100% Autonomous Software Procurement

Deconstructing an AI agent's successful execution of the Agentic Commerce Protocol (ACP).

Execution Date: 2026-06-01 | **Environment:** ods-qa | **Engine:** GLM-5.1

The machine began with a domain and a target, but no map.

Constraints

No browser UI

No human-in-the-loop

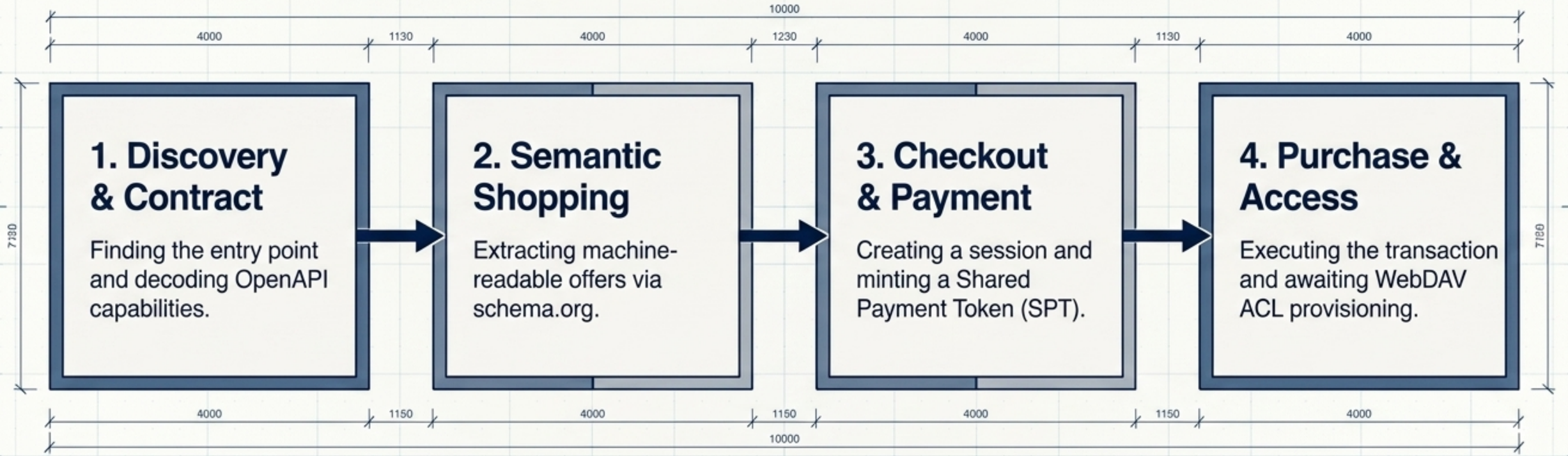
No API documentation

Starting Parameters

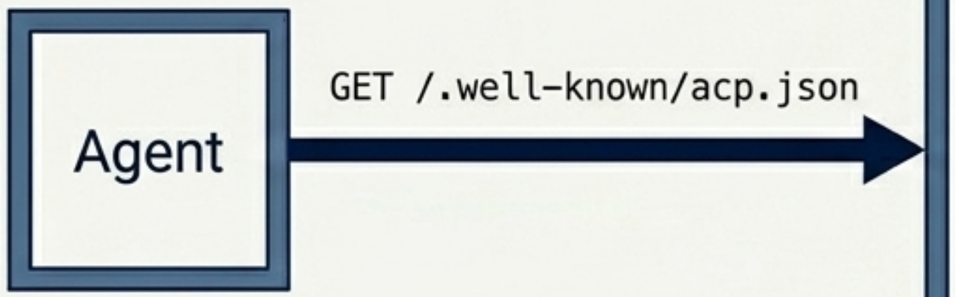
Target Domain	`https://ods-qa.openlinksw.com`
Objective Item	`Test File Access license (1 Year, $9.99)`
Authorisation	`Bearer Token [REDACTED]`
Payment Rail	`Stripe Test Key sk_test_...`

The core constraint: The agent must discover the endpoint, learn the contract, parse semantic offers, and pay via tokenised credentials entirely autonomously.

