

KeyConf 24
@ARCOTEL Kaiserwasser, Vienna, Austria 

Keycloak's Updates on Emerging Paradigm of Identity and Compliance with Security Specifications

September 19, 2024

Takashi Norimatsu


OSS Solution Center

Hitachi, Ltd.

Takashi Norimatsu (**tnorimat** in GitHub) :

Keycloak maintainer (since Oct 2021),

Technical lead of Keycloak community “OAuth SIG”,

Senior OSS Specialist, Hitachi, Ltd., Japan 

■ Contributing security features to Keycloak since 2018.

- W3C Web Authentication API support (Passkeys authentication)
- Security features support (e.g., RFC 7636 PKCE, RFC 8705 OAuth MTLs, OIDC CIBA, RFC 9126 PAR, RFC 8032/8037 EdDSA, RFC 9449 DPoP, RFC 9207 OAuth2 Authz Server Issuer Identification)
- API security profiles support (e.g., FAPI 1.0 Baseline, FAPI 1.0 Advanced, FAPI-CIBA, FAPI 2.0 Baseline, FAPI 2.0 Message Signing, OAuth 2.1)

Contents

- 1. Emerging Paradigm of Identity : OID4VCI**
- 2. Compliance with Security Specifications**

1. Emerging Paradigm of Identity : OID4VCI

Keycloak's community is working on supporting OID4VCI.

Motivation :

The European Commission released "[The European Digital Identity Wallet Architecture and Reference Framework](#) (*1)" which describes that OID4VCI MUST be implemented as an Issuance Protocol.

➡ Keycloak can be used as an Issuer in this framework if Keycloak supports OID4VCI.

Credential Formats :

- [JWT VC](#) (*2)
Standardized by Decentralized Identity Foundation (DIF)
- [Selective Disclosure JWT \(SD-JWT\)](#) (*3)
Standardized by Internet Engineering Task Force (IETF)
- [Verifiable Credentials Data Model \(VCDM\)](#) (*4)
Standardized by World Wide Web Consortium (W3C)
- [ISO.18013-5 Mobile driving license \(mDL\)](#) (*5)
Standardized by International Organization for Standardization (ISO)

*1 : <https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework>

Credential Issuance Protocol

- [OpenID for Verifiable Credential Issuance \(OID4VCI\)](#) (*6)
Standardized by OpenID Foundation (OIDF)

Issuance Flows :

- Pre-authorization code flow
- Authorization code flow

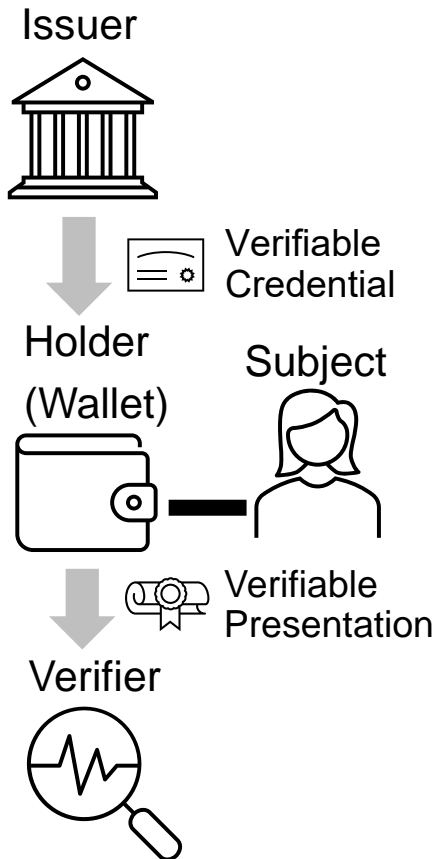
*2 : <https://identity.foundation/jwt-vc-presentation-profile/>

*3 : <https://www.ietf.org/archive/id/draft-ietf-oauth-selective-disclosure-jwt-07.html>

*4 : <https://www.w3.org/TR/vc-data-model/>

*5 : <https://www.iso.org/standard/69084.html>

*6 : https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html



*1 : "Verifiable Credentials Data Model v2.0" <https://www.w3.org/TR/vc-data-model-2.0/>

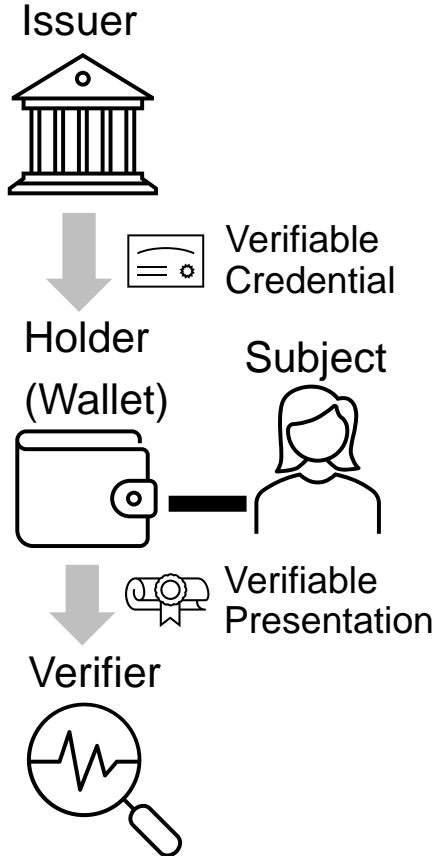
Issuer:= asserting claims about one or more subjects, creating a verifiable credential from these claims, and transmitting the **verifiable credential** to a **holder**. (quoted from *1)

Holder:= possessing one or more **verifiable credentials** and generating **verifiable presentations** from them.

A holder is often, but not always, a subject of the verifiable credentials they are holding. (quoted from *1)

Verifier:= receiving one or more **verifiable credentials**, optionally inside a **verifiable presentation** for processing.

Other specifications might refer to this concept as a relying party. (quoted from *1)



Verifiable Credential (VC):= a tamper-evident credential that has authorship that can be cryptographically verified.

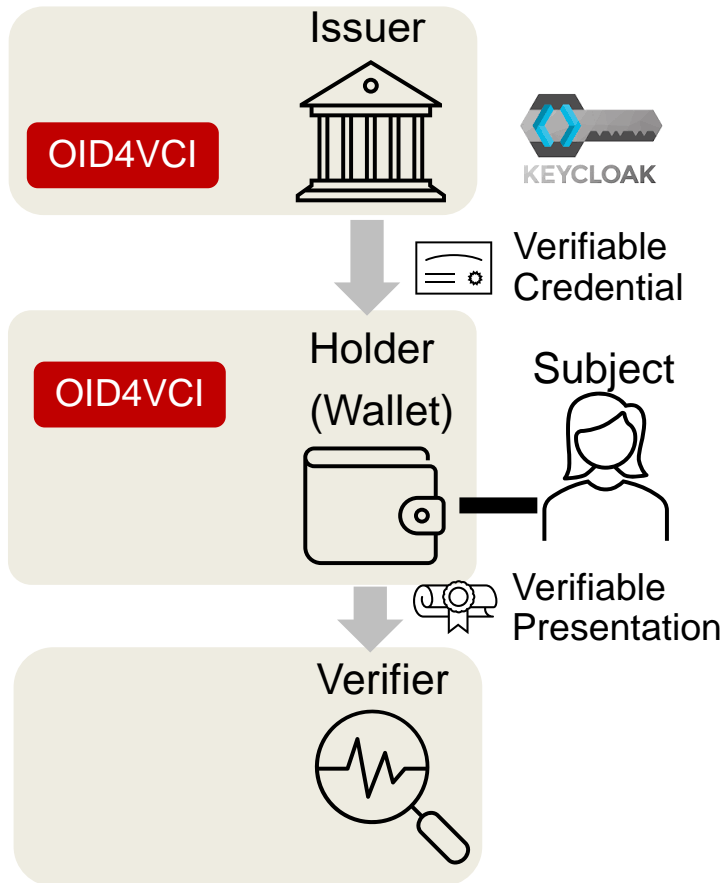
Verifiable credentials can be used to build **verifiable presentations**, which can also be cryptographically verified. (quoted from *1)

Subject:= A thing about which claims (an assertion made about the subject) are made. (quoted from *1)

Verifiable Presentation (VP):= a tamper-evident presentation encoded in such a way that authorship of the data can be trusted after a process of cryptographic verification. (quoted from *1)

*1 : "Verifiable Credentials Data Model v2.0" <https://www.w3.org/TR/vc-data-model-2.0/>

[Keycloak's community is working on supporting OID4VCI.](#)
(*1)

