

Open Banking Reference Technical Architecture - (opinionated) Best Current Practices

Version 0.1

Executive Summary

Over 95 jurisdictions are pursuing open banking and open data¹, and only a handful are live today. That means many jurisdictions are developing their policies and evaluating their technical choices simultaneously. Each ecosystem needs to determine what “tech stack” will help it comply with its laws and regulations.

At the time of writing this whitepaper, the OIDF supports 26 ecosystems, with 13 of those being open data deployments that have already launched or have selected FAPI. In this capacity, the OpenID Foundation has a unique, “front row seat” on the due diligence and implementation decisions for these ecosystems. These learnings can benefit the ecosystems that follow.

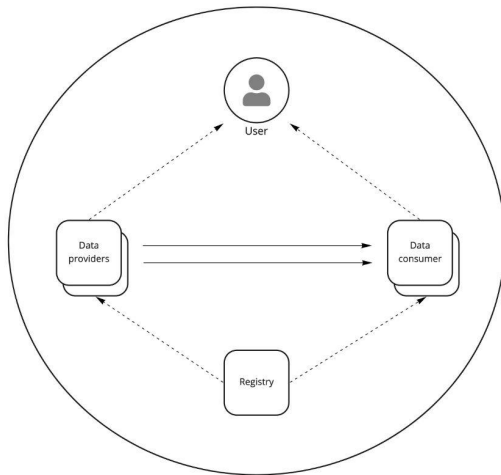
This paper offers open data policy makers and technologists an “opinionated” view on how to layer specifications and what specs and configuration decisions they will want to consider to deliver an open data ecosystem that can not only provide domestic interoperability and scale domestically, but can also allow for cross-border interoperability in the future. The tech stack characterized here is likely to meet the needs of most ecosystems; however, these implementations will only be practical if they are deployed using best practices. Ecosystem decision makers are encouraged to join the OIDF’s Ecosystem Community CG to discuss trade-offs on these and other governance decisions they are facing. There is no fee to participate or membership requirement to participate in the Ecosystem Support CG².

¹ <https://www.jbs.cam.ac.uk/faculty-research/centres/alternative-finance/publications/the-global-state-of-open-banking-and-open-finance-report/>

² <https://openid.net/cg/ecosystem-support-community-group/>

1. Open Banking ecosystem building blocks

To set up any API ecosystem, you need to select technical standards and define business and legal rules to ensure a common, interoperable, and secure process for all participants.



Technical standards

Rules and regulations
(business and legal)

2. Technical building blocks

The typical API ecosystem that involves customer data sharing between participants requires the same technical building blocks as shown below:

Technical standards

Jurisdiction specific profile (minimal, if required)

System
registration,
discovery and
management

Consent capture, management and enforcement

Non-repudiation
framework

User
Identity

API Security profile

API specifications

2.1. API Security and Authorization Framework

2.1.1. API Security Profile

FAPI is a de facto industry standard in the open banking and open data world. This security profile has been adopted in many jurisdictions worldwide.

FAPI 2 security profile is a secure OAuth profile that utilizes authorization code flow with PAR:

FAPI 2 = OAuth 2 + OAuth 2 Best Current Security practices

OIDF approved the FAPI 2 security profile as the final version in February 2025.

https://openid.net/specs/fapi-security-profile-2_0-final.html. The FAPI WG maintains FAPI in the OpenID Foundation.

At the time of writing this document, there are already live ecosystems that have adopted FAPI 2:

- Notably, Norwegian Health Network (HelseID), ConnectID in Australia, and yes.com in Germany.
- New ecosystems that have selected and are rolling out FAPI 2.0 are India's Aadhaar, the US FDX, and the Chilean Monetary Authority.
- Existing ecosystems, such as the Australian Open Banking (CDR) and the Saudi Arabian Monetary Authority, as well as the UAE, have committed to migrating from FAPI 1 to FAPI 2.

Why FAPI 2:

- Significantly simpler to implement than FAPI 1, especially on the client side (e.g., fintech).
- Designed ground up, starting with the attacker model first. It is recommended that each ecosystem review the attacker model to understand the limitations and key assumptions of its security profile.
- Formally analyzed by the University of Stuttgart³.
- Worldwide ecosystem adoption and vendor support. The FAPI 2 profile primarily consists of existing specifications with some additional security controls.
- An extensive conformance testing and certification program is available from the OpenID Foundation.