The Agentic Commerce Protocol (ACP) Execution Arc



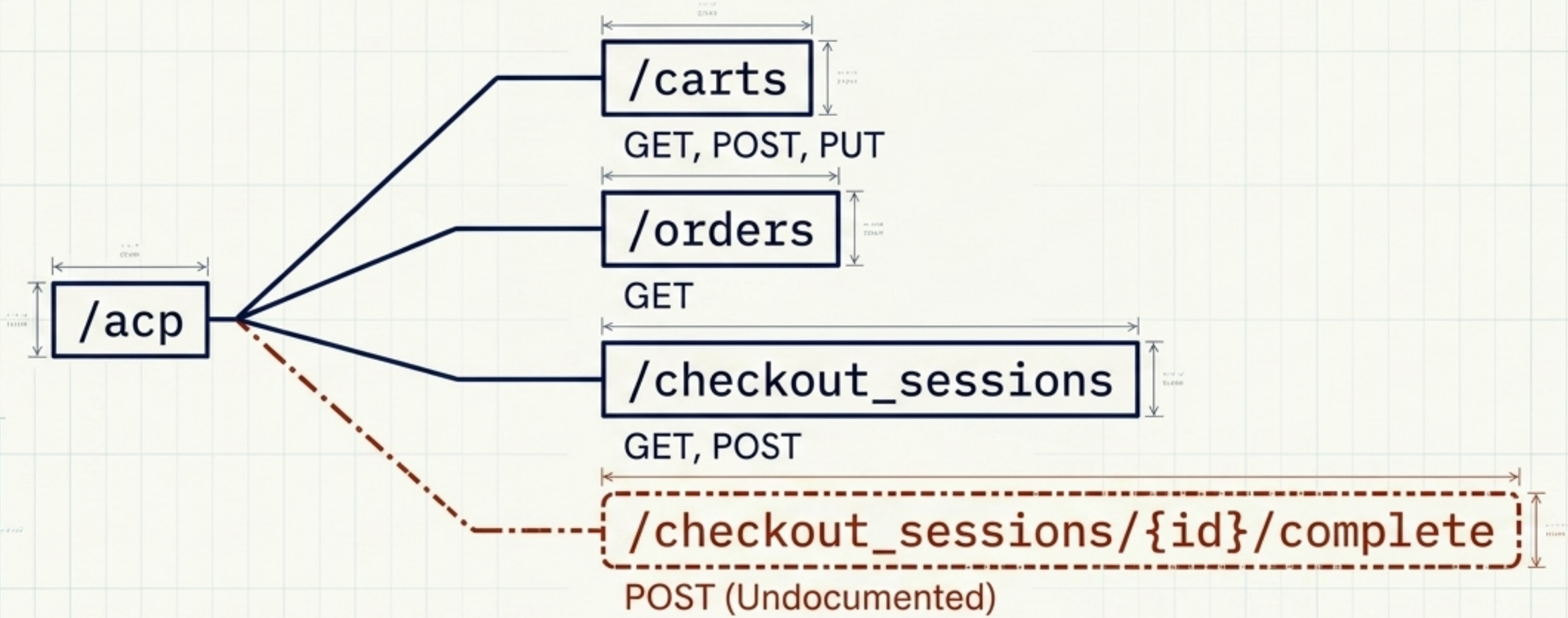
Endpoint autodiscovery starts at the .well-known boundary.



The Protocol Envelope Response

protocol.name	acp	Confirms target identity
protocol.version	2026-01-30	Critical for later mutation headers
api_base_url	/acp	The root for all subsequent API calls
capabilities.services	['checkout', 'orders', 'carts']	Available resources

