SECUREAUTH
Welcome to Better Identity.

HOT TOPIC

AGENTICAI

AUTOMATED TRUST: OPENID FEDERATION & CDR-STYLE ACCREDITATION

Manual trust doesn't scale. This piece introduces OpenID Connect Federation 1.0 and CDR-style accreditation as the next evolution in securing agentic AI—enabling dynamic, policy-driven trust between identity domains without custom integrations or pre-registration bottlenecks. It's the blueprint for automated, federated identity in cross-boundary AI ecosystems.

ARCHITECTING IDENTITY FOR AGENTIC AI: PART 5

This is **Part 5** in the series **Architecting Identity for Agentic AI**, a technical guide for software architects securing AI systems with OAuth, OIDC, and trust frameworks. Designed for architects building scalable, policy-driven identity across enterprise and cross-boundary environments.

BY LUKASZ RADOSZ

SVP of Engineering, SecureAuth

With over two decades in identity and access management, Lukasz brings deep expertise in authentication, authorization, and API security to the SecureAuth team. A champion of open standards like OAuth and OIDC, he's an advocate for modern identity architectures across machine identity, open banking, and transactional access control.

Introduction

In modern **agentic AI systems**, autonomous agents often need to access data or services across organizational boundaries. Establishing mutual trust between an enterprise and an external AI or SaaS provider is critical, especially as frameworks like the **Model Context Protocol (MCP)** emerge to connect AI assistants with enterprise data sources. (MCP is an open standard enabling secure, two-way connections between data sources and AI-powered tools.) In **Part 4** of this series, we covered Dynamic Client Registration (DCR) and trust registries as a semi-automated way to onboard trusted clients. Now in **Part 5**, we evolve to **OpenID Federation 1.0**, a specification designed for fully automated, policy-governed trust establishment at scale.

OpenID Connect Federation 1.0 (OIDF 1.0) provides a standardized mechanism for parties with **no prior direct relationship** to dynamically establish trust based on digital signatures and agreed trust anchors. In essence, it allows an Identity Provider (IdP) and a Relying Party (RP) to accept each other's identity assertions **without manual pre-registration**, by leveraging **signed federation metadata** and **trust chains**. This approach is well-suited to **agentic AI** and **MCP** integrations, where an enterprise IdP, AI service, and other SaaS components must rapidly form trust relationships under consistent policies. In this article, we provide a primer on OpenID Federation 1.0's key innovations—from **entity statements** to **trust chains**—and illustrate how **CDR-style accreditation** (as seen in Open Banking/Consumer Data Right) can be implemented through federation policy. We will then explore practical use cases: onboarding an external AI agent via federation, and federating enterprise IdPs for AI delegation across domains.

OpenID Federation 1.0 Primer

OpenID Federation 1.0 extends OpenID Connect by introducing a **federation layer** for automating trust. The core idea is that every party (whether an IdP/Authorization Server, an RP/Client, or an intermediary authority) is a **Federation Entity** with its own signed metadata. Key innovations include:

Federation Entity & Metadata
Each entity publishes a JSON metadata document, called an Entity Configuration, at a well-known URL (e.g. https://entity.example.com/.well-known/openid-federation). This contains its identity (issuer URL), role (IdP, RP, etc.), cryptographic keys (JWKS), and other OIDC metadata, all self-signed by that entity's private key. Because the entity signs its own configuration, it can be verified against the entity's public key (included in the metadata as jwks), forming a base level of authenticity.

Entity Statements (Signed Metadata)
Trust in OpenID Federation is established by having trusted authorities digitally **sign the metadata of other entities**. An **Entity Statement** is a JWT in which an **issuer** (an authority) asserts the metadata of a subject entity. For example, a federation operator or trust authority can issue a signed statement vouching for a client's metadata. These statements are chained together to build trust.