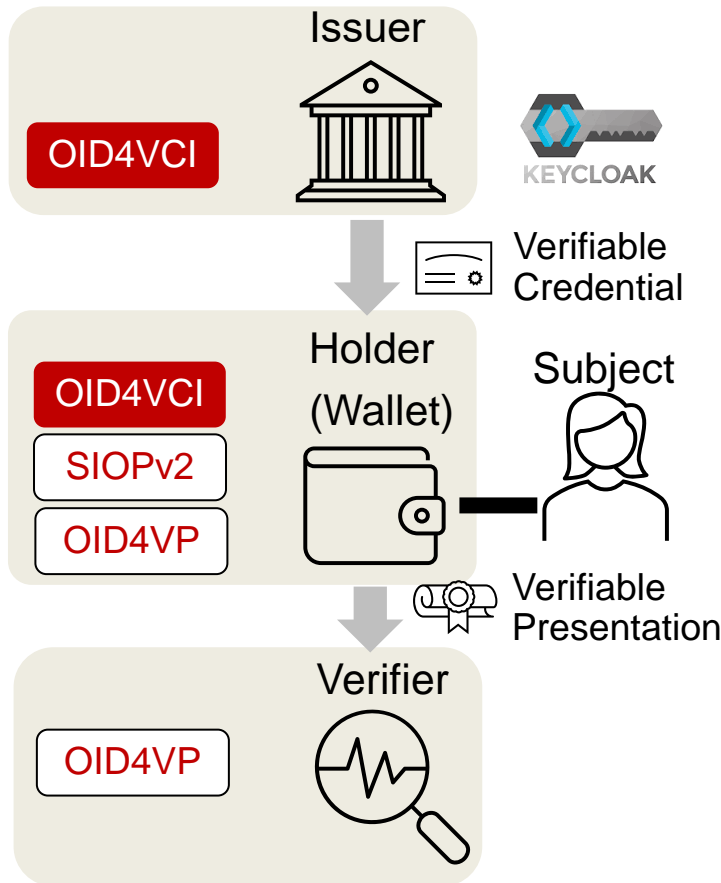


OpenID for Verifiable Credential Issuance (OID4VCI): = Defining an API that is used to issue Verifiable Credentials. (quoted from *2)

*1 :
<https://github.com/Keycloak/Keycloak/issues/25936>

*2 : https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html

[Keycloak's community is working on supporting OID4VCI.](#)
(*1)



OpenID for Verifiable Credential Issuance (OID4VCI) := Defining an API that is used to issue Verifiable Credentials. (quoted from *2)

Self-Issued OpenID Provider v2 (SIOPv2) := An OpenID Provider controlled by the End-User. (quoted from *3)

OpenID for Verifiable Presentations (OID4VP) := Defining a mechanism on top of OAuth 2.0 that enables presentation of Verifiable Credentials as Verifiable Presentations. (quoted from *4)

*1 :
<https://github.com/Keycloak/Keycloak/issues/25936>

*2 : https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html

*3 : https://openid.net/specs/openid-connect-self-issued-v2-1_0.html

*4 : https://openid.net/specs/openid-4-verifiable-presentations-1_0.html

The Current Keycloak's Support for OID4VCI

OID4VC Support - General

| | |
|---|--------------------------------|
| OID4VCI-supported Keycloak's version | 25.0.0 or later |
| OID4VCI support feature level | Experimental |
| Referred Version of OID4VCI specification | Implementer's Draft (draft 13) |

OID4VC Management/Administration

| | |
|------------------------------|---|
| Admin REST API direct access | ✓ |
| Admin CLI (kcadm) | ✓ |
| Admin Console | ✗ |

VC Issuance Flow

| | |
|--------------------------|---|
| Pre-Authorized Code Flow | ✓ |
| Authorized Code Flow | ⚠ |

Authorization Request Parameter

| | |
|--------------------------------------|---|
| scope | ✓ |
| authorization_details (RFC 9396 RAR) | ✗ |
| issuer_state | ✗ |

VC Issuance Variation

| | |
|-------------------------------|---|
| Immediate Credential Issuance | ✓ |
| Deferred Credential Issuance | ✗ |
| Batch Credential Issuance | ✗ |

VC Credential Offer

| | |
|--------------|---|
| Same-Device | ✓ |
| Cross-Device | ? |

VC Format

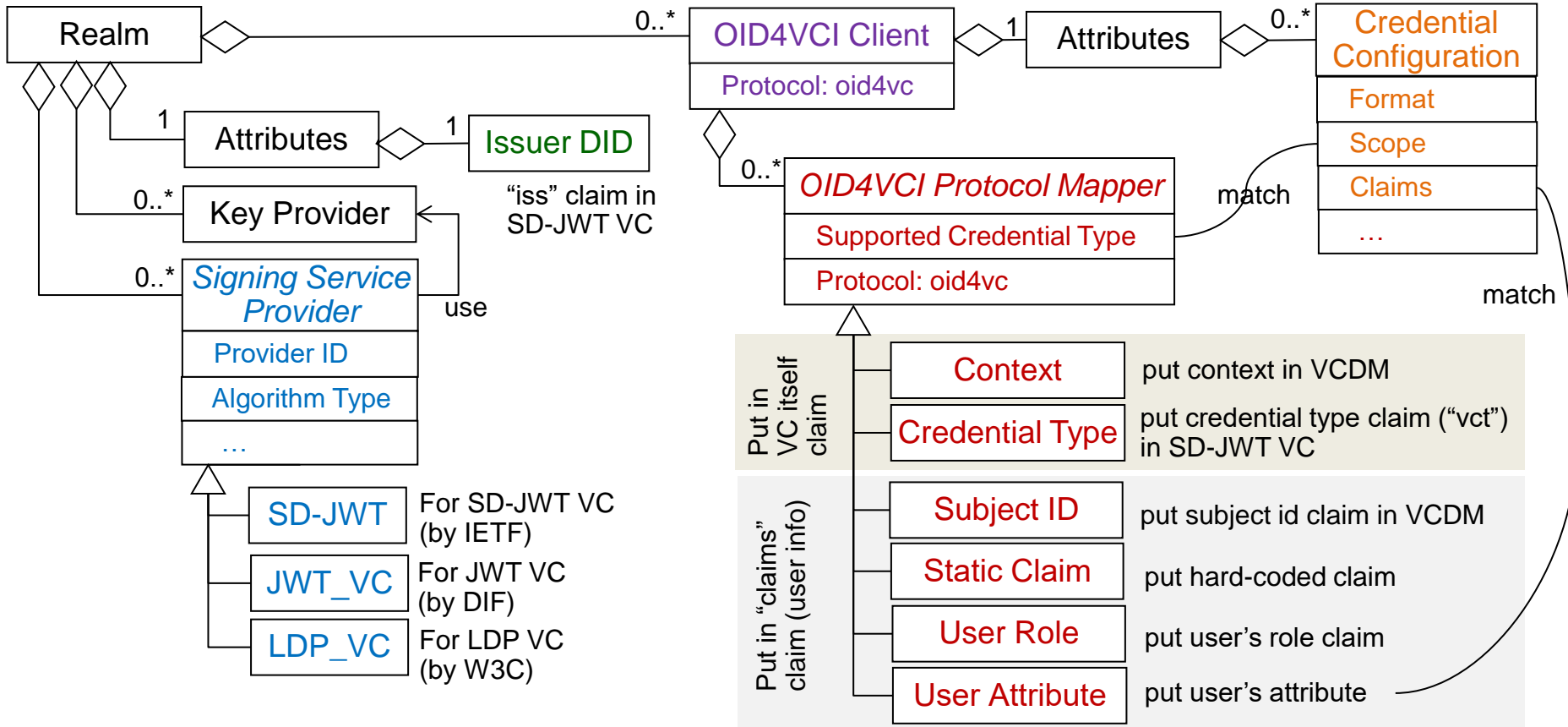
| | |
|-------------------|---|
| SD-JWT VC (IETF) | ✓ |
| JWT VC (DIF) | ? |
| LDP VC (W3C) | ? |
| mDL (ISO.18013-5) | ✗ |

VC Issuance Proof (Key Binding)

| | |
|---------------|---|
| jwt (SD-JWT) | ✗ |
| cwt | ✗ |
| ldp_vp (VCDM) | ✗ |

⚠ : will be supported from Keycloak 26 or later

Components for OID4VCI in Keycloak (25.0.0)



Represented as a realm component model.

Ex. SD-JWT Signing Service Provider(*)

```
{
  "id": "sd-jwt-signing_IdentityCredential",
  "name": "sd-jwt-signing-service for IdentityCredential",
  "providerId": "vc+sd-jwt",
  "providerType": "org.keycloak.protocol.oid4vc.issuance.signing.VerifiableCredentialsSigningService",
  "config": {
    "algorithmType": ["ES256"],
    "hashAlgorithm": ["sha-256"],
    "tokenType": ["vc+sd-jwt"],
    "vcConfigId": ["IdentityCredential"],
    "vct": ["https://credentials.example.com/identity_credential"],
    "decoys": [2]
  }
}
```

* : https://github.com/adorsys/keycloak-ssi-deployment/blob/main/signing_service-test-credential.json

Represented as client attributes.

Import/Export Keycloak JSON representation (*)

Hierarchized by “.”(dot).

```
{
  "id": "oid4vci-client",
  "clientId": "oid4vci-client",
  "name": "OID4VC-VCI Client",
  "protocol": "oid4vc",
  "enabled": true,
  "publicClient": true,
  "attributes": {
    "vc.test-credential.expiry_in_s": 100,
    "vc.test-credential.format": "vc+sd-jwt",
    "vc.test-credential.scope": "test-credential",
    "vc.test-credential.vct": "https://credentials.example.com/test-credential",
    "vc.test-credential.credential_signing_alg_values_supported": "ES256,ES384",
    "vc.test-credential.claims": "{\"firstName\":{},\"lastName\":{},\"email\":{}}",
    "vc.test-credential.display.0": "{\"name\": \"Test Credential\"}",

    "vc.IdentityCredential.expiry_in_s": 31536000,
    "vc.IdentityCredential.format": "vc+sd-jwt",
    "vc.IdentityCredential.scope": "identity_credential",
    "vc.IdentityCredential.vct": "https://credentials.example.com/identity_credential",
    "vc.IdentityCredential.cryptographic_binding_methods_supported": "jwk",
    "vc.IdentityCredential.credential_signing_alg_values_supported": "ES256,ES384",
```