Fetching the contract reveals standard resources—and one hidden endpoint

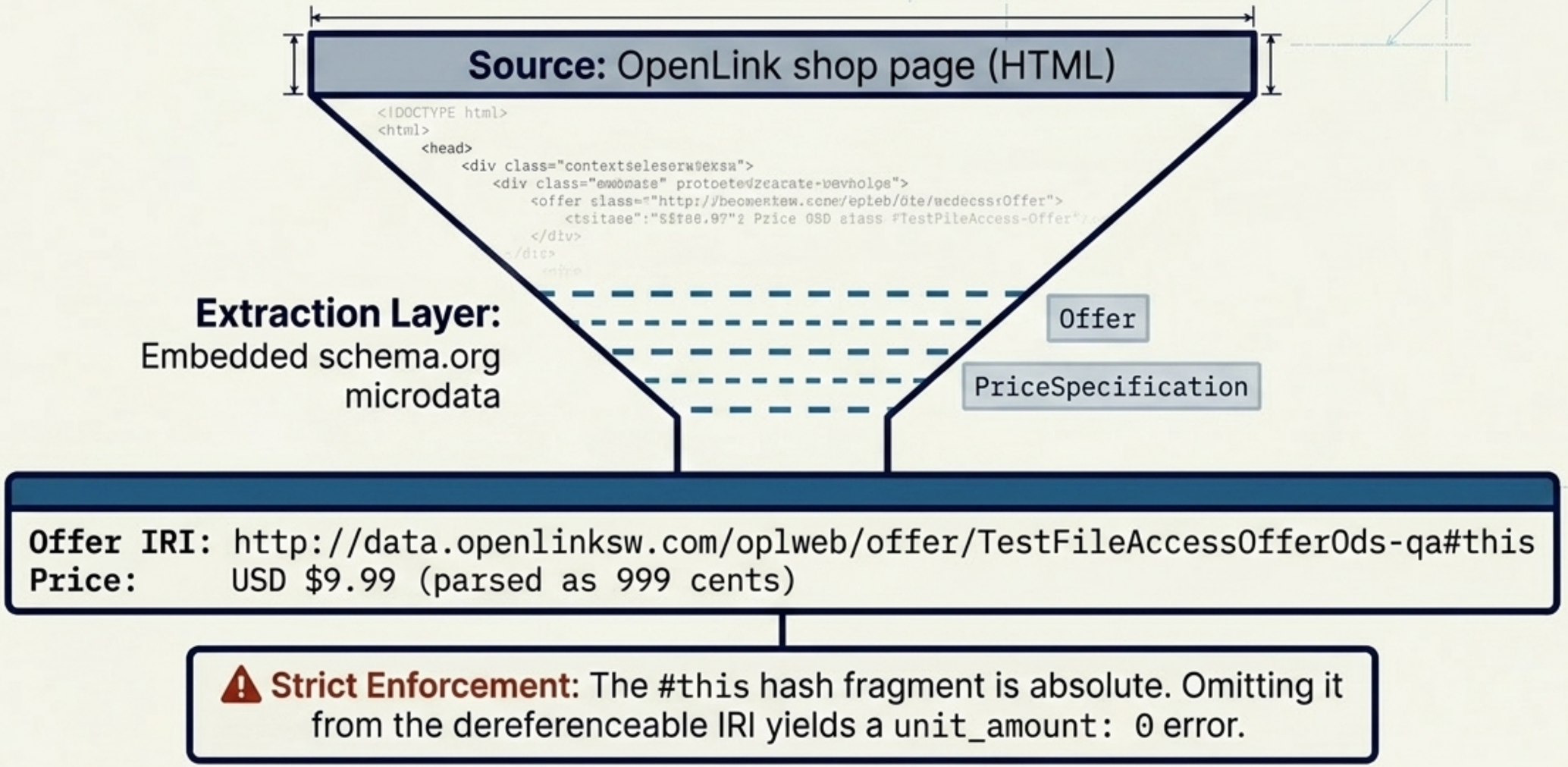


The OpenAPI 3.0.0 specification declares standard REST capabilities and OAuth2 Bearer token auth. However, the crucial /complete sub-resource is absent from the spec and must be discovered via runtime exploration.

Aligning the request headers requires endpoint-specific logic.

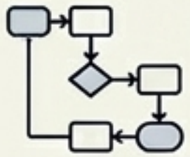
Standard Operations (/carts, /checkout_sessions)	**Mutation Operations** (/checkout_sessions/{id}/complete)
Authentication: `Authorization: Bearer <token>`	Authentication: `Authorization: Bearer <token>`
Content Type: `Content-Type: application/json`	Content Type: `Content-Type: application/json`
Versioning: `API-Version: 1.0`	**Versioning:** **`API-Version: 2026-01-30`** **(Matches protocol version)**
Traceability: `Idempotency-Key` **(for safe retries)**	**Traceability:** **`Idempotency-Key`** **AND** **`Request-Id`** **(UUID for deduplication)**

Semantic resolution turns human web pages into machine-actionable offers.



Carts are optional staging grounds; agents use direct checkout sessions.