Trust Chains
A **trust chain** is a sequence of entity statements (JWTs) linking a **leaf entity** (the party we want to trust, such as an RP or OP) to a **Trust Anchor** (a top-level authority everyone in the federation trusts). The leaf entity's own self-signed configuration forms one link, intermediate authorities provide middle links, and finally a trust anchor provides the final link. Conceptually, this is analogous to a certificate chain in PKI (with trust anchors and intermediates similar to root and intermediate CAs). When the chain is cryptographically valid and the trust anchor is recognized, the two parties can trust each other's credentials. Notably, trust chains are uni-directional: an RP validating an OP's chain is one direction, and the OP validating the RP is another. For mutual trust, **two trust chains** (one in each direction) are needed, ensuring both IdP and client are accredited under a common framework.

Automatic Client Onboarding
Federation metadata includes provisions to automate client registration. For instance, an IdP's metadata can indicate support for **automatic registration** of clients (via a **client_registration_types** field set to "automatic"). If an RP presents a valid trust chain, the IdP can accept it and issue tokens without any manual client setup. Similarly, an RP can automatically accept an IdP's identity tokens if the IdP's trust chain is valid. This eliminates the cumbersome exchange of static credentials or out-of-band registration, which is vital in dynamic AI-to-service interactions.

In summary, OIDF 1.0 creates a framework where trust is derived from **cryptographic assertions and policies** rather than prior agreements or hand-configured registries. Next, we delve into how these trust chains are discovered and verified in practice.

Trust Chain Discovery & Validation

Establishing trust via OIDF is all about discovering and validating the **trust chain** between entities. Let's walk through how a trust chain is constructed and validated when, for example, an RP (AI client) encounters a new OP (enterprise IdP) it wishes to use:

1. Entity Configuration Fetch
The RP starts by retrieving the OP's entity configuration from its well-known URL (appending **/.well-known/openid-federation** to the OP's issuer URL). This is a self-signed JWT from the OP containing its metadata and keys. Because it's self-signed, it proves the OP controls that issuer URL and corresponding keys, but not yet that it's trusted by any third party.

2. Read Authority Hints
Inside the OP's entity configuration, the **authority_hints** claim lists one or more candidate authorities that vouch for this OP. Think of these as pointers to the OP's "parent" in a trust hierarchy (or members of a trust mesh). An authority can be a federation operator, trust framework authority, or any entity that has issued a statement about this OP. For example, an OP might list a trust federation URL or an accreditation authority's entity ID in its **authority_hints** array, indicating who to ask about its trust status.

3. Fetch Entity Statement from Authority
The RP then contacts each hinted authority's federation fetch endpoint to retrieve an entity statement about the OP. (The authority's fetch endpoint URL is published in its own entity configuration.) Essentially, the RP asks, "Do you have a signed statement for this subject (the OP)?" If the OP is indeed registered with that authority, the authority returns a JWT: an entity statement asserting the OP's metadata, signed by the authority. This step is repeated iteratively: the RP may now have a statement from an intermediate authority, which itself might have an **authority_hints** pointing to a higher authority.

