DNSSEC, the security extension that shows the records were not tampered with. Each of the four zones publishes, with DANE, the public key that every server a verifier contact must present. The seller's and the buyer's running instances each hold their own SPIFFE identity, and every ledger entry names the instance that wrote it.
Accounted for	Each agent is tied to the owner of its domain, and ANS records each registration in a transparency log. Each purchase is tied to the person who approved it with a passkey, and the approval record names the assistant the request came through.
Differentiated according to capacity	The buyer agent searched on its own account, with no credential. It could buy only on the person's behalf, with their approval, under signed AP2 mandates. The ledger records both capacities, so the two acts can be told apart afterwards.
Catalogued and centrally managed	The seller is published under its own domain using DNS-AID and in its zone's ARD catalogue, and the seller and buyer are registered in ANS. The seller is discovered through DNS and accepts the approval service by its own published policy. The operator can remove or modify its agent records, and it can withdraw an agent's key or DNS record whenever it chooses.

By tying an agent's identity to a domain name, organisations inherit accountability mechanisms that already operate at internet scale, which answers the framework's ask that each agent be accounted for:

- **Registration.** Every domain has a registrant, who can be identified through established request processes.
- **Abuse handling.** Registrars act on abuse reports, and can suspend a domain used for harmful activity.
- **Dispute resolution.** Disputes over names, such as lookalike domains, are resolved through established procedures such as ICANN's Uniform Domain-Name Dispute-Resolution Policy (UDRP).
- **Protective DNS.** Organisations can use protective DNS services, as recommended in the Cyber Security Agency of Singapore's addendum on securing agentic AI systems (June 2026), to monitor the domains their agents look up and block connections to malicious domains before first contact.

Because only the registrant can create names beneath a domain, a counterparty can attribute a DNSSEC-signed record to the registrant and know a response has not been tampered with. The domain name outlives keys and platforms: keys rotate, agents are redeployed and hosting moves between providers, while the domain stays the same.

Before transacting, the buyer agent ran fourteen checks on the selected seller agent and wrote each result into its ledger, signed by its own instance. Read together, the checks answer four questions. Nine concern the seller's identity; the five ucp_ checks concern its commerce profile.

Table 6 Fourteen checks ran by buyer on seller before transacting

Question	Checks in the buyer's ledger	What each check confirms
Is this the record the domain's owner published?	record_validated, card_url_declared , record_signature, dane_key_host	The DNS record's signature is valid; the record names the card it should; the publisher's own signature over the record verifies; the key that verifies it is served from a host whose certificate is published in DNS.
Am I connecting to the owner's server, and not an impostor's?	dane_endpoint, endpoint_in_zone, seller_instance	The certificate presented by seller.skybound-travel.com carries the public key the owner published in DNS; the endpoint lies inside the seller's own zone; the running instance holds an identity the domain issued.
Is the description I fetched the one the owner meant?	card_digest	The agent card matches the fingerprint in DNS byte for byte.
Does anyone else vouch for this agent, and can I do business with it?	ans_registration, ucp_profile, ucp_negotiation, ucp_key, ucp_handler, ucp_issuers	The seller is registered in ANS, with a receipt in a public log; its commerce profile is well formed, shares capabilities with the buyer's, carries the same key the instance uses, offers a payment handler the buyer can use, and accepts the approval service the buyer's person uses.

Each check uses public records only. The buyer needed no shared directory or code with the seller. Anyone given access to the ledger can repeat every check from the domain name alone. Readers can repeat these checks against a live seller in the Open Agent Commerce lab, at <https://openagentcommerce.org/labs>.

3.2 Authority: scoped, least privilege, time-bound, non-transferable and bounded by the person

An agent's identity tells the seller who is asking. Authority tells it whether the request is allowed. Dimension 1 of the IMDA MGF asks each agent's authority to be scoped, least-privilege, time-bound and non-transferable, and bounded by the permissions of the person who authorises it.