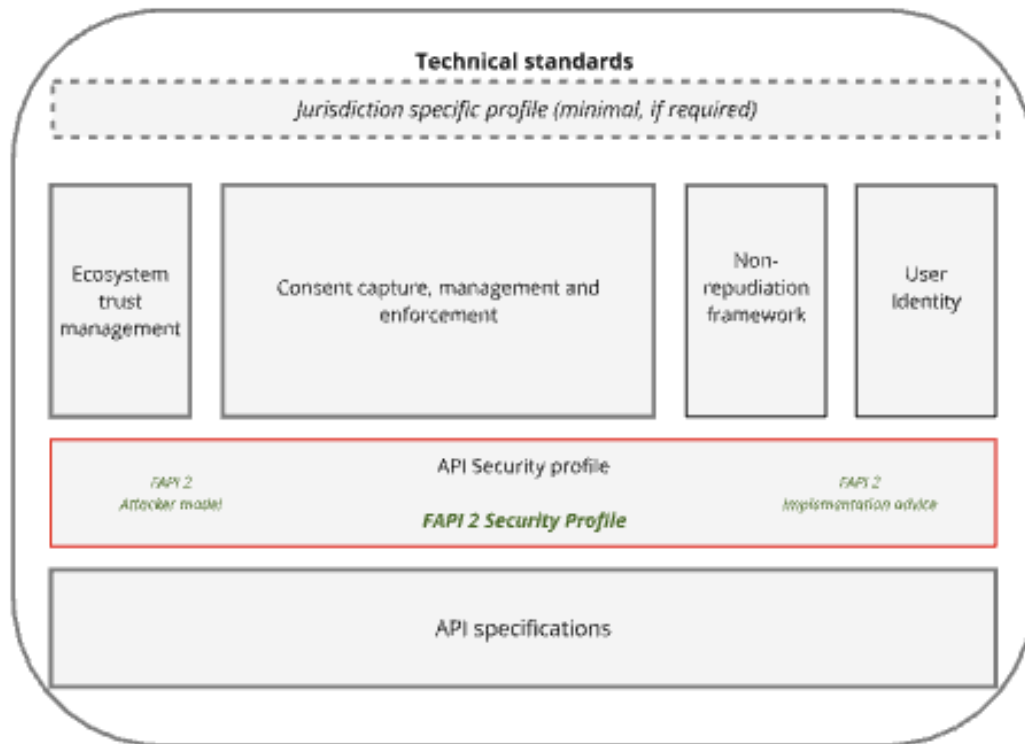
Lock it in: **FAPI 2**.

Specifications:

- FAPI 2.0 Security Profile: https://openid.net/specs/fapi-security-profile-2_0-final.html
- FAPI 2.0 Attacker Model: https://openid.net/specs/fapi-attacker-model-2_0-final.html

³ https://openid.net/wordpress-content/uploads/2022/12/Formal-Security-Analysis-of-FAPI-2.0_FINAL_2022-10.pdf

- FAPI 2.0 Implementation guide *TBC link*



2.1.2. FAPI 2 Ecosystem profile choices.

2.1.2.1 Closed or open ecosystem.

From a security perspective, FAPI 2 equally allows ecosystems to be both open and closed, depending on whether the ecosystem governing body wants to control who is allowed to connect to its ecosystem participants at the transport level (via MTLS).

Most of the existing and newly developed FAPI1 and FAPI2 ecosystems chose a closed model to provide additional governance control at the transport level. These ecosystems usually operate a certificate authority or maintain a trust list of CAs.

As an additional layer of protection, it can minimise the impact of potential unknown vulnerabilities by limiting the number of participants connected to the ecosystem.

Lock it in: **MTLS locked down ecosystem.**

2.1.2.2. Access Token binding

FAPI 2 supports two mechanisms for token binding: MTLS and DPOP.

- 1) MTLS is considered to be the gold standard and has been deployed by virtually all existing open banking ecosystems.
- 2) DPop is a newer and less battle-tested specification.

Lock it in: **MTLS for token binding.**

Note: MTLS is also an effortless choice if you operate in an MTLS lockdown ecosystem (see 2.1.2.1 above).

2.1.2.3. 3rd party client authentication

In a context of client authentication for open banking, to avoid using shared secrets, the FAPI 2 mandates one of the following authentication methods:

- Private_key_jwt as specified in Section 9 of *OpenID Connect*⁴.
- MTLS as specified in Section 2 of *RFC8705*⁵.

