Comprehensive Overview
FAPI 1 and 2

Dr. Torsten Lodderstedt, yes.com
What is FAPI?

- A security and interoperability profile for OAuth for open banking and other use cases with high security requirements
- Includes new specifications as required
Versions

- **FAPI 1**
  - Developed from 2016 onwards and used existing OpenID Connect security mechanisms to patch OAuth security issues
  - Final specifications published
  - Adopted by UK OpenBanking, FDX, CDR, and Brasil

- **FAPI 2**
  - The next evolutionary step, simpler to use and with a broader scope
  - Based on analysis of most PSD 2 and other open banking initiatives as well as requirements from eHealth and eGovernment
  - Adopted in yes open banking scheme (~1000 banks)
Main differences between FAPI 1 and FAPI 2

- Simpler to use
  - through new mechanisms (e.g. Pushed Authorization Requests/PAR, no ID Token as detached signature required)

- Well-understood and better-defined security
  - FAPI 2 Baseline has same protection level as FAPI 1 Advanced
  - FAPI 2 Baseline **fully** protects against attacker model

- Broader interoperability
  - through coverage of rich authorization / consent management and secure access to APIs

- More versatile
  - through alternative mechanism for token replay protection (DPoP)
FAPI 2 Main Components
Pushed Authorization Requests (PAR) replace bespoke solutions like external resources with references in scope/claims, custom authorization request parameters, … → **Simplified development** through vendor support and reliance on TLS (signed requests possible) → Minimize data in front-channel to **improve security and increase robustness**
Rich Authorization Requests (RAR) enable fine-grained and complex consents.