Table 7 IMDA MGF agent authority requirements

IMDA Requirement	Evidence in this experiment
Scoped, least privilege, non-transferable	The Checkout Mandate covers only the seller and flight in the seller's signed UCP checkout; the Payment Mandate presets one payee and one amount. Both are valid for a bound time duration and work only with the buyer agent's own key. The buyer cannot sign mandates itself.
Bounded by the human's permissions	The approval service applied the person's own limits before signing, and the mandates carry only what the person approved. The limits a person can set include a per-purchase ceiling, period budgets, blocked sellers and a minimum seller trust score checked against the verifier's scorecard. Key milestones were recorded in a public transparency log that none of the four parties controls.

The seller, before agreeing to transact, ran its own sequence of checks shown in the following table.

Table 8 Checks run by seller before transacting

Check	What the seller verifies
1	Size, content type, method, protocol version
2	Signed request envelope: algorithm, key id, forbidden headers, 60-second window, nonce, audience, method, path and body hash
3	Buyer identity: attestation on the buyer's domain, SVID chain, not revoked
4	Mandate issuer is on the seller's allow-list; the signature verifies under the issuer's key set; the person and buyer match
5	Audience: the mandate is for this seller
6	Seller, action, stage, currency and amount equal the checkout total
7	The quote exists, has not expired, and its hash recomputes
8	Time window has not expired
9	Single use; the mandate key equals the key that signed the request
10	Not cancelled: status read live at the approval service

Checks 4 to 10 are where the framework's authority criteria are enforced. The mandate names one seller and one checkout (scoped), carries only what the person approved (least privilege and bounded by the person), expires within a set window (time-bound), and works only with the key it is bound to (non-transferable).

If any check from 2 to 10 fails, the seller answers with a typed refusal, such as `identity_revoked` at check 3 or `mandate_revoked` at check 10, and writes a signed refusal entry to its public ledger. Forged and tampered requests against the seller agent were refused, and none led to a payment.

To test these checks, the verifier ran several rogue agents against the live seller, sending forged and tampered requests. Every forged and tampered request was refused, and none led to a payment.

3.3 Withdrawing identity and revoking authority

People change their minds, and operators retire keys. A purchase can be stopped at four levels.

Table 9 Four levels to stop a transaction

Who	What
The person	Stops their agent on the approval service's panel, with the passkey or with a stop-only link. Every mandate still out is revoked, waiting requests are denied, and nothing new is issued.
The person	Cancels one approval on its own page, until its mandate expires. The seller reads the mandate's status live at check 10, so the next attempt is refused.
The operator	Withdraws the agent's key, by publishing a signed withdrawal on the agent's own domain. The seller refuses that identity at check 3, and a mandate bound to the key can no longer be used.
The operator	Removes the agent's DNS record. Once cached copies expire, the agent can no longer be found.

The last two levels are the identity controls. Both act at the domain that owns the agent, so the operator can withdraw an identity without asking any registry or counterparty.

3.4 Auditability: A record each side can hold the other to

Each of the buyer agent, seller agent and the mandate authority that took part in the purchase keeps a public, hash-chained ledger of what it did, including the refusals.

After the purchase, each service signed a checkpoint of its ledger, the height and the hash of its latest entry, and registered it in two places: the verifier's witness and a public SCITT log run by a third party, which returned a receipt with a proof of inclusion. From that moment nobody, the buyer agent included, could rewrite the history of this purchase without the change showing up for anyone who compared the ledger with the registered checkpoints.