Formal JSON representation

```
{
  "credential_configurations_supported": {
    "test-credential": {
      "expiry_in_s": 100,
      "format": "vc+sd-jwt",
      "scope": "test-credential",
      "vct": "https://credentials.example.com/test-credential",
      "credential_signing_alg_values_supported": ["ES256", "ES384"],
      "claims": {
        "firstName": {},
        "lastName": {},
        "email": {}
      },
      "display": {
        "0": {
          "name": "Test Credential"
        }
      }
    },
    "IdentityCredential": {
      "expiry_in_s": 31536000,
      "format": "vc+sd-jwt",
      "scope": "identity_credential",
      "vct": "https://credentials.example.com/identity_credential",
      "cryptographic_binding_methods_supported": "jwk",
      "credential_signing_alg_values_supported": ["ES256", "ES384"],
      "claims": {
```

* : https://github.com/adorsys/keycloak-ssi-deployment/blob/main/signing_service-test-credential.json

OID4VC Client's Credential Configuration(*)

```
{
  "id": "oid4vci-client",
  "clientId": "oid4vci-client",
  "name": "OID4VC-VCI Client",
  "protocol": "oid4vc",
  "enabled": true,
  "publicClient": true,
  "attributes": {
    "vc.test-credential.expiry_in_s": 100,
    "vc.test-credential.format": "vc+sd-jwt",
    "vc.test-credential.scope": "test-credential",
    "vc.test-credential.vct": "https://credentials.example.com/test-credential",
    "vc.test-credential.credential_signing_alg_values_supported": "ES256,ES384",
    "vc.test-credential.claims": "{\"firstName\":{},\"lastName\":{},\"email\":{}}",
    "vc.test-credential.display.0": "{\"name\": \"Test Credential\"}",

    "vc.IdentityCredential.expiry_in_s": 31536000,
    "vc.IdentityCredential.format": "vc+sd-jwt",
    "vc.IdentityCredential.scope": "identity_credential",
    "vc.IdentityCredential.vct": "https://credentials.example.com/identity_credential",
    "vc.IdentityCredential.cryptographic_binding_methods_supported": "jwk",
    "vc.IdentityCredential.credential_signing_alg_values_supported": "ES256,ES384",
    "vc.IdentityCredential.claims": "{\"given_name\":{\"display\":[{\\"name\\":\\"الاسم الشخصي\"}]}",
    "vc.IdentityCredential.display.0": "{\"name\": \"Identity Credential\"}",
    "vc.IdentityCredential.proof_types_supported": "{\"jwt\":{\\"proof_signing_alg_values_sup
  },
}
```

Protocol Mapper (User Attribute) (*)

```
{
  "id": "given_name-mapper-001",
  "name": "given_name-mapper",
  "protocol": "oid4vc",
  "protocolMapper": "oid4vc-user-attribute-mapper",
  "config": {
    "subjectProperty": "given_name",
    "userAttribute": "firstName",
    "supportedCredentialTypes": "identity_credential"
  },
}
```



Digest in a VC

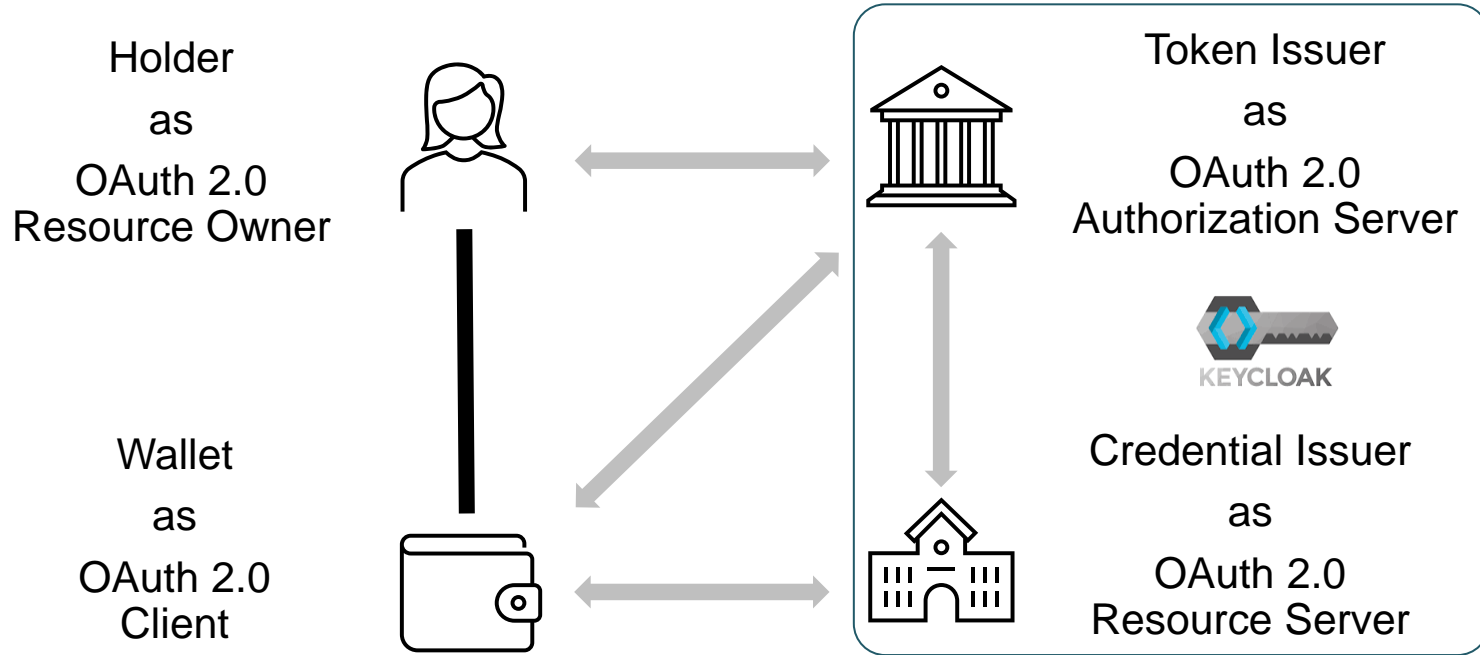
```
WyJfZ002U1R0ZGw0ek5SbE5XMHdiZUVVRiIiwgImZpcnN0TmFtZSIsICJGcmFuY2l1I0
[ "_gM6STtd14zNR1NW0wbeEQ", "firstName", "Francis" ]
```

* : https://github.com/adorsys/keycloak-ssi-deployment/blob/main/signing_service-test-credential.json

0. Registering a user, a key provider, etc.
1. Registering Issuer De-centralized Identifier (DID)
2. Registering a signing service for a verifiable credential (VC)
3. Registering a client for OID4VC
4. Registering the client's credential configurations
5. Registering the client's protocol mappers

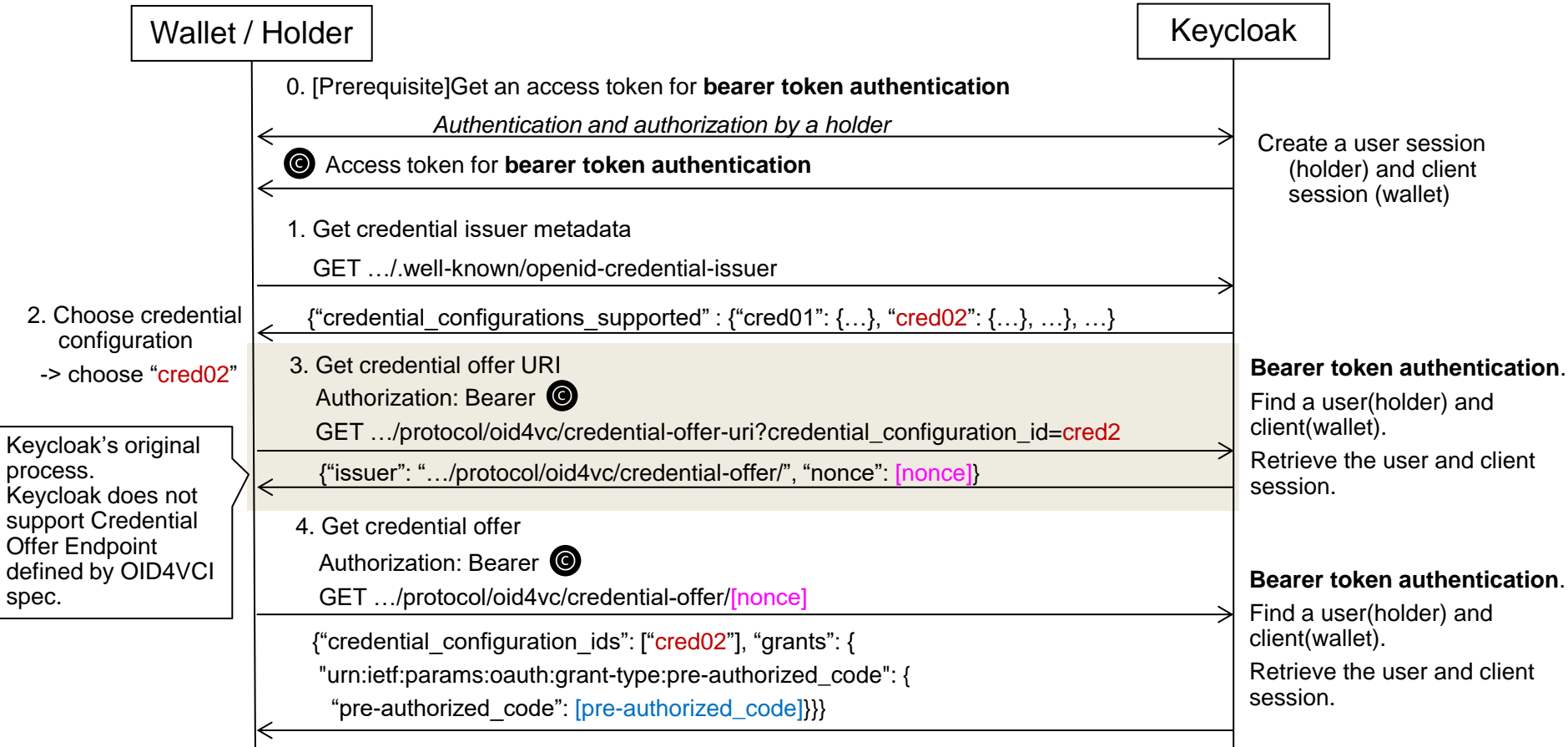
How to register : Admin CLI or Admin REST API direct access
(Admin Console cannot be used)