Human-in-the-Loop The Cart API (/carts)	**Autonomous Agent** The Session API (/checkout_sessions)
- Designed for staging items. - Presents UI for manual user review. - Multi-step validation process. - **Conclusion:** Wholly optional in ACP.	- The direct agentic path. - Takes the resolved Offer IRI directly. - Backend automatically creates a backing cart. - **Conclusion:** Zero explicit /carts API calls required.



Architectural Conclusion: An agent can proceed directly from offer discovery to **`POST /checkout_sessions`**.

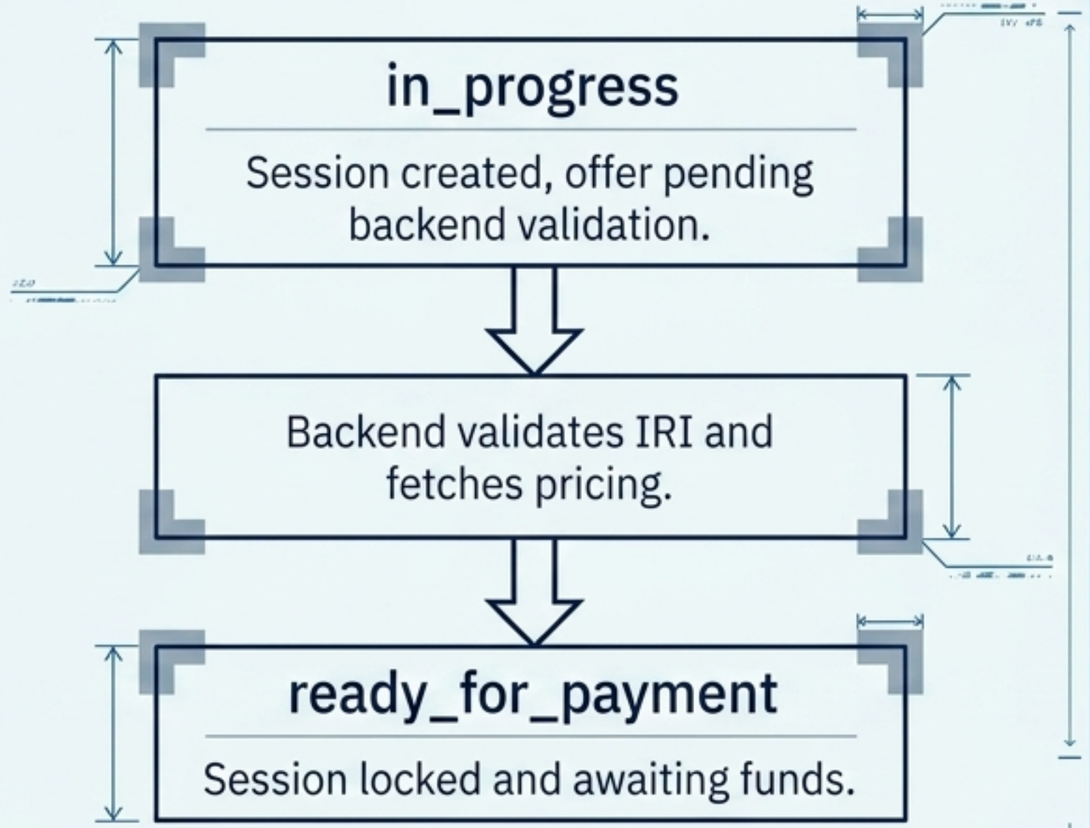
Generating the checkout session validates the semantic offer.