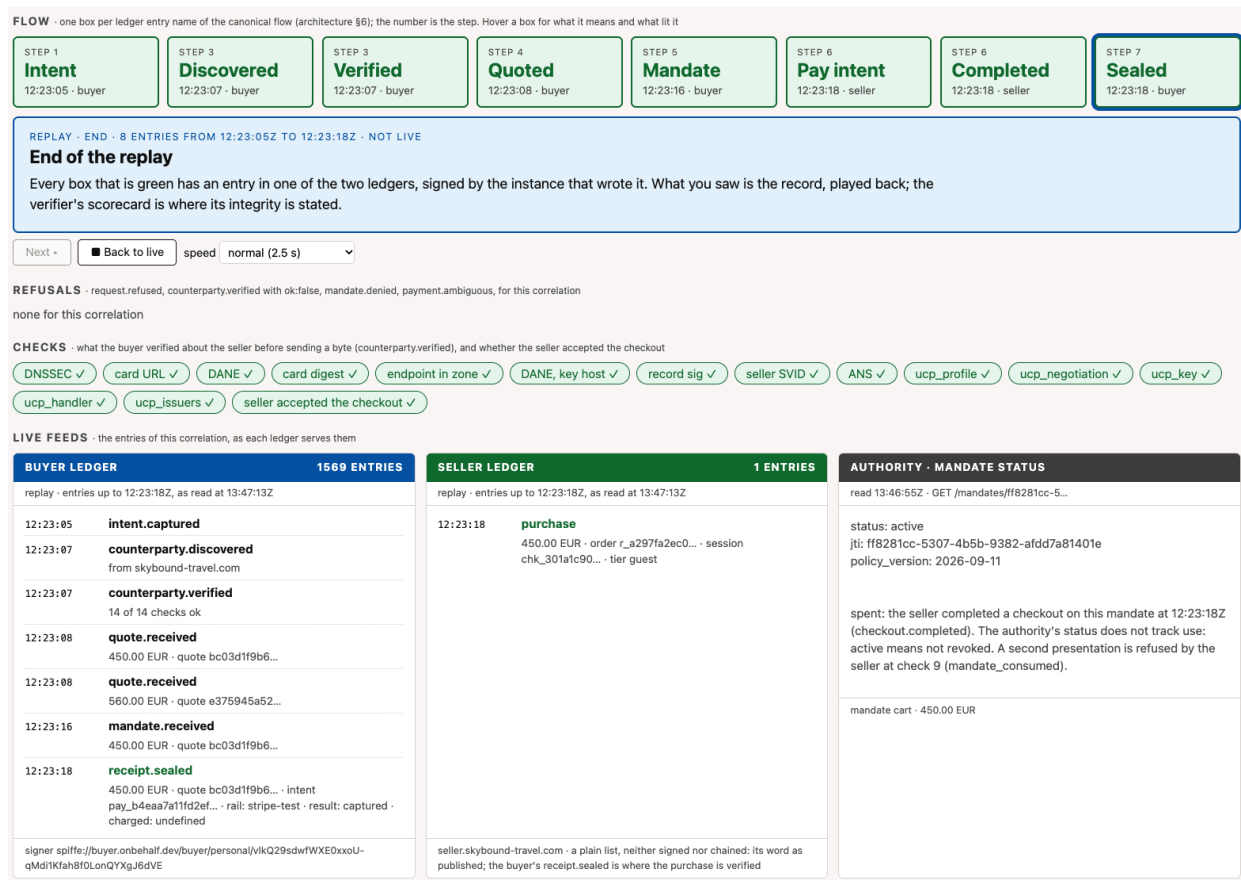
Following the transaction, the verifier gathered the public records of the three services. For this purchase it found seven entries at the buyer and four at the seller, with matching fingerprints and the same payment reference on both sides, and one entry at the approval service naming the same mandate. It asked the buyer and the seller separately for the mandate chains they had exchanged, received identical copies, and ran them through the AP2 reference software.

The verifier can replay these checks from three domain names, and publishes a signed scorecard with the raw evidence it captured. The scorecard examines nine dimensions, among them discovery, publisher control, registered identity, instance identity, mandate, attribution, revocation and rogue requests. The three ledgers, the checkpoints in a public log and the verifier's signed scorecard are the record that each side can hold the other to, and that a third party independent verifier can examine.

For illustration, we publish the ledgers that the verifier sees in this purchase on a live public dashboard at <https://provedby.dev/monitor.html>. It shows each step of the most recent purchase as the corresponding ledger entry appears, along with any refusals, the buyer's checks on the seller and the status of the mandate. See Figure 7 below.