Pre-Authorization Code Flow & Wallet-initiated & Same-device Credential Offer

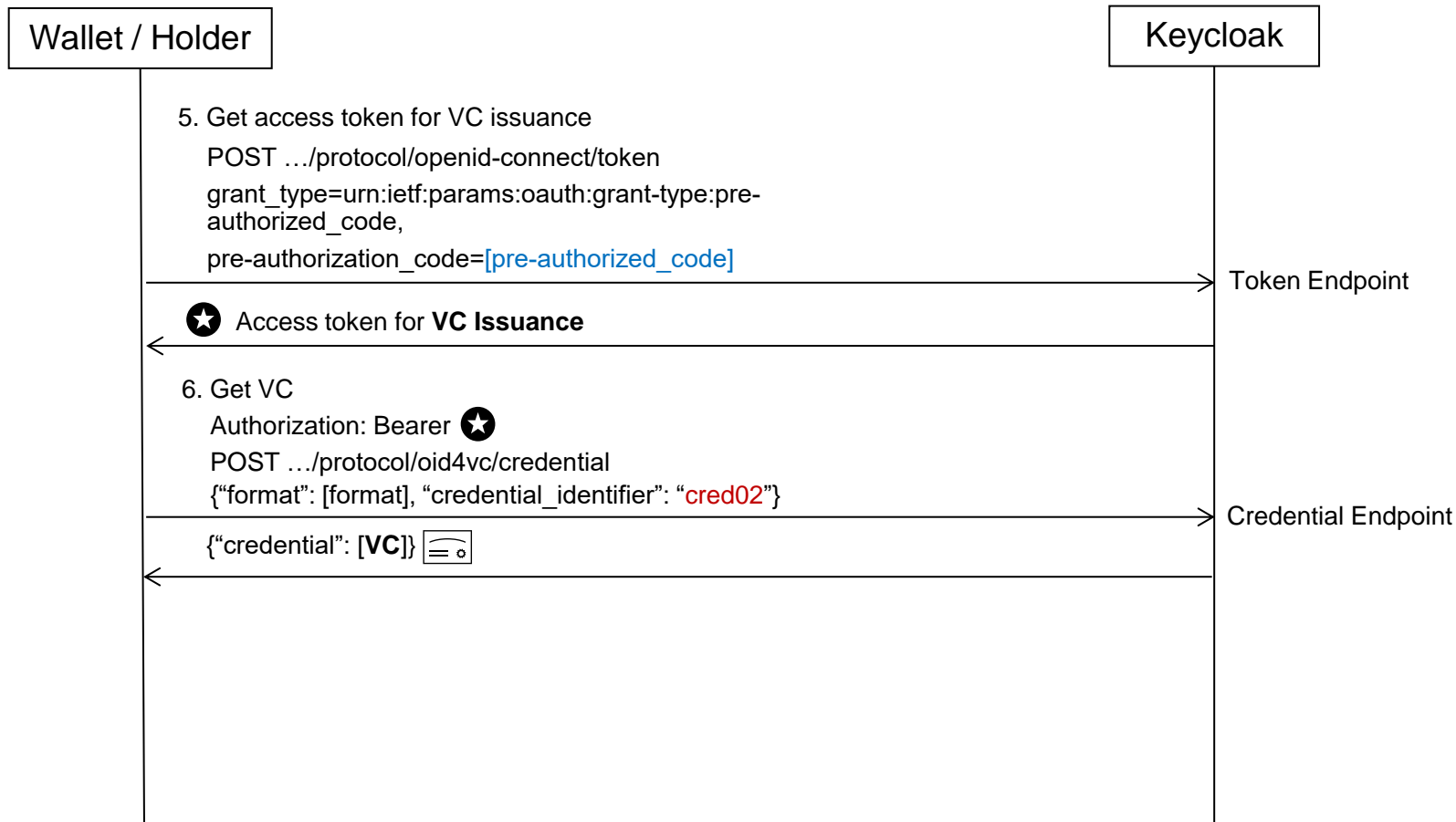


0. [Prerequisite] A wallet makes a holder who is a subject of a VC do authentication and authorization in Keycloak to get an access token.
1. The wallet gets credential issuer metadata to know supported credential configurations.
2. The wallet chooses one of supported credential configurations.
3. The wallet gets a credential offer URI (w/ the credential configuration chosen in step 2 and the access token got in step 0).
4. The wallet gets a credential offer by accessing the credential offer URI in step 4 to receive a pre-authorization code (w/ the access token got in step 0).
5. The wallet gets an access token for the VC in pre-authorization code flow (w/ the pre-authorization code got in step 4).
6. The wallet gets the VC (w/ the access token for the VC got in step 5, and the credential configuration chosen in step 2 as a credential identifier).

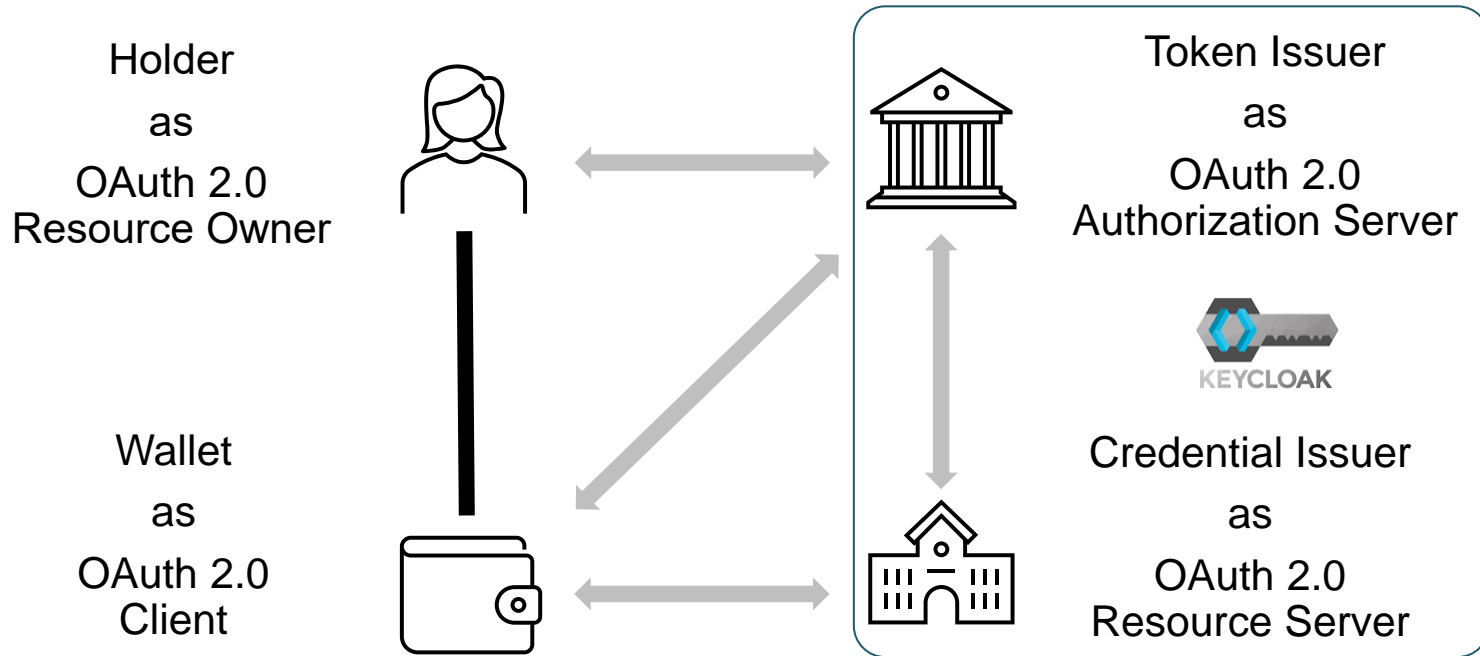
Pre-Authorization Code Flow & Wallet-initiated & Same-device Credential Offer (1 of 2)



Pre-authorization Code Flow & Wallet-initiated & Same-device Credential Offer (2 of 2)