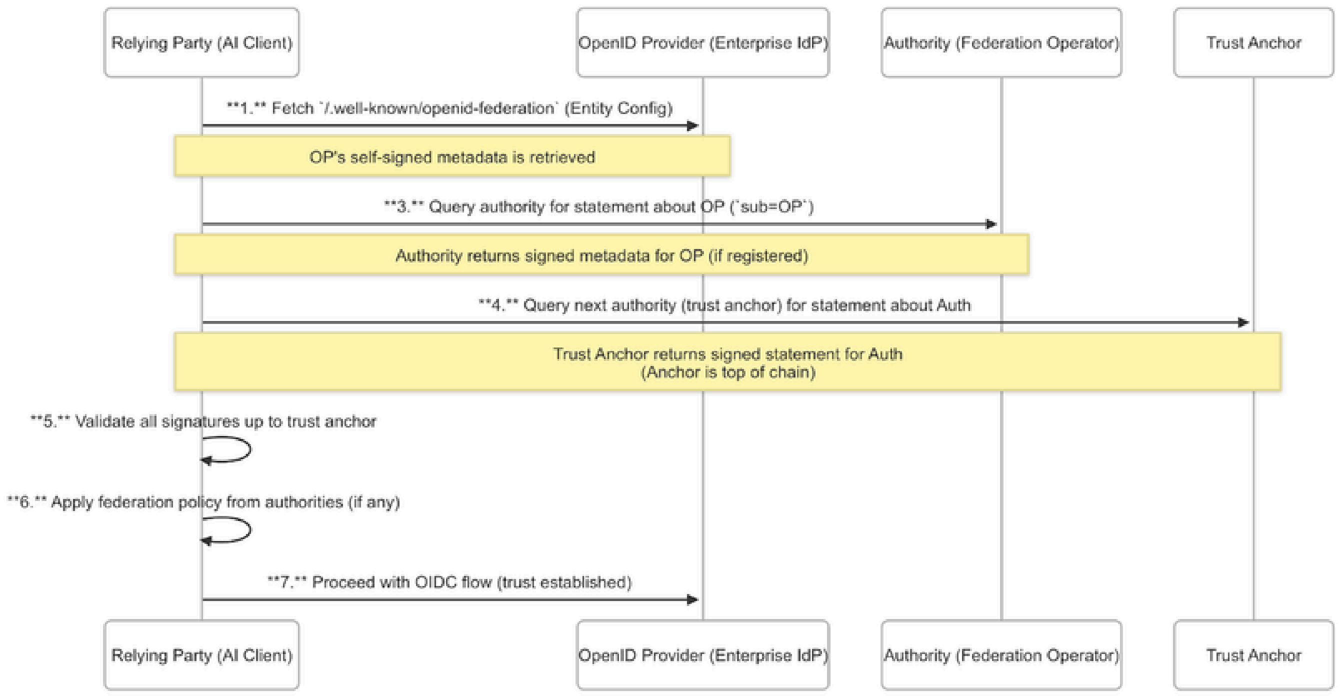
4. Iterate Up to Trust Anchor
Federation metadata includes provisions to automate client registration. For instance, an IdP's metadata can indicate support for **automatic registration** of clients (via a **client_registration_types** field set to "automatic"). If an RP presents a valid trust chain, the IdP can accept it and issue tokens without any manual client setup. Similarly, an RP can automatically accept an IdP's identity tokens if the IdP's trust chain is valid. This eliminates the cumbersome exchange of static credentials or out-of-band registration, which is vital in dynamic AI-to-service interactions.

5. Validate Signatures and Metadata
Now the RP validates the chain. This involves checking that each JWT in the chain is properly signed by the issuer's key, that each link's subject matches the next link's issuer (forming a continuous chain), and crucially that the final link's issuer is a trust anchor the RP recognizes. If any signature is invalid or the chain is broken, trust fails. If all signatures check out and the top issuer is trusted, the OP's metadata is considered verified. At this point, the RP has confidence that the OP is an **authorized member** of a known federation and that the metadata (including keys, endpoints, etc.) can be trusted.

6. Apply Policy (if any)
During chain validation, the RP (and likewise an OP in reverse) also applies any **metadata policy** that the authorities have defined. OIDF 1.0 allows authorities to impose a **metadata_policy** in their statements—rules that constrain or adjust the subordinate's metadata. For instance, a trust anchor might require that all tokens use specific algorithms or that certain claims be present. The RP will combine any policies from the trust anchor and intermediates and apply them to the OP's metadata before using it. The effective result is a metadata set that is guaranteed to conform to the federation's trust framework requirements.

7. Establish Trust & Use Metadata
After successful validation and policy application, the RP deems the OP trustworthy. The RP can now use the OP's metadata (e.g. issuer URL, JWKS keys, endpoints) to interact. For example, it can initiate an OIDC flow to this OP, confident that it's dealing with a legitimate provider. Because the OP's metadata likely advertised automatic client registration, the RP can even start sending authorization requests immediately—the OP will dynamically accept the RP because the RP itself will present a trust chain in its requests (the reverse direction).