While there is no significant difference between the two methods in the context of open banking, the majority of current ecosystems choose to use *private_key_jwt*, keeping authentication at the application layer.

Lock it in: **Private_key_jwt.**

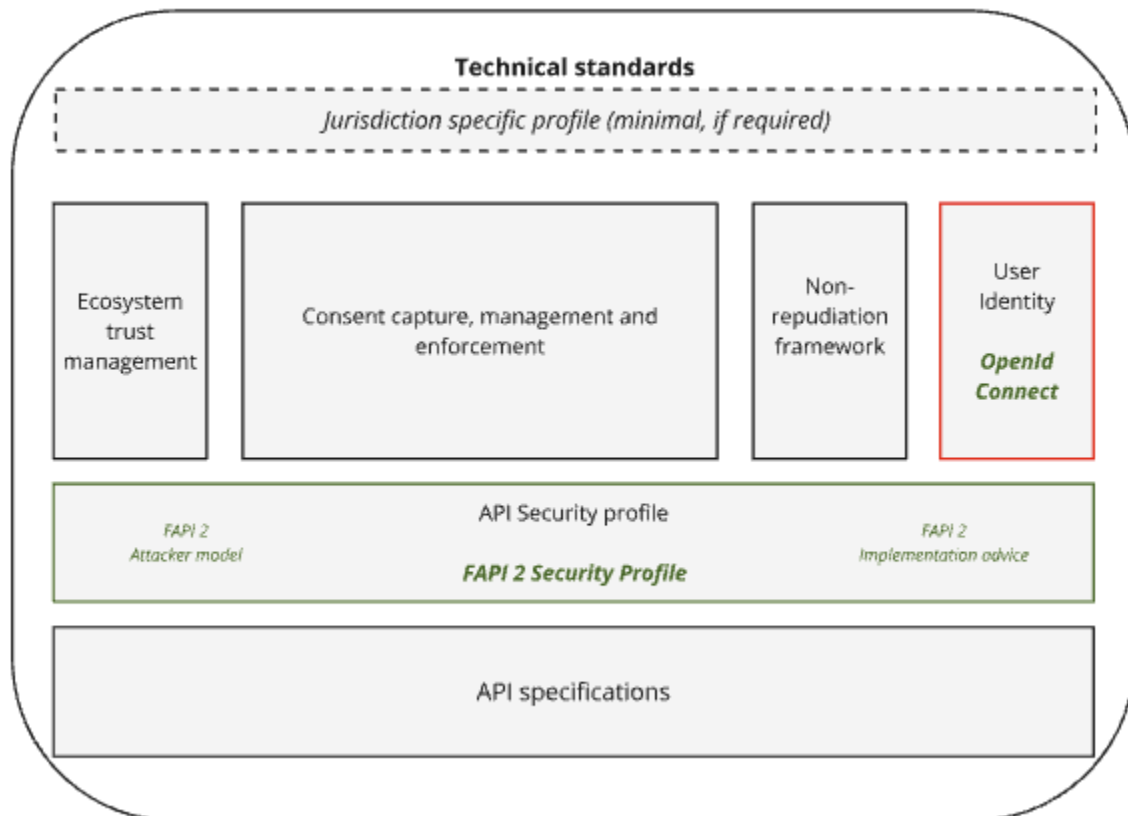
Note: Ecosystems utilize *private_key_jwt* client authentication in conjunction with MTLS transport control to provide ecosystem governing bodies with two distinct controls, each with different operational characteristics at the application and transport layers, respectively.

⁴ https://openid.net/specs/openid-connect-core-1_0.html

⁵ <https://www.rfc-editor.org/info/rfc8705>

2.2. Identity

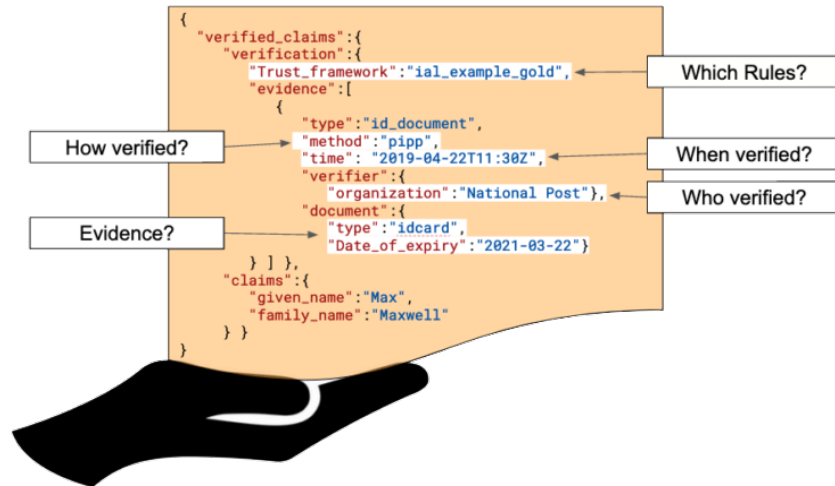
Identity is a protocol that defines how identity information is transferred from a Data Provider to a Data Consumer with the End-User's consent. Most ecosystems worldwide have adopted [OpenID Connect 1.0](#). This end-user authentication OAuth 2 extension has become a de facto industry standard, both within and outside the Open Banking ecosystems, with broad vendor support.