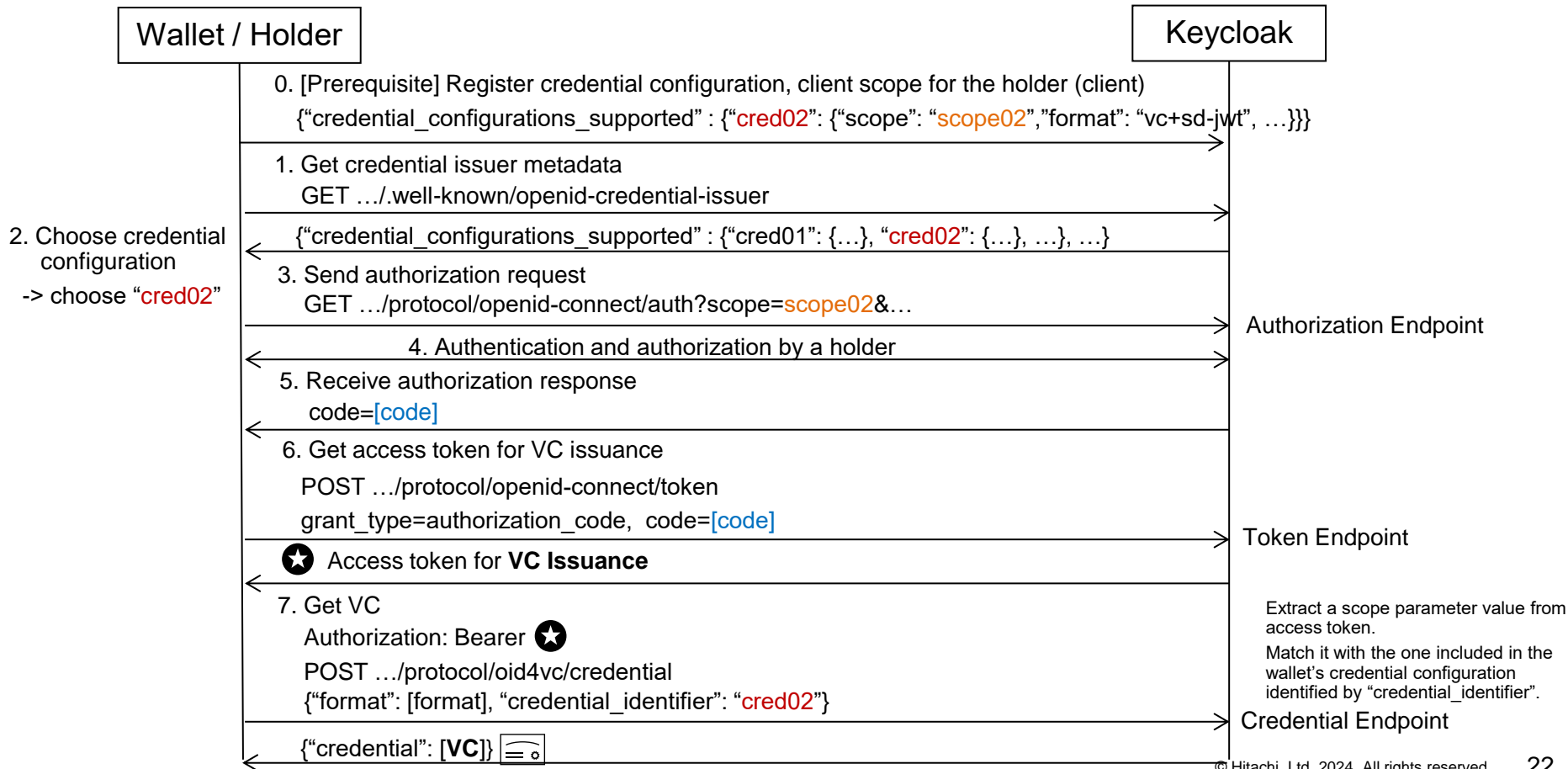
Authorization Code Flow for VC Issuance



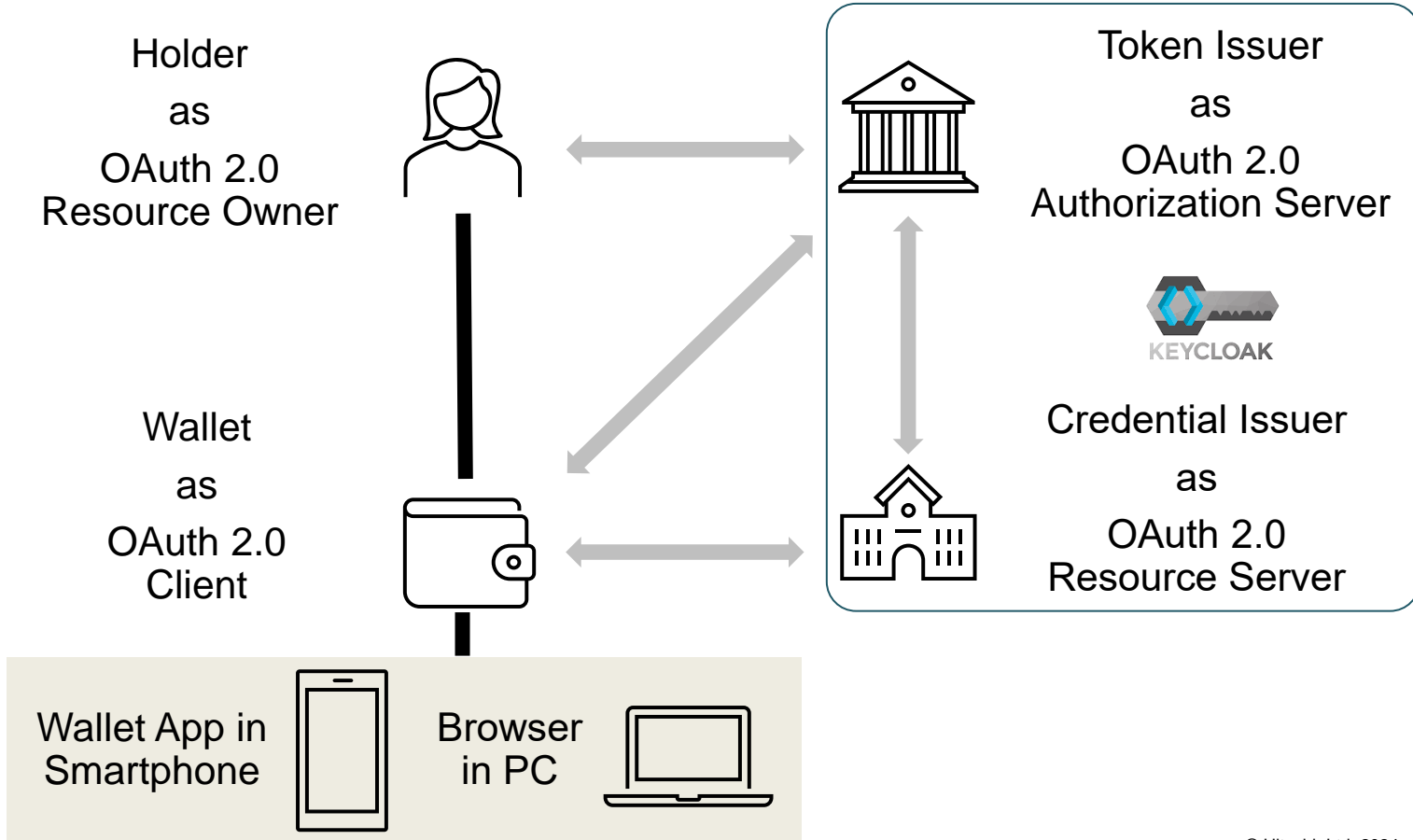
⚠ : It will be supported from Keycloak 26 or later

0. [Prerequisite] An administrator registers client scopes corresponding to “scope” value of supported credential configurations for a wallet and sets them to the wallet (as a client) in Keycloak.
1. The wallet gets credential issuer metadata to know supported credential configurations.
2. The wallet chooses one of supported credential configurations.
3. The wallet send an authorization request to Keycloak (w/ scope parameter value corresponding to a chosen credential configuration in step 2).
4. A holder does authentication and authorization (consent) in Keycloak.
5. The wallet receives an authorization response.
6. The wallet gets an access token for the VC.
7. The wallet gets the VC (w/ the access token for the VC got in step 6, and the credential configuration chosen in step 2 as a credential identifier).