The diagram below illustrates this **trust chain resolution** process in sequence form:


Relying Party (AI Client)
OpenID Provider (Enterprise IdP)
Authority (Federation Operator)
Trust Anchor
1. Fetch `/.well-known/openid-federation` (Entity Config)
OP's self-signed metadata is retrieved
3. Query authority for statement about OP (`sub=OP`)
Authority returns signed metadata for OP (if registered)
4. Query next authority (trust anchor) for statement about Auth
Trust Anchor returns signed statement for Auth (Anchor is top of chain)
5. Validate all signatures up to trust anchor
6. Apply federation policy from authorities (if any)
7. Proceed with OIDC flow (trust established)
Relying Party (AI Client)
OpenID Provider (Enterprise IdP)
Authority (Federation Operator)
Trust Anchor

In practice, OpenID Federation software can automate this chain discovery so that the RP only needs to know its trust anchors. A similar process happens in reverse when the OP needs to verify the RP's trust chain (for the OP to accept the incoming client). Once both sides have validated each other (finding a common trust anchor in their respective chains), they have established mutual trust **on the fly**, without any prior direct integration.

Federation Trust Models: Hierarchical vs. Mesh Authorities

The above example assumes a simple hierarchy (one intermediate authority and a trust anchor). OIDF 1.0 is flexible to support different trust models:

Hierarchical Trust

This is analogous to classic PKI: a single Trust Anchor (or a small set of anchors) sits at the top, and all federation members are directly or indirectly certified by it. For instance, a Federation Operator could run a trust anchor that onboards all participants (issuing entity statements for each member's metadata). This model is straightforward—if two parties share the same trust anchor, they can establish trust. Many industry trust frameworks adopt a hierarchical model with a central authority (e.g., a government or consortium). The Consumer Data Right (CDR) open banking ecosystem in Australia, for example, has an accreditation registry run by a central authority; that authority could serve as a trust anchor that signs statements for each accredited Data Recipient or Data Holder.

Mesh or Multilateral Trust

In some cases, there may not be a single top authority. An entity could be part of multiple federations or trust networks. OIDF allows multiple entries in authority_hints and an RP can trust multiple anchors. Trust is established if any trusted anchor yields a valid chain. This enables more of a mesh trust model, where, say, different sector authorities cross-recognize each other's credentials. For example, an AI service might be accredited in both a healthcare federation and a financial services federation—an enterprise in either sector could trust the service via the respective authority. Mesh models are more complex (involving possibly intersecting trust roots), but Federation's design supports it by allowing intersection of trust chains. In practice, mesh trust still usually relies on a few well-known anchors rather than complete decentralization.

Federation Operators and Authorities

A **Federation Operator** is an entity that administers a federation—often operating the trust anchor or intermediate authorities. In a hierarchical model, the operator might be the trust anchor itself. In a distributed model, the operator might coordinate policies among multiple authorities. Federation operators often define the **policies** (technical and legal) of the trust framework. They may also provide **discovery services**—e.g. a **resolve endpoint** that given an entity ID returns a fully validated metadata (performing steps 1–7 internally and caching results). This can simplify implementation for RPs/OPs. The federation operator's role is analogous to that of a certificate authority or a central trust registry, but with richer, dynamic metadata handling.

Below is an architectural view of a **hierarchical trust model** with a single trust anchor (which could be a federation operator or sector authority) signing metadata for multiple entities:

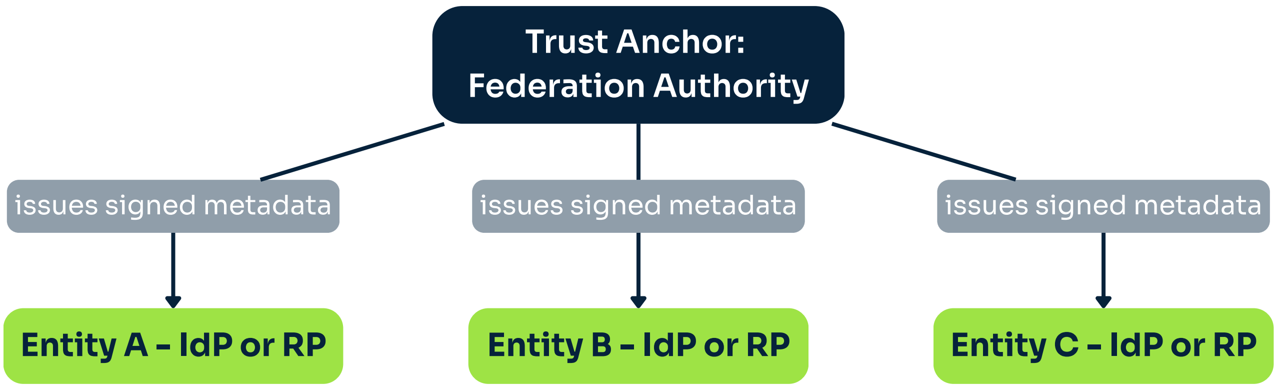


Figure: A federation trust hierarchy.