Lock it in: **OpenID Connect**.

When standard and straightforward OpenID Connect claims are insufficient to transmit identity verification data or metadata, another specification should be used: OpenID Connect for Identity Assurance.

For example, if an RP is required by law to receive additional metadata, such as the date and place of verification, or the documents and sources used.



Ecosystems can utilize template requests and responses developed by industry participants for various use cases or develop them in-house. OIDC4IDA allows both models.

Lock it in: **OpenID Connect for Identity Assurance** (optional).

Specifications:

- OpenID Connect core v1: https://openid.net/specs/openid-connect-core-1_0.html
- (Optional) OpenID Connect for Identity Assurance v1: https://openid.net/specs/openid-connect-4-identity-assurance-1_0.html

2.3. Non-repudiation framework

Most ecosystems require a level of non-repudiation to guarantee that messages between ecosystem participants haven't been tampered with. This is especially important for any "write changes" requested by a third party, for example, payment initiation or account opening/closure.

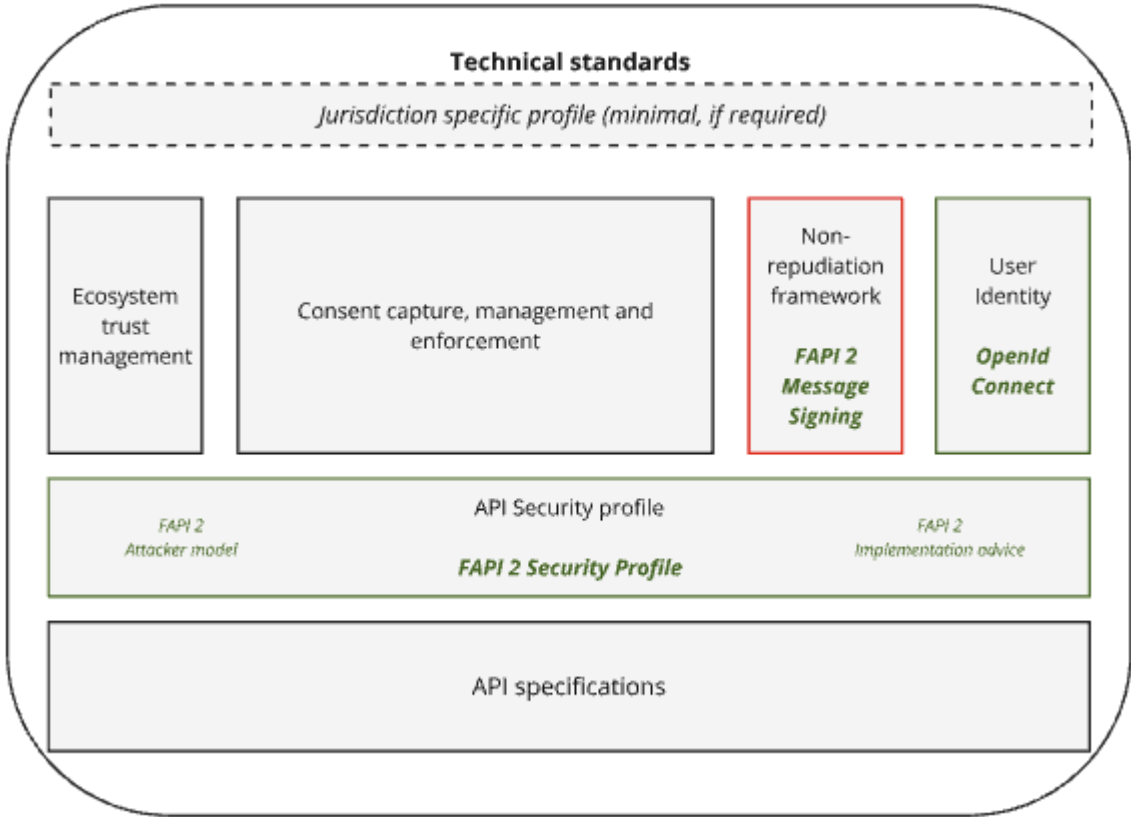
The FAPI technical framework and FAPI 2 Message signing specification provide different levels of non-repudiation that can be achieved.