Authorization Code Flow for VC Issuance



Pre-Authorization Code Flow & Wallet-initiated & Cross-device Credential Offer

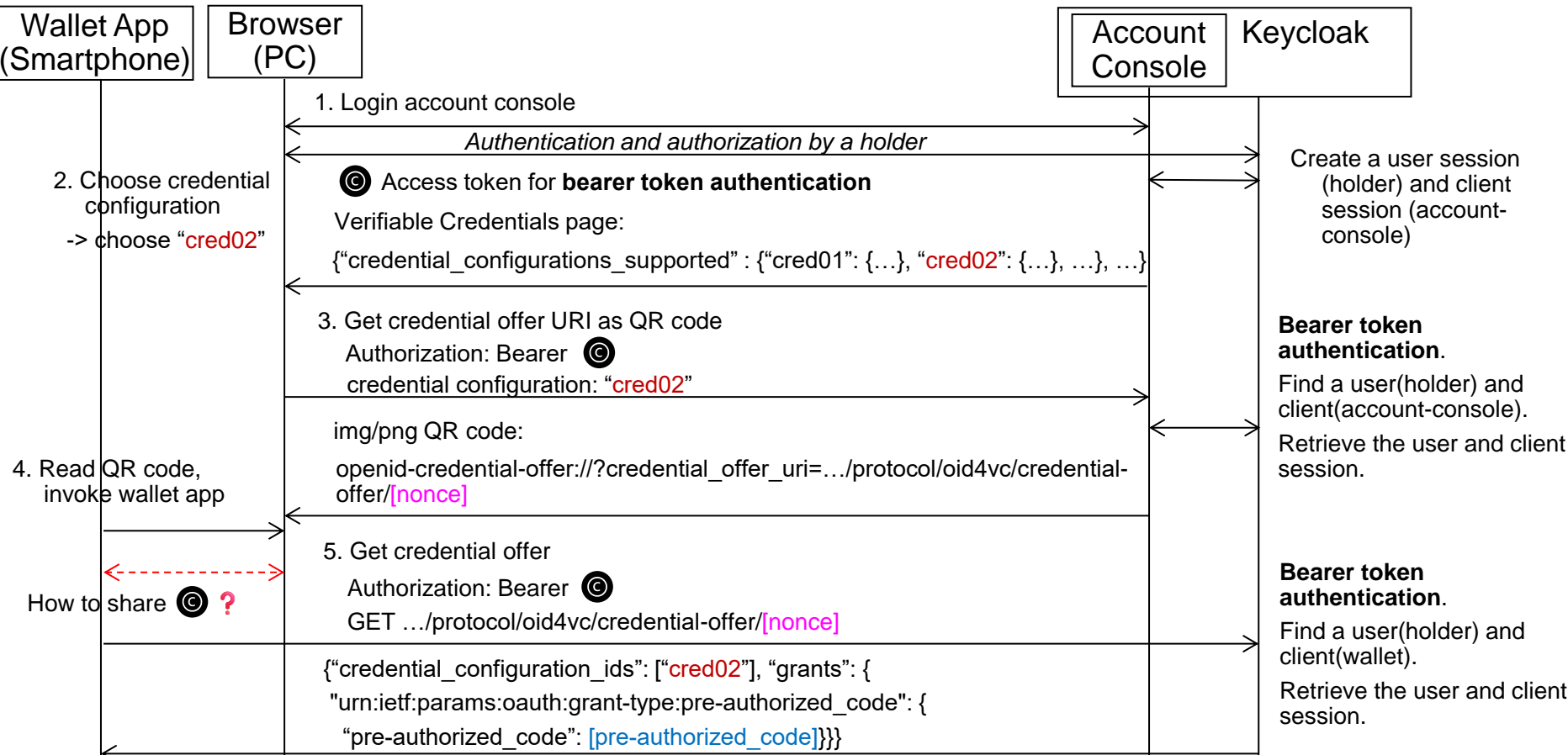


QR code for a credential offer URI on Account Console

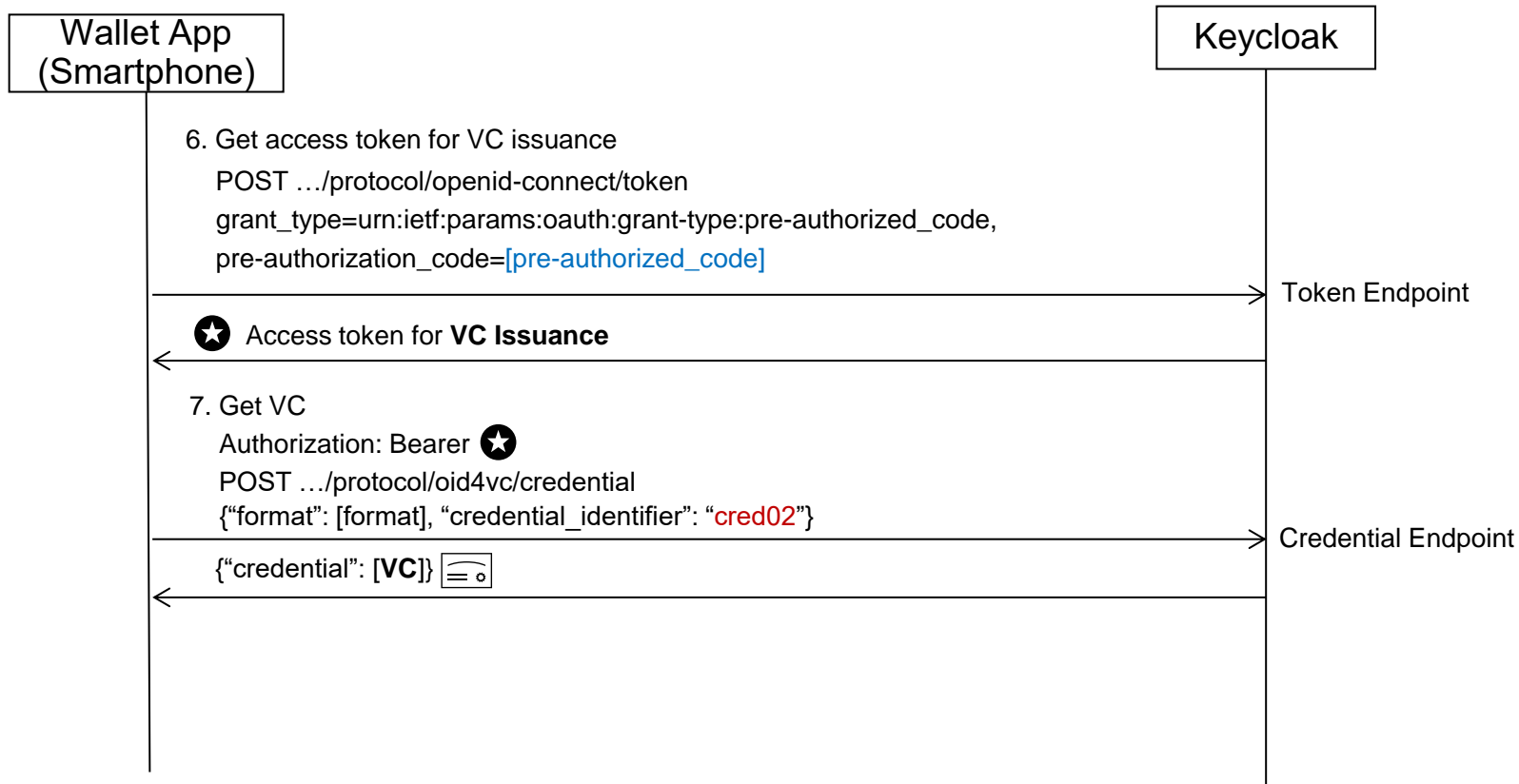
The screenshot displays the 'Verifiable Credentials' section of an account console. On the left is a dark sidebar with navigation options: 'Personal info', 'Account security', 'Applications', and 'Verifiable Credentials' (which is highlighted). The main content area is titled 'Verifiable Credentials' and contains the instruction 'Select the credential for import into your wallet.' Below this is a dropdown menu with 'test-credential' selected. A light gray tooltip is visible, showing two options: 'IdentityCredential' and 'test-credential' (which is selected with a blue checkmark). To the right of the dropdown is a large QR code.

1. Using a browser in a PC, a holder login Keycloak's account console.
2. On the account console, the holder chooses one of supported credential configurations.
3. On the account console, the holder gets credential offer URI shown as QR code (w/ the credential configuration chosen in step 2 and the access token got in step 1). wallet chooses one of supported credential configurations.
4. A smartphone reads the QR code, and invokes an appropriate wallet app.
5. The wallet app a credential offer by accessing the credential offer URI in step 4 to receive a pre-authorization code (w/ the access token got in step 1 ? ? ?).
6. The wallet gets an access token for the VC in pre-authorization code flow (w/ the pre-authorization code got in step 4).
7. The wallet gets the VC (w/ the access token for the VC got in step 5, and the credential configuration chosen in step 2 as a credential identifier.

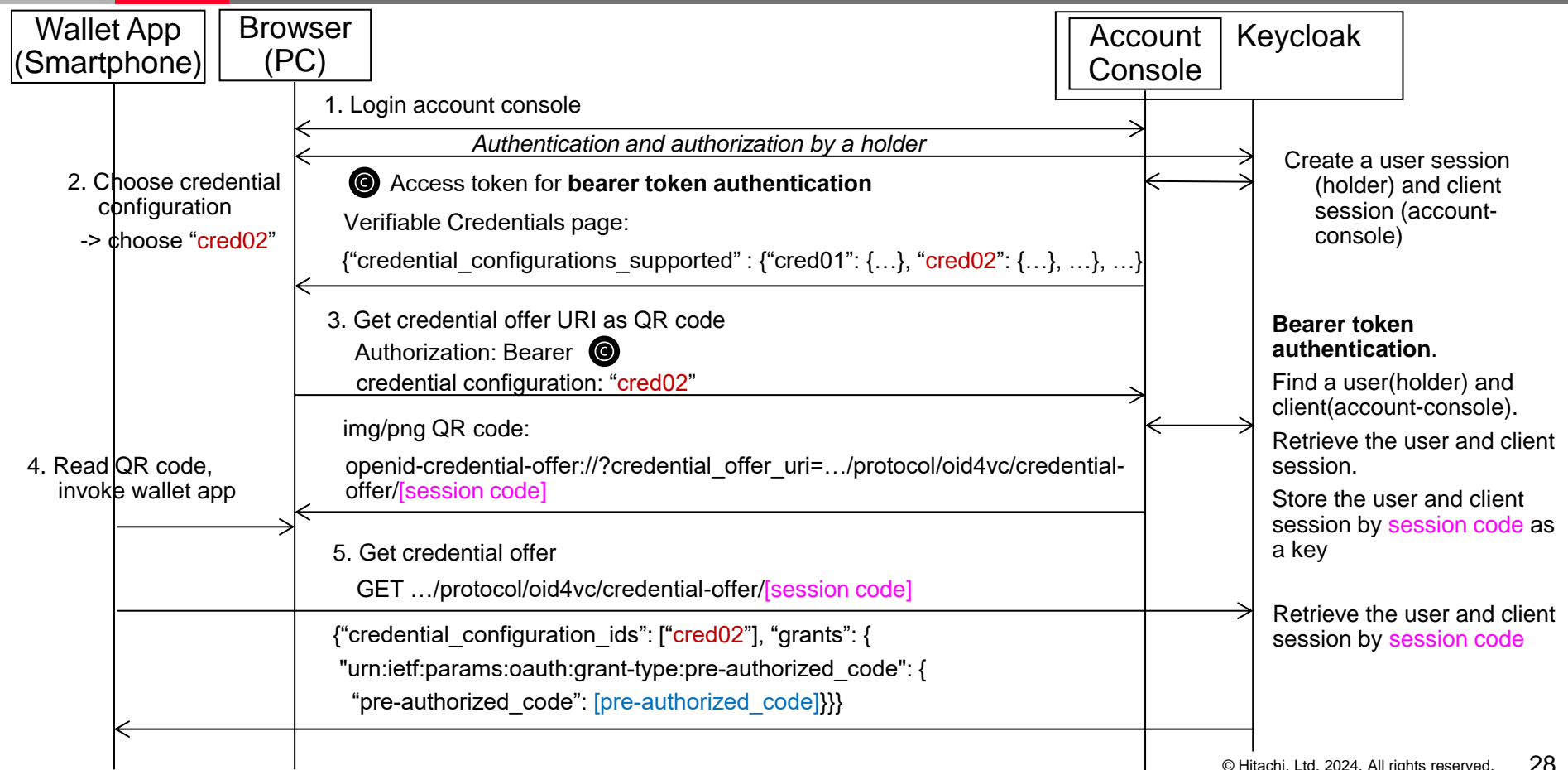
Pre-Authorization Code Flow & Wallet-initiated & Cross-device Credential Offer (1 of 2) (KC25)