In this example, **all entities trust the anchor**. If Entity A wants to trust Entity B, it finds B's metadata signed by the shared **Trust Anchor**, which serves as the root of trust. In a mesh scenario, an entity could have multiple such arrows from different trust anchors, and trust is established if at least one chain connects through an anchor that the relying party knows and accepts.

Critically, OpenID Federation 1.0 ensures that whether the trust model is centralized or multilateral, **the mechanism of verification is uniform**. The software architect's task is mainly to configure the appropriate trust anchors (and required policies) that align with their trust framework.

Applying Federation to Agentic AI & MCP Integrations

With the federation concepts in hand, let's look at how **OIDF 1.0** can be applied to real-world scenarios involving AI agents and multi-party data integrations.

Use Case 1: Onboarding an External AI Agent (SaaS Integration)

Imagine an enterprise wants to allow an external AI SaaS (an AI assistant platform) to access internal data on behalf of users—for instance, an AI scheduling agent that needs to read employees' calendars via an API. In a traditional model, the enterprise would manually register the AI platform as an OAuth client in its IdP or API gateway, or use API keys—a process that doesn't scale well and requires significant coordination (as covered in Part 4 with trust registries). With **OpenID Federation**, this onboarding can be seamless and automated:

Mutual IdP-Client Trust

The AI provider (SaaS) likely operates its own IdP to manage agent identity or user sessions on its side. The enterprise has its IdP (e.g., an Azure AD or similar) protecting the calendar API. To integrate, the AI platform (as an OIDC client) needs to obtain an access token from the enterprise IdP for the API. Using federation, the enterprise IdP and the AI client establish trust at the OIDC level. The AI platform presents its **federation entity ID** and metadata when initiating the OAuth flow. The enterprise IdP, upon receiving an authorization request from an unfamiliar client, triggers trust chain resolution: it fetches the AI client's entity configuration and walks the chain to a known trust anchor. If, for example, the AI platform is accredited under a certain **AI services trust network**, the enterprise might trust that network's anchor. Once the chain validates, the IdP knows this client is trustworthy and also sees any metadata and policy attached (e.g., permitted grant types, redirect URIs, etc.).

Automatic Client Registration

Because the AI client's metadata is now verified, the enterprise IdP can automatically treat this client as registered. In practice, that means issuing tokens (or allowing the authorization request to proceed) without a human admin configuring the client in advance. OIDF 1.0's automatic registration flow (sometimes called automatic client onboarding) relies on the federation metadata having indicated the client is allowed to register automatically. The IdP would also apply any federation policies before actually issuing tokens—for instance, ensuring the scopes requested by the client are allowed under the trust framework's rules.

AI Trusting Enterprise IdP

On the flip side, the AI platform verifies the enterprise IdP's identity via federation as well. Perhaps both the AI platform and the enterprise IdP are members of the same federation (or have overlapping trust anchors). The AI platform's agent will fetch the enterprise IdP's entity configuration, follow its authority_hints, and confirm the IdP is legitimate and accredited (e.g., it's the genuine IdP for that enterprise, certified by some corporate or industry authority). This protects against the AI agent accidentally sending user credentials or tokens to a rogue IdP impersonator.

Token Exchange and Access
Once the OIDC handshake succeeds, the enterprise IdP issues an OAuth access token and/or ID token. The AI agent can now call the enterprise's MCP (Model Context Protocol) **server or API** using that token, to retrieve the needed data (in our example, the calendar info). The enterprise API, when it sees the token, trusts it because it's issued by its own IdP (normal internal trust), or if the token was issued by the AI's IdP, the API could similarly verify a trust chain. Typically, one would design it so that the enterprise's IdP issues the token for access, which is straightforward once federation allowed the AI client to authenticate the user.