Ecosystems can select and implement various non-repudiation options, ranging from none to all, or some but not others.

Current default configuration for most of the ecosystems is:

| Signing | Enabled | Specification |
|-----------------------|---------|---------------|
| Client request | | |

| | | |
|---------------------------------------|-----|---|
| Authorisation Request | Yes | FAPI 2 Message signing: Signed Requests Objects |
| Authorization Server responses | | |
| Authorization Response | No* | FAPI 2 Message signing: JARM |
| ID token | Yes | FAPI 2 Message signing: OpenID Connect |
| Resource server responses | | |
| API response | No | FAPI 2 HTTP Messages |



Lock it in: **FAPI Message signing.**

Specifications:

- *FAPI 2 Message signing* https://openid.net/specs/fapi-message-signing-2_0-02.html

- *(Optional) FAPI 2 HTTP Messages (under development at the time of publishing of this document)*⁶*

⁶ If your ecosystem requires non-repudiation for API responses, please contact OpenID Foundation FAPI WG to collaborate on specification development, deployment and testing.

2.4. Fine-grained consent

2.4.1. Rich Authorisation Request - adding fine-grained details to the request

The use of OAuth 2.0 Rich Authorization Requests (RAR) [<https://www.rfc-editor.org/info/rfc9396>] is recommended when the scope parameter is not expressive enough to convey the authorization that a client may want to obtain:

- Implementers can utilize RAR to standardize the envelope and data payload for conveying complex, fine-grained authorization requests (for example, for privacy-preserving data sharing or payments).
- RAR allows the expression of local requirements and data models specific to locally defined functional APIs or other functionality. OIDF community aims to develop multiple templates across various ecosystems to establish a shared library, similar to the JSON schema for OpenID Connect and Identity Assurance. These schemas can ideally be plugged into the OIDF conformance test suite via configuration, rather than requiring an additional build.
- RAR can be used across existing authorization flows and express the grant's fine-grained permissions desired by the client. The authorization server would be responsible for initial RAR processing.

Here is an example of the fine-grained authorization request:

```
"authorization_details": [  
  {  
    "type": "account_information",  
    "actions": [  
      "list_accounts",  
      "read_balances",  
      "read_transactions"  
    ],  
    "locations": [  
      "https://example.com/accounts"  
    ]  
  }  
],
```

The outer part of the data envelope (**above, in bold**) and the outer data structure are standardized. This part falls within the scope of OIDF certification and is subject to formal security testing.

The inner part of the data envelope ([above in blue](#)) is localized to ecosystem-specific and local requirements. ODF sees value and can facilitate ecosystems by maintaining standard templates for some local requirements

Some ecosystems, such as Open Finance UAE, have already implemented RAR in their ecosystem.

The previous non-standardized and non-interoperable alternative used a custom payload with a custom API and scope values. Every ecosystem had to re-invent one or copy and paste it from another.

Lock it in: **Rich Authorization Request (RAR)**.

Specification: <https://www.rfc-editor.org/info/rfc9396>

2.4.2. Grant Management - Adding fine-grained authorization to the transaction flow

OAuth 2 Grant Management (GM) [<https://openid.net/specs/oauth-v2-grant-management-ID1.html>] is recommended for ecosystems that require interoperable grant (authorization) management.

OAuth 2 Grant Management is a simple OAuth extension implemented by Authorization Servers to provide third parties with a standardized ability to:

- Identify a specific authorization (grant) using `grant_id`. It is beneficial where multiple-party authorization or concurrent consents⁷ are required.
- View the details of the specific authorization. For example, in a case where multi-party approval is required, a client can query the authorization server to find out if the grant has been fully authorized or not yet.
- Changing, updating, replacing, or withdrawing a specific authorization opens up a range of business and technical use cases that can now be supported interoperably, such as a simpler user experience for reauthorizing consent and amending it, or facilitating technical migration to new consents.