Pre-Authorization Code Flow & Wallet-initiated & Cross-device Credential Offer (2 of 2) (KC25)



Pre-Authorization Code Flow & Wallet-initiated & Cross-device Credential Offer (1 of 2) (KC26?)



■ Goal: Keycloak can work as an issuer of VCs.

- Phase 1: supported as an experimental feature
- Phase 2: supported as a preview feature
- Phase 3: supported officially

Completed by KC 25

Nov 2023 - Jun 2024

In Progress

Jun 2024 -

■ SIG : OAuth SIG

Mainly supports OAuth and its related specifications to Keycloak.

GitHub repository : <https://github.com/keycloak/kc-sig-fapi> (*1)

CNCF slack channel : **#keycloak-oauth-sig**

■ Epic issue : Support OpenID for Verifiable Credentials(OID4VC)

- <https://github.com/keycloak/keycloak/issues/25936>

■ Discussion: OpenID for Verifiable Credential Issuance

- <https://github.com/keycloak/keycloak/discussions/17616>

■ Design: OpenID Verifiable for Credential Issuance

- <https://github.com/keycloak/keycloak-community/blob/main/design/OID4VCI.md>

■ Guide: OpenID Verifiable for Credential Issuance

- <https://github.com/adorsys/keycloak-ssi-deployment>

■ Current focus points

- How to determine which VC is issued
 - Client-based to Scope-based
- Where we define the credentials
 - Per Client (client attributes) to Per Realm (protocol mapper's configuration or realm attributes)
- “Protocol” attribute of a client for OID4VCI
 - Different Protocol (oid4vc) to Same Protocol (oidc)
Different Protocol : oidc client for OAuth2/OIDC while oid4vc client for OID4VCI
Same Protocol : oidc client for both OAuth2/OIDC and OID4VCI
KC25 implementation : Pre-authorization code flow : protocol = oid4vc
Authorization code flow : protocol = oidc
- VC Format
 - mDL (ISO.18013-5)
- VC Issuance Proof (Key Binding)
 - jwt (SD-JWT)
- Follow the latest version of OID4VCI specification (Implementer's Draft)
 - Draft version 13 (KC 25 followed) to 14 (current ver), ...

OID4VC Support - General

| | |
|---|--------------------------------|
| OID4VCI-supported Keycloak's version | 26.0.0 or later |
| OID4VCI support feature level | Experimental |
| Referred Version of OID4VCI specification | Implementer's Draft (draft 14) |

OID4VC Management/Administration

| | |
|------------------------------|---|
| Admin REST API direct access | ✓ |
| Admin CLI (kcadm) | ✓ |
| Admin Console | ✗ |

VC Issuance Flow

| | |
|--------------------------|-----|
| Pre-Authorized Code Flow | ✓ |
| Authorized Code Flow | ⚠ ✓ |

Authorization Request Parameter

| | |
|--------------------------------------|---|
| scope | ✓ |
| authorization_details (RFC 9396 RAR) | ✗ |
| issuer_state | ✗ |

VC Issuance Variation

| | |
|-------------------------------|---|
| Immediate Credential Issuance | ✓ |
| Deferred Credential Issuance | ✗ |
| Batch Credential Issuance | ✗ |

VC Credential Offer

| | |
|--------------|---|
| Same-Device | ✓ |
| Cross-Device | ? |

VC Format

| | |
|-------------------|---|
| SD-JWT VC (IETF) | ✓ |
| JWT VC (DIF) | ? |
| LDP VC (W3C) | ? |
| mDL (ISO.18013-5) | ✗ |

VC Issuance Proof (Key Binding)

| | |
|---------------|---|
| jwt (SD-JWT) | ✗ |
| cwt | ✗ |
| ldp_vp (VCDM) | ✗ |

⚠ : will be supported from Keycloak 26 or later

2. Compliance with Security Specifications

Keycloak supported specifications in the following categories:

- OAuth 2.0
by Internet Engineering Task Force (IETF)
- OpenID Connect (OIDC)
by OpenID Foundation
- Financial-grade API Security Profiles (FAPI)
by OpenID Foundation
- Open Banking Security Profiles
by some country's regulatory body
- Security Assertion Markup Language 2.0 (SAMLv2)
by Organization for the Advancement of Structured Information Standards (OASIS)
- User Managed Access (UMA)
by Kantara Initiative
- Web Authentication API (WebAuthn)
by World Wide Web Consortium (W3C)
- The Federal Information Processing Standard Publication 140-2 (FIPS 140-2)
by U.S. National Institute of Standards and Technology (NIST)

? What “supporting a specification” means?
! Implementing features as the specification describes.

? How can we check whether Keycloak implements features as the specification describes?

! Passing a **conformance test** for a specification.

Some standardization body provides a conformance test for a specification.

! **Certified** by a standardization body that defines the specification.

Some standardization body certifies a product with the specification.

Ex. Certification Program

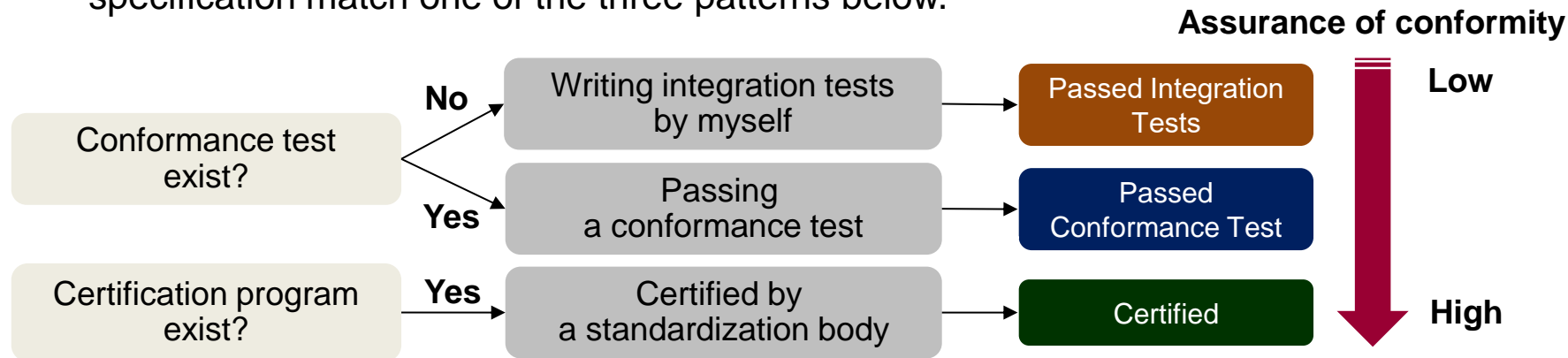
- NIST : FIPS 140-2,3
- FIDO Alliance : FIDO2 (WebAuthn, CTAP2)
- OIDF : OIDC and its related specifications, FAPI, Open Banking security profiles

? How can we check that when no conformance test exists?

! ... writing **self-integration tests** for the specification and passing them.



In this slide, I decided that Keycloak supports a specification if Keycloak and the specification match one of the three patterns below.



What this slide describes are **not** Keycloak project's official opinion. These were derived by just my investigation against the latest version of Keycloak (25.0.1).