The result: The external AI service was onboarded almost instantly, by virtue of presenting the right credentials in a federation. The heavy lifting (ensuring the AI service is accredited/allowed) was done out-of-band by the trust framework—e.g., the AI provider might have obtained a **trust mark** or been whitelisted by an authority, and now OIDC Federation uses that to technically enforce trust. A trust mark, in federation terms, is a "statement of conformance to a set of requirements as determined by an accreditation authority"—for instance, a badge indicating the AI provider is compliant with specific security or privacy standards. In OIDF, trust marks are digital and can be included in entity metadata, so the enterprise IdP could even require a certain trust mark to be present in the AI client's metadata during chain validation.

Why is this valuable for agentic AI? Because AI systems might need to connect to countless enterprise services, a federation approach means an enterprise can trust any number of qualified AI agents without bespoke onboarding for each. As long as those agents are in the federation (i.e., have the proper authority signatures or trust marks), the identity federation layer will automatically enable secure authentication and authorization. This drastically reduces integration time and the risk of misconfigured credentials.

Use Case 2: Federating Enterprise IdPs for AI Delegation Across Domains

Now consider a large enterprise (or a coalition of organizations) where multiple identity domains exist—for example, a company with separate IdPs for different business units, or a partner network where each organization has its own IdP. An AI agent operating in this environment may need to move seamlessly between domains. A concrete scenario: an internal AI assistant that can orchestrate tasks involving an HR system (authenticated via Corporate AD) and a Finance system (authenticated via a separate IdP), or an AI service that delegates user-specific actions across subsidiaries.

Using OpenID Federation, the enterprise can establish a federated trust fabric among its IdPs and services to support such cross-domain delegation:

Common Trust Anchor
The enterprise (or consortium) can act as a trust framework operator by setting up a trust anchor that represents corporate policy. Each internal IdP and major service registers under this anchor (meaning their metadata is signed by the corporate authority). For instance, CorpIDP-A and CorpIDP-B (two domains) would both have entity statements from the corporate trust anchor. Additionally, any internal AI services or clients would also be registered under the same anchor or a subordinate authority.

Cross-Domain SSO for AI
If an AI agent in Domain A needs to access an application in Domain B on a user's behalf, federation makes it possible to do this as an OIDC federated authentication rather than a one-off trust. The application in Domain B (RP) can accept an IdP from Domain A if a trust chain exists via the corporate anchor. Essentially, when the user's agent tries to authenticate to Domain B's service using Domain A's IdP, Domain B will see the IdP's metadata and verify it against the corporate trust anchor. Since both IdPs are policy-compliant under one trust framework, Domain B will trust the incoming identity assertion from Domain A. This can enable single sign-on or token exchange across domains, which is crucial for AI that straddles multiple identity silos.

Delegated Token Exchange
Alternatively, consider an AI service that holds a token from Domain A's IdP (for a user) but needs to call an API protected by Domain B's IdP. Federation can support a standardized token exchange if both IdPs trust each other's credentials via the anchor. For example, Domain B's token endpoint might accept a JWT from Domain A's IdP (or from the AI service) if it's signed and rooted in the shared trust anchor. The federation metadata could advertise support for specific grant types or token exchange profiles in each IdP's metadata, making inter-operability explicit.

Unified Policy Enforcement
Because a single trust framework governs all these domains, the enterprise can enforce consistent rules—possibly akin to an internal "CDR-style" accreditation, except internal. For instance, the corporate trust anchor's metadata policy might mandate that all IdPs use a certain LoA (Level of Assurance) or that all tokens have an azp claim for the authorized party. Since the anchor signs off on each entity's metadata, these rules propagate throughout, ensuring that when the AI agent transits from one domain to another, no domain is a weak link in terms of security compliance.