The Final Payload

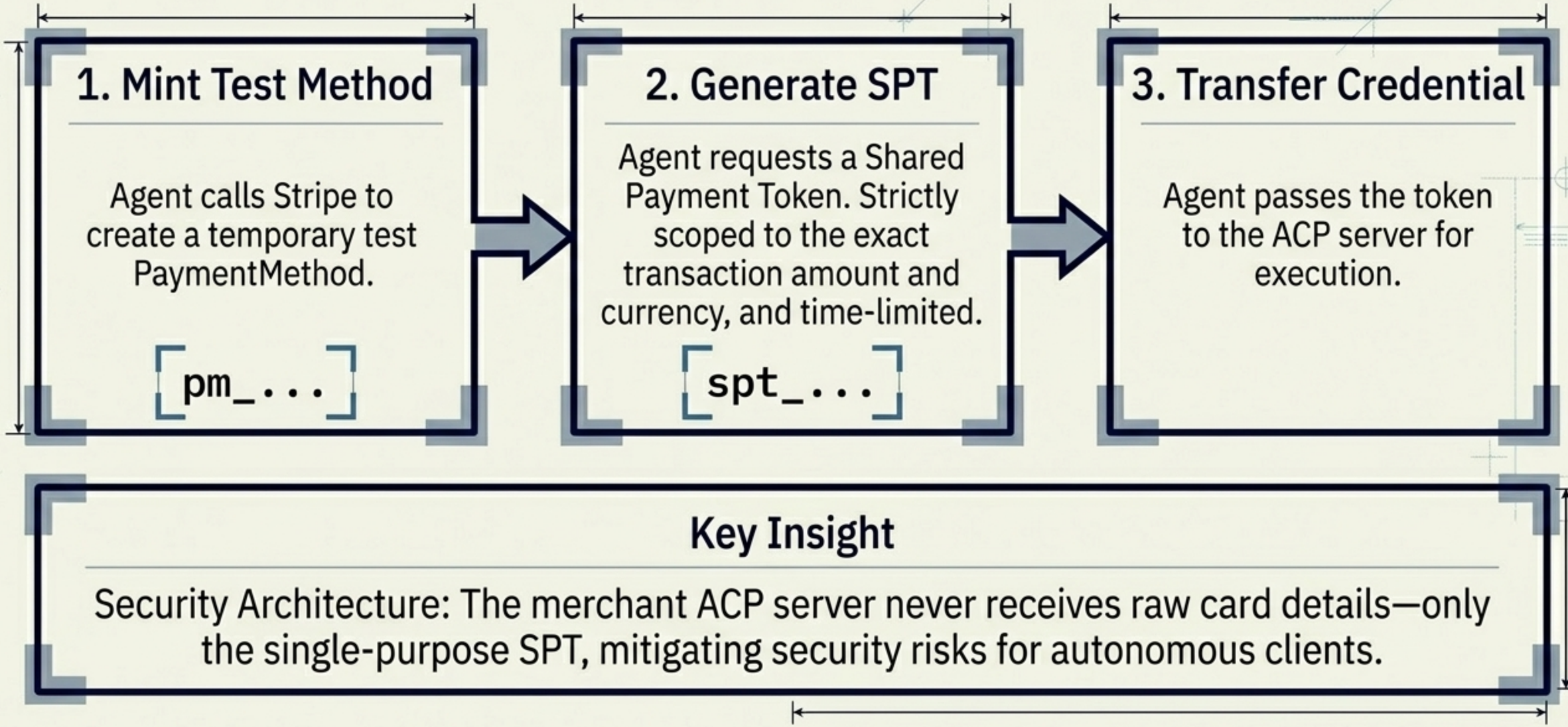
`POST /checkout_sessions` requires three critical keys:

```
{
  "line_items": "[Offer IRI]",
  "currency": "'usd'",
  "capabilities": "['payment']"
}
```

State Transition



Tokenising payment protects card data via strict scoping.



Key Insight

Security Architecture: The merchant ACP server never receives raw card details—only the single-purpose SPT, mitigating security risks for autonomous clients.

Firing the sub-resource finalises the autonomous transaction.