- Structure of authorization details can be defined as needed (e.g. per application)
- Supports Multi-Consents

```json
[
{
  "type": "payment_initiation",
  "actions": [
    "initiate"
  ],
  "locations": [
  ],
  "instructedAmount": {
    "currency": "AUD",
    "amount": "123.50"
  },
  "creditorName": "Merchant123",
  "creditorAccount": {
    "bsb": "123-456",
    "accountNumber": "1234567890"
  },
  "paymentDescription": "INV123456 Description123"
}
]
Grant Management enables support for

- consent state synchronization
- consent revocation
- concurrent consents
- consent update & renewal
- Dashboards

Closely aligned with Australian requirements because it was started during AU CDR consent proposal discussions.
Grant Management (request new grant id)

(Pushed) Authorization Request)

POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0Mzo3Rm...

response_type=code&
client_id=s6BhdRkqt3
&grant_management_action=create
&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
&code_challenge_method=S256
&code_challenge=K2-ltc83acc4h...
&authorization_details=%5B%7B%2...

Token Response

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store

{"access_token": "2YotnFZFEjr1zCsicMWpAA",
"token_type": "example",
"expires_in": 3600,
"refresh_token": "tGzv3JOkF0XG5Qx2TIKWIA",
"grant_id": "0a15a804-b5b4-4a45-9cd9-18b1a44f3383",
"authorization_details": [...]}

(Pushed) Authorization Request)
Grant Management (API)

Query

GET /grants/0a15a804-b5b4-4a45-9cd9-18b1a44f3383
Host: as.example-bank.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Content-Type: application/json

{
    "authorization_details": [...]}

Revoke

DELETE /grants/0a15a804-b5b4-4a45-9cd9-18b1a44f3383
Host: as.example-bank.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

HTTP/1.1 204 No Content
Grant Management (request use of certain grant)

(Pushed) Authorization Request)

POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0Mzo3Rm...

response_type=code&
client_id=s6BhdRkqt3
&grant_management_action=update
&grant_id=0a15a804-b5b4-4a45-9cd9-18b1a44f3383
&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
&code_challenge_method=S256
&code_challenge=K2-ltc83acc4h...
&authorization_details=%5B%7B%2B...
PKCE

PKCE (RFC 7636) is used to detect code replay and CSRF

Dynamically generated cryptographically random key used to bind transaction to browser/device

Replaces ID token as detached signature

→ security check moved to AS

→ simple and robust
## Feature Comparison

<table>
<thead>
<tr>
<th>Topic</th>
<th>FAPI 1</th>
<th>FAPI 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Integrity</td>
<td>Signed Request Objects</td>
<td>PAR</td>
</tr>
<tr>
<td>CSRF</td>
<td>state + s_hash in ID Token</td>
<td>PKCE</td>
</tr>
<tr>
<td>Code Replay</td>
<td>ID Token as detached signature or JARM or PKCE</td>
<td>PKCE</td>
</tr>
<tr>
<td>Mix-Up</td>
<td>iss claim in ID token or JARM</td>
<td>iss response parameter</td>
</tr>
<tr>
<td>Access Token Replay</td>
<td>mTLS</td>
<td>mTLS or DPoP</td>
</tr>
<tr>
<td>Rich authorizations data</td>
<td><strong>not covered (custom solutions)</strong></td>
<td>PAR+RAR</td>
</tr>
<tr>
<td>Consent management</td>
<td><strong>not covered (custom solutions)</strong></td>
<td>Grant Management</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>Signed Request Objects, ID Token as detached signature API not covered</td>
<td>JAR, JARM, Signed Introspection Response, Simple HTTP Message Integrity Protocol</td>
</tr>
</tbody>
</table>
FAPI 1 (lodging intent) vs FAPI 2 (PAR+RAR)

User

Client

Authorization Server

API

Request Access Token

/POST /token
grant_type=client_credentials&scope=accounts

Return Access Token

{"access_token":"5/CKN69L8gdSY5..., ..."}

Create Account Request

/POST /account-access-consents

Return Account Request

{"ConsentId":"EXAMPLE-B-10000006010120", ...}

Authorization request

/authorize?request=eyJrWQiOiJyN...&...

Login and Consent

Authorization Response

code=HS9najKwp901hBcK348IUHiuH8374&
_id_token=eyJrWQiOiJQUz1INIs...&state=...

Exchange code for access token

/POST /token

code=HS9najKwp901hBcK348IUHiuH8374&code_verifier=...&...

Return access token

{"access_token":"4/CKN69L8gdSY5..., ..."}

API Access

GET /request/...

Authorization Response

code=HS9najKwp901hBcK348IUHiuH8374

Exchange code for access token

/POST /token

code=HS9najKwp901hBcK348IUHiuH8374&code_verifier=...&...

Return access token

{"access_token":"4/CKN69L8gdSY5..., ..."}

API Access

GET /request/...

{"authorization":"4/CKN69L8gdSY5..., ..."
FAPI 2 Security

- FAPI 1 RW Security Level with simpler to implement features and less reliance on client
- Increased interoperability (rich authorization + grant management)

=>

- Facilitates more secure implementations
Roadmap

● FAPI 2 Baseline
  ○ in first public draft for vote
  ○ implementers draft approval - June
  ○ underlying specifications (apart from GM) are stable specs with multiple implementations and vendor support

● Grant management
  ○ first public draft for vote in May
  ○ implementers draft approval - July

● FAPI 2 Signing
  ○ Under development

● FAPI 2 Advanced
  ○ first implementers draft: dependent on signing
FAPI adoption in new ecosystems

● Reasons to use FAPI 1
  ○ If vendors in an ecosystem already support FAPI 1
  ○ FAPI 1 is a mature and widely supported security profile.

● Reasons to use FAPI 2
  ○ FAPI 2 is easier to implement
  ○ FAPI 2 covers complex authorization requests and grant lifecycle management aspects
  ○ FAPI 2 (as profile for API access authorization) better fits with OpenID Connect (for identity claims provisioning) then FAPI 1
Ecosystems already using FAPI 1

- **Benefit for adoption:**
  - Simpler protocol and improved interoperability
  - Specification aligned with the latest OAuth best practices and security advice

- **Incremental adoption of FAPI 2 modules possible:**
  - Example: Australia adopted PAR with FAPI 1
  - PAR + RAR + Grant Management as full lifecycle consent management solution for FAPI 1

- **Running both profile in parallel is possible**
  - Would allow new clients to utilize the simpler protocol (and existing clients to migrate)