The diagram below illustrates a cross-domain interaction facilitated by federation, with an AI agent bridging two identity domains:

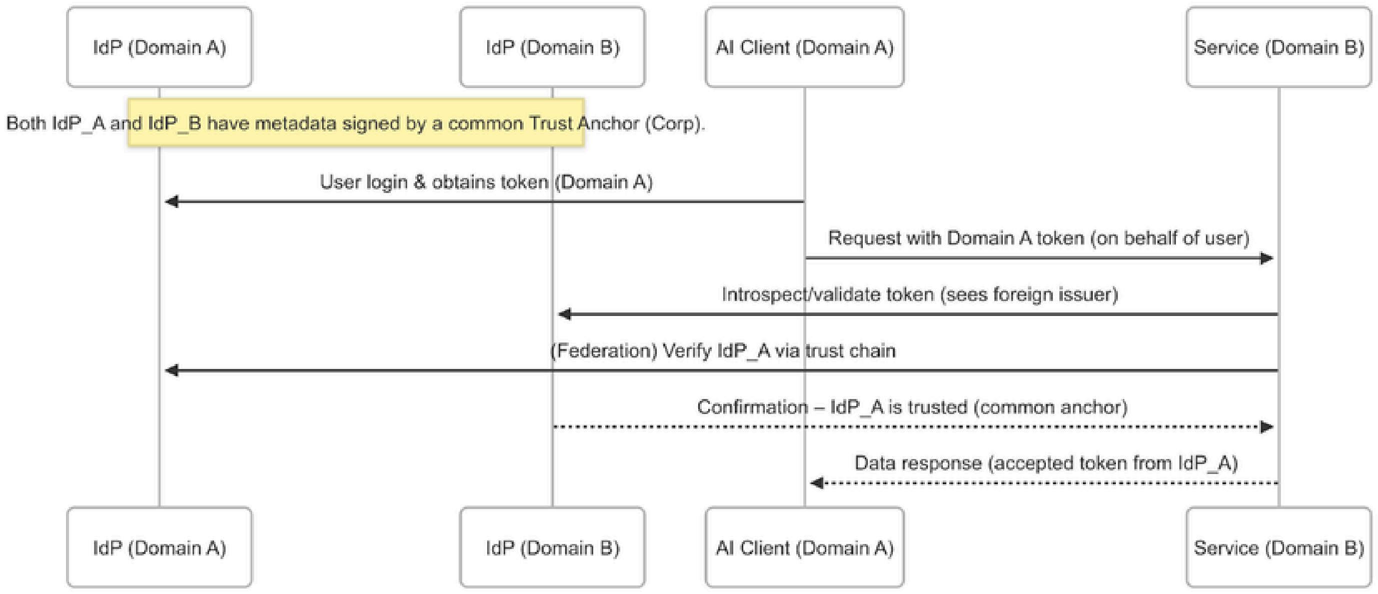

IdP (Domain A)
IdP (Domain B)
AI Client (Domain A)
Service (Domain B)
Both IdP_A and IdP_B have metadata signed by a common Trust Anchor (Corp).
User login & obtains token (Domain A)
Request with Domain A token (on behalf of user)
Introspect/validate token (sees foreign issuer)
(Federation) Verify IdP_A via trust chain
Confirmation – IdP_A is trusted (common anchor)
Data response (accepted token from IdP_A)
IdP (Domain A)
IdP (Domain B)
AI Client (Domain A)
Service (Domain B)

Figure: Federated trust across two domains.

Here, IdP_A and IdP_B might belong to the same company or partners. When Service_B (protected by IdP_B) encounters a token issued by IdP_A (via the AI agent's request), it uses OpenID Federation to validate IdP_A's trust chain against the corporate trust anchor. Since the chain checks out, Service_B accepts the token as if it were its own, allowing the AI's request to succeed. This kind of automated trust brokered by a common policy is what enables agentic AI workflows to scale across organizational boundaries.