The ledgers are made public here for illustration. In a commercial deployment, the parties are expected to keep the ledgers private and grant read access only to the verifiers they choose. Payments in the experiment ran in the payment provider's test mode, so the records show the full flow without moving real funds.

Figure 7: Live dashboard of the checks witnessed by the verifier



4 Conclusion

This technical paper contributed to the IMDA's Model AI Governance Framework for Agentic AI by demonstrating that AI agents from different organisations can verify each other's identity and authority with no prior arrangement between them. The agents need no new platform and no directory that every party has to trust. The open internet standards that the internet already runs at scale can be leveraged to identify and authenticate agents and hold humans behind them accountable.

Agentic commerce is at an early stage. The experiment is a reference architecture to demonstrate interoperability is achievable among agents built across organisations and platforms. We offer this reference architecture as a starting point, and invite regulators, standards bodies and builders to test it and help the standards it relies on mature.

Annex: Standards Adopted in Case Study

Standard	Most recent version and link	Body or maintainer
SVCB (Service Binding and Parameter Specification via the DNS)	RFC 9460, November 2023 https://www.rfc-editor.org/rfc/rfc9460	IETF
DNSSEC (Domain Name System Security Extensions)	RFC 4033, RFC 4034, RFC 4035, March 2005 (RFC 9364, BCP 237) https://www.rfc-editor.org/rfc/rfc9364	IETF
DANE (DNS-Based Authentication of Named Entities (TLSA records))	RFC 6698, August 2012 https://www.rfc-editor.org/rfc/rfc6698	IETF
DNS-AID (DNS for AI Discovery)	draft-mozleywilliams-dnsop-dnsaid-02, 27 May 2026 (draft) https://datatracker.ietf.org/doc/draft-mozleywilliams-dnsop-dnsaid/	IETF
ANS (Agent Name Service)	draft-narajala-courtney-ansv2-01, 13 Apr 2026 (draft) https://datatracker.ietf.org/doc/draft-narajala-courtney-ansv2/	IETF
ARD (Agentic Resource Discovery)	v0.91 https://github.com/ards-project/ard-spec	Community specification
A2A (Agent2Agent Protocol)	v1.0.1, 28 May 2026 (deployment serves 0.3) https://github.com/a2aproject/A2A/releases	Linux Foundation
MCP (Model Context Protocol)	2026-07-28 (deployment runs 2025-06-18 and 2025-03-26) https://modelcontextprotocol.io/specification/2026-07-28	Agentic AI Foundation
UCP (Universal Commerce Protocol)	v2026-08-25 https://github.com/Universal-Commerce-Protocol/ucp/releases	UCP Governing Council
HTTP Message Signatures; Digest Fields (Content-Digest)	RFC 9421, February 2024; RFC 9530, February 2024 https://www.rfc-editor.org/rfc/rfc9421	IETF
SPIFFE (Secure Production Identity Framework for Everyone)	SPIFFE standards (current main branch) https://github.com/spiffe/spiffe/tree/main/standards	CNCF
Passkeys (Web Authentication (WebAuthn))	Web Authentication Level 3 https://www.w3.org/TR/webauthn-3/	W3C and FIDO Alliance
AP2 (Agent Payments Protocol)	v0.2.0, 28 Apr 2026 https://github.com/google-agentic-commerce/AP2/releases	FIDO Alliance

Standard	Most recent version and link	Body or maintainer
SD-JWT (Selective Disclosure for JSON Web Tokens)	RFC 9901, November 2025 https://www.rfc-editor.org/rfc/rfc9901	IETF
SCITT (Supply Chain Integrity, Transparency and Trust (architecture))	RFC 9943, June 2026 https://www.rfc-editor.org/rfc/rfc9943	IETF
Certificate Transparency (Merkle-tree proofs)	RFC 6962, June 2013 https://www.rfc-editor.org/rfc/rfc6962	IETF