POST Request

Target: `/acp/checkout_sessions/{id}/complete`

Headers:
- `API-Version: 2026-01-30`
- `Request-Id: <UUID>`

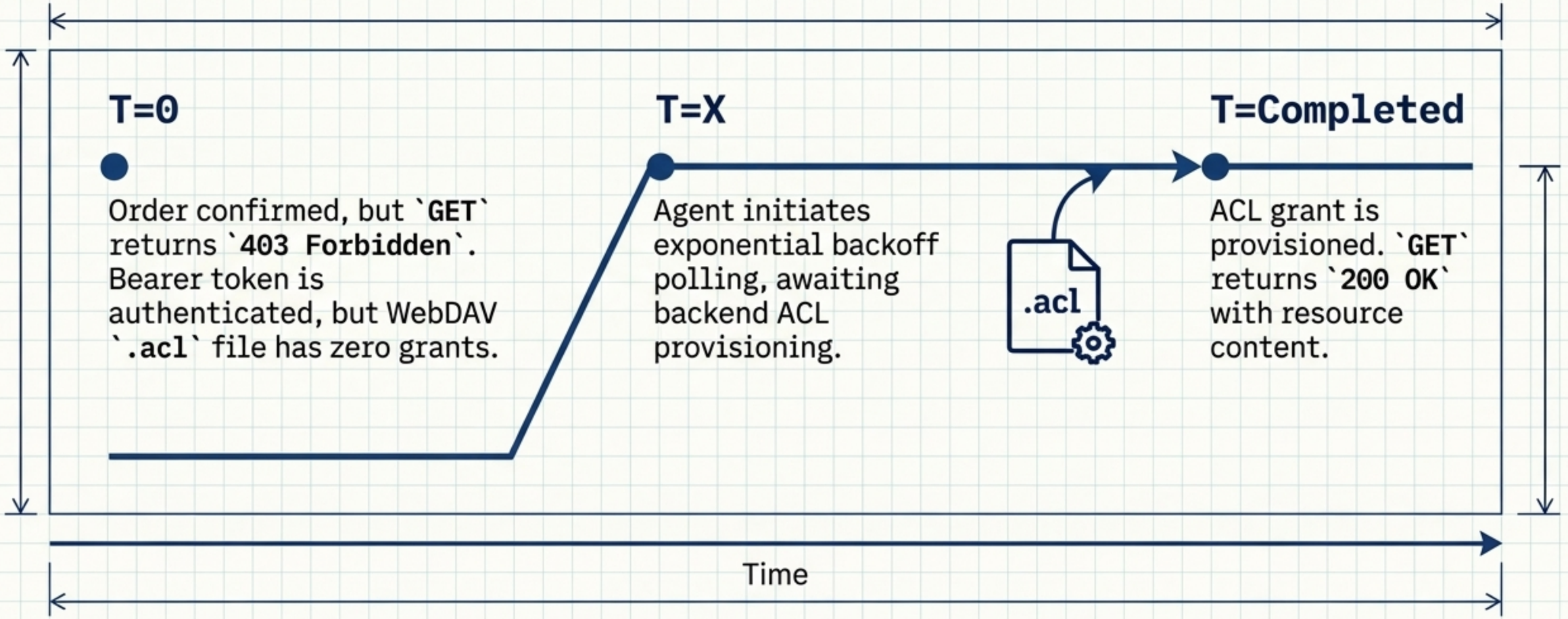
Payload:

```
{
  "payment_data.handler_id": "card_tokenized",
  "payment_data.instrument.credential.token": "spt_…"
}
```

200 OK Response

- Status shifts to `'completed'`.
- Embedded order object returns status: `'confirmed'`.
- Yields target License IRI for Test File Access.

Agents must bridge the asynchronous access gap via polling



Handling protocol edge cases: One-Time vs. Subscription logic.

One-Time Purchase		Subscription Trigger	
Target Offer:	`TestFileAccess`	Target Offer:	`XYZFileAccess`
Order Status:	`confirmed`	Order Status:	`confirmed`
Messages Array:	`Empty`	Messages Array:	`subscription_payment_required`
Links Array:	`Empty`	Links Array:	`rel='subscription_payment'` (Contains Stripe Invoice URL)
Next Steps:	`Poll resource for WebDAV access.`	Next Steps:	Present URL for browser-based activation/3DS challenge.

The Agentic Catalog: 19 structured OPAL offers on ods-qa.

Band 1: **OPAL Modules (Data Twingler & Support Assistant)**
Tiers: Entry (Free) to Max ($49.99)
Band 2: **API & Chat Services (BYOK)**
Tiers: API Access & Bring Your Own Key chat services.
Band 3: **File & Graph Access**
Includes `TestFileAccess` and `DemoGraphAccess` testing grounds.

Note: All catalog items exist as dereferenceable Linked Data resources terminating in the absolute **#this** fragment.

The Autonomous Receipt: A tokenised ledger of a human-free transaction.

The Analog Relic

The Agentic Receipt

Order Record:	`ord_... (status: confirmed)`
Financials:	`USD $9.99 (Stripe SPT processed)`
Semantic Offer:	`TestFileAccessOfferOds-qa#this`
License Minted:	`.../TestFileAccessLicenseOds-qa#this`
Resource Granted:	`https://ods-qa.../DAV/data/ TestFile.txt (WebDAV ACL)`

The **Agentic Commerce Protocol** enables **fully sovereign machine-to-machine** procurement through **semantic data, strict capability constraints,** and **tokenised trust.**