Before Grant Management, all authorization servers already managed grants internally; however, there was no standardized interface for third parties to understand (let alone request changes), resulting in divergent internal and external behavior.

⁷ Concurrent consents allow establishment of multiple consents between a client and AS for the same user. It's a regulatory requirement in some ecosystems like CDR.

Current ecosystems have had to forgo fine-grained authorization or use non-standardized Consent or Intent APIs. OpenID Foundation's formal analysis and conformance testing don't cover custom solutions. Non-standard APIs can also be deployed outside your security infrastructure, thus creating additional security risks.

The Grant Management API was initially designed to address the complex requirements of emerging ecosystems, such as the CDR in Australia. In 2023, the OpenID Foundation and its community reviewed and approved the second draft of the implementer's document.

This specification closes the gap in a standardized way that aligns with the other specifications in the FAPI 2.0 framework, as seen in Diagram 2.

Lock it in: **OAuth 2 Grant management API**

Specification: <https://openid.net/specs/oauth-v2-grant-management-ID1.html>

2.6. Decoupled flow

FAPI-CIBA

3. Standards Conformance

Mandatory conformance testing.

5. Work-in-progress

5.1. Ecosystem trust management

Open Banking profile of OpenID Federation.

Currently, most open data Ecosystems are utilizing X.509 certificates to establish ecosystem trust management.

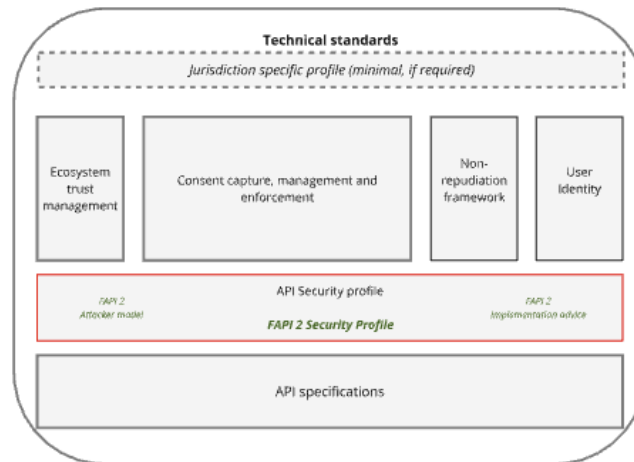
However, emerging new open banking and open data ecosystems will want to consider whether OpenID Federation might be a better fit for their ecosystem. OpenID Federation is a comprehensive trust management framework that is suitable to fulfill the requirements of Open Banking ecosystems.

5.2. Third-party model

5.3. Real-time event sharing framework

Open Banking profile for Shared Signals and events Framework

Reference Architecture summarised



| # | Building block | Recommendation | Specification URLs |
|-----|----------------------------------|---|---|
| 1.1 | API Security Profile | FAPI 2 | https://openid.net/specs/fapi-security-profile-2_0-final.html |
| 1.2 | Closed or open ecosystem | MTLS lockdown | |
| 1.3 | Access Token binding | MTLS | |
| 1.4 | 3rd party client authentication | Private_key_jwt | |
| 2 | Identity | OpenID Connect | https://openid.net/specs/openid-connect-core-1_0.html |
| | | OpenID Connect for Identity Assurance (if required) | https://openid.net/specs/openid-connect-4-identity-assurance-1_0.html |
| 3.1 | Non-repudiation framework | FAPI 2 Message signing | https://openid.net/specs/fapi-message-signing-2_0-02.html |
| 3.2 | Authorisation Request | Signed | |
| 3.3 | ID token | Signed | |
| 3.4 | Authorization Response | Signed (if required) | |

| | | | |
|---|-----------------------------|-----------------------------------|---|
| 4 | Fine-grained consent | Rich Authorization Request (RAR). | https://www.rfc-editor.org/info/rfc9396 |
| | | OAuth 2 Grant management API | https://openid.net/specs/oauth-v2-grant-management-ID1.html |

Conclusion

The OIDF believes that each ecosystem and the global infrastructure supporting data sharing and digital identity will be more secure and resilient through collaboration, enabling us to future-proof the path to global interoperability. Regular contact will also help establish the critical relationships needed if compromises to ecosystem security emerge.

References

<https://openid.net/standardized-fine-grained-authorization-with-oauth2-grant-management-and-rich-authorization-requests/>

Contributors

Gail Hodges
Mark Verstege
Joseph Heenan
Ralph Bragg