Trust Frameworks and Federation Policy Overlays

A major strength of OpenID Federation is that it can embody **Trust Frameworks**—the legal, security, and compliance rules of an ecosystem—in a technical form. The **Consumer Data Right (CDR)** and various Open Banking regimes are good examples of trust frameworks that require participants to be accredited and follow certain standards. Here's how such frameworks map to OIDF 1.0 concepts:

Accreditation Authority as Trust Anchor
In a CDR-style model, a government body or appointed authority accredits Data Recipients. In OIDF terms, this authority would operate a Trust Anchor. Each accredited entity (e.g., a bank or fintech app) publishes an entity configuration, and the trust anchor issues an entity statement for each, effectively saying "I attest that this software is accredited under CDR." All participants (banks and fintechs) would configure their systems to trust the anchor's key. This way, whenever a data sharing request occurs, the parties can validate each other's accreditation status automatically via the trust chain to the anchor.

Federation Metadata Policies
Trust frameworks often impose uniform requirements—for example, "OAuth clients must use MTLS or private key JWT," "Tokens must be compliant with FAPI standards," etc. The **metadata_policy** in federation allows these to be baked into the trust process. The trust anchor can include policy in its statements such that any participant's metadata is automatically checked for compliance. If, say, an RP tries to register with an OP but doesn't meet the policy (e.g., it wants to use an insecure auth method not allowed by the federation), the trust chain evaluation will fail or the metadata will be adjusted to conform. Essentially, the federation policy overlay ensures **technical interoperability and security rules of the trust framework are uniformly enforced**.

Trust Marks for Fine-Grained Qualifications
Some trust frameworks categorize participants (for instance, an entity might be Read-Only Accredited vs Read/Write Accredited in a data-sharing context). OIDF 1.0 supports **Trust Marks**, which are like digital badges issued by an authority, asserting a specific qualification or compliance. A trust mark is defined as a statement of conformance to specific requirements by an accreditation authority. For example, an authority might issue a trust mark indicating compliance with **Open Banking Security Profile** or a certification like ISO27001. These trust marks appear in the entity's metadata and can be required by counterparties. In a CDR mapping, one could imagine a trust mark for "CDR Accredited Data Recipient". An OP could refuse to issue tokens unless the incoming client presents that trust mark in its trust chain. Trust marks thus add an extra layer of policy on top of the basic yes/no trust decision, enabling sector-specific nuances to be captured.

Dynamic Lifecycle Management
Trust frameworks aren't static—new participants join, others revoke or expire. Federation handles this via metadata refresh and expiration times on the JWT statements. If an accreditation is revoked, the trust anchor can stop signing that entity's metadata (or update a trust mark status endpoint). Relying parties and IdPs will periodically refresh or revalidate chains (they cache them with a validity period). This is more scalable and less error-prone than updating static trust lists or CRLs, since the federation endpoints themselves distribute the changes. For architects, this means less manual oversight to remove or add trusted partners; the federation automates it according to the authoritative source.

In summary, OpenID Federation 1.0 provides the technical rails to implement governance-heavy trust frameworks in a standardized way. Whether it's an open banking ecosystem or a cross-company AI collaboration, the combination of **trust anchors, signed metadata chains, policy rules, and trust marks** can mirror the real-world accreditation and trust requirements in a machine-readable, automated fashion. This lets software architects focus on **policy configuration** rather than writing custom logic for each new integration.

Conclusion

As agent-driven systems and multi-party protocols like MCP become integral to business workflows, the need for **scalable trust** is paramount. OpenID Federation 1.0 offers a powerful solution: it enables federated trust at internet scale, allowing organizations to automatically establish and verify trust relationships through cryptographically assured metadata. By evolving from the manual or semi-automatic approaches of Dynamic Client Registration and static trust registries (as discussed in Part 4) to fully **policy-governed federation**, architects can significantly reduce friction in onboarding new AI agents or services.

In this article, we explored how OIDF 1.0 works—from **federation metadata and trust chains** to real-world deployment scenarios with agentic AI. The takeaway for software architects is that **automated trust frameworks** are not just theoretical; they are practical and increasingly necessary. By adopting OpenID Federation (and layering on existing trust frameworks like CDR or OpenBanking), organizations can ensure that any AI agent or service they integrate is verifiably trustworthy and compliant from the first handshake. This paves the way for a future where intelligent agents interact across organizational boundaries with security and confidence, without weeks of paperwork or custom integration per partner. The path from closed systems to an open, federated ecosystem of AI and services is now clearer—and it's built on standards like OpenID Federation.