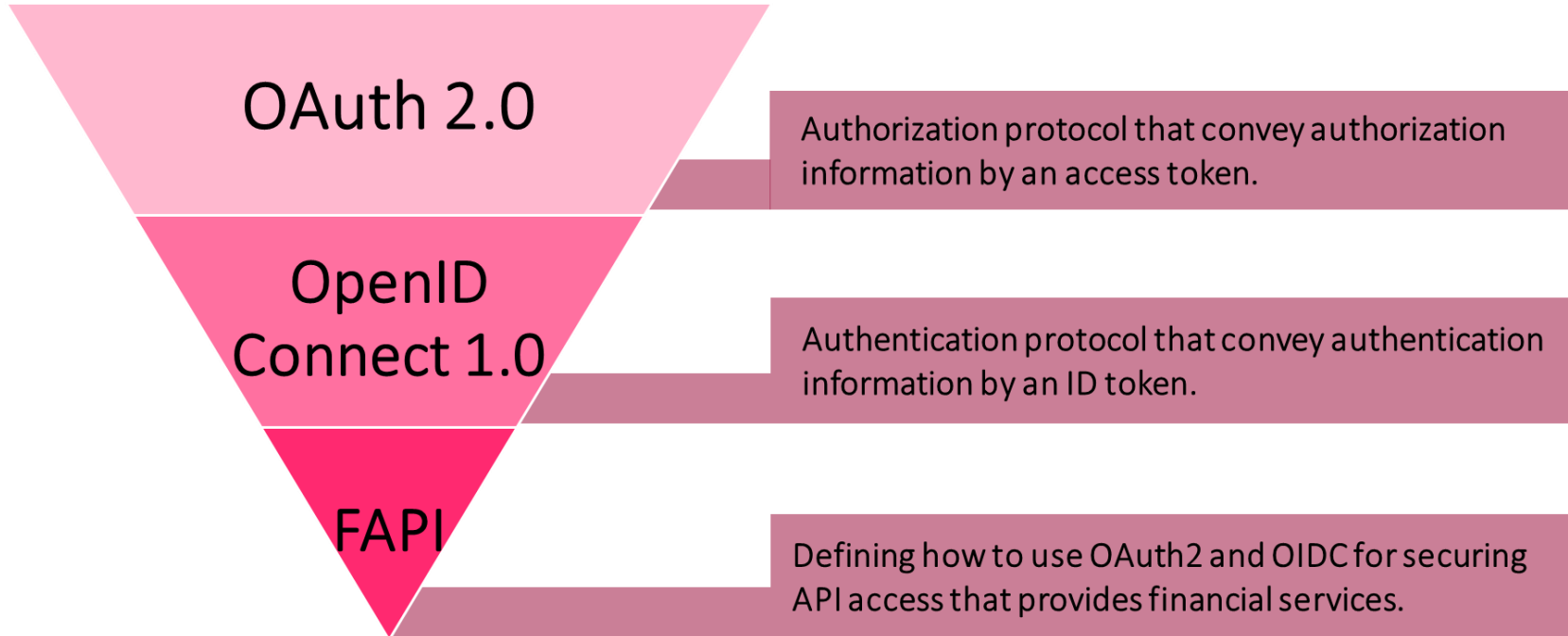


“Supporting a specification” does **not** mean that Keycloak supports **all** features of the specification. Especially, in “Passed Integrations Tests”.



“Supporting a specification” does **not** mean that Keycloak officially supports features of the specification. Some supports of specifications are treated as “**Preview**” or “**Experimental**” features.

- Hardening OAuth 2.0 authorization and OpenID Connect 1.0 authentication protocol. It is standardized by OpenID Foundation.





[UK : OpenBanking]

OpenBanking Financial Grade API (FAPI) Profile

OpenBanking CIBA Profile



[Australia : Consumer Data Right (CDR)]

Consumer Data Right Security Profile



[Brazil : Open Banking Brazil]

Open Banking/Finance Brazil Financial-grade API Security Profile



[Saudi Arabia : SAMA/KSA Open Banking]

OpenBanking Financial Grade API (FAPI) Profile



Standardization Body: OpenID Foundation (OIDF)

Certified

| # | Specification | WG | Status |
|----|--|--------|--------|
| 1 | OpenID Connect Core 1.0 | OIDC | Final |
| 2 | OpenID Connect Discovery 1.0 | OIDC | Final |
| 3 | OpenID Connect Dynamic Client Registration 1.0 | OIDC | Final |
| 4 | OAuth 2.0 Multiple Response Type Encoding Practices | OIDC | Final |
| 5 | OAuth 2.0 Form Post Response Mode | OIDC | Final |
| 6 | OpenID Connect RP-Initiated Logout 1.0 | OIDC | Final |
| 7 | OpenID Connect Session Management 1.0 | OIDC | Final |
| 8 | OpenID Connect Front-Channel Logout 1.0 | OIDC | Final |
| 9 | OpenID Connect Back-Channel Logout 1.0 | OIDC | Final |
| 10 | OpenID Connect Client Initiated Backchannel Authentication Flow - Core 1.0 | MODRNA | Final |
| 11 | Financial-grade API Security Profile 1.0 - Part 1: Baseline | FAPI | Final |
| 12 | Financial-grade API Security Profile 1.0 - Part 2: Advanced | FAPI | Final |
| 13 | JWT Secured Authorization Response Mode for OAuth 2.0 (JARM) | FAPI | Final |

Standardization Body: OpenID Foundation (OIDF)



| # | Specification | | WG | Status |
|----|--|--------------------------|------|---------------------|
| 14 | OpenID for Verifiable Credential Issuance | Passed Integration Tests | OIDC | Implementer's Draft |
| 15 | Financial-grade API: Client Initiated Backchannel Authentication Profile | Certified | FAPI | Implementer's Draft |
| 16 | FAPI 2.0 Security Profile | Passed Conformance Test | FAPI | Implementer's Draft |
| 17 | FAPI 2.0 Message Signing | Passed Conformance Test | FAPI | Draft |





“Supporting a specification” does **not** mean that Keycloak officially supports features of the specification. Some supports of specifications are treated as “**Preview**” or “**Experimental**” features.

Ex. support for #14 “OpenID for Verifiable Credential Issuance” is treated as “**Experimental**” feature by the latest version of Keycloak (25.0.1).

Standardization Body: OpenID Foundation (OIDF)














| # |  Specification |  Conformance Profile | Status | Version |
|----|---|---|-----------|---------------|
| 1 | OpenID Connect Core 1.0 | Basic OP | Certified | 2.3.0, 18.0.0 |
| 2 | OpenID Connect Core 1.0 | Implicit OP | Certified | 2.3.0, 18.0.0 |
| 3 | OpenID Connect Core 1.0 | Hybrid OP | Certified | 2.3.0, 18.0.0 |
| 4 | OpenID Connect Core 1.0 | Config OP | Certified | 2.3.0, 18.0.0 |
| 5 | OpenID Connect Core 1.0 | Dynamic OP | Certified | 2.3.0, 18.0.0 |
| 6 | OpenID Connect Core 1.0 | Form Post OP | Certified | 18.0.0 |
| 7 | OpenID Connect Core 1.0 | 3rd Party-Init OP | Not Yet | - |
| 8 | OpenID Connect RP-Initiated Logout 1.0 | RP-Initiated OP | Certified | 18.0.0 |
| 9 | OpenID Connect Session Management 1.0 | Session OP | Certified | 18.0.0 |
| 10 | OpenID Connect Front-Channel Logout 1.0 | Front-Channel OP | Certified | 18.0.0 |
| 11 | OpenID Connect Back-Channel Logout 1.0 | Back-Channel OP | Certified | 18.0.0 |

Standardization Body: OpenID Foundation (OIDF)







| # |  Specification |  Conformance Profile | Status | Version |
|----|---|---|-----------|---------|
| 1 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ MTLS | Certified | 15.0.2 |
| 2 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ MTLS, PAR | Certified | 15.0.2 |
| 3 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ Private Key | Certified | 15.0.2 |
| 4 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ Private Key, PAR | Certified | 15.0.2 |
| 5 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ MTLS, JARM | Certified | 15.0.2 |
| 6 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ Private Key, JARM | Certified | 15.0.2 |
| 7 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ MTLS, PAR, JARM | Certified | 15.0.2 |
| 8 | FAPI 1.0 - Advanced | FAPI Adv. OP w/ Private Key, PAR, JARM | Certified | 15.0.2 |
| 9 | FAPI-CIBA Profile | FAPI-CIBA OP poll w/ MTLS | Certified | 15.0.2 |
| 10 | FAPI-CIBA Profile | FAPI-CIBA OP poll w/ Private Key | Certified | 15.0.2 |
| 11 | FAPI-CIBA Profile | FAPI-CIBA OP Ping w/ MTLS | Certified | 15.0.2 |
| 12 | FAPI-CIBA Profile | FAPI-CIBA OP Ping w/ Private Key | Certified | 15.0.2 |

Keycloak-supported specifications : Open Banking

Standardization Body: OpenID Foundation (OIDF)

| # |  Specification |  Conformance Profile | Status | Version |
|---|---|---|------------------|---------|
|  | 1 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ MTLS | Certified | 15.0.2 |
|  | 2 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ Private Key | Certified | 15.0.2 |
|  | 3 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ MTLS, PAR | Certified | 15.0.2 |
|  | 4 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ Private Key, PAR | Certified | 15.0.2 |
|  | 5 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ MTLS, JARM | Certified | 15.0.2 |
|  | 6 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ Private Key, JARM | Certified | 15.0.2 |
|  | 7 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ MTLS, PAR, JARM | Certified | 15.0.2 |
|  | 8 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP w/ Private Key, PAR, JARM | Certified | 15.0.2 |
|  | 9 Brazil Open Banking (FAPI 1.0 - Advanced) | BR-OB Adv. OP DCR | Not Yet | - |
|  | 10 Australia CDR (FAPI 1.0 - Advanced) | AU-CDR Adv. OP w/ Private Key | Certified | 15.0.2 |
|  | 11 Australia CDR (FAPI 1.0 - Advanced) | AU-CDR Adv. OP w/ Private Key, PAR | Certified | 15.0.2 |

Standardization Body: OpenID Foundation (OIDF)

| # |  Specification |  Conformance Profile | Status | Version |
|---|---|--|--------------------|---------|
|  | 1 UK Open Banking (FAPI 1.0 - Advanced) | UK-OB Adv. OP w/ MTLs | Test Passed | 20.0.0 |
|  | 2 UK Open Banking (FAPI 1.0 - Advanced) | UK-OB Adv. OP w/ Private Key | Test Passed | 20.0.0 |
| | 3 FAPI 2.0 Security Profile Second & Message Signing | FAPI2SP MTLs + MTLs | Test Passed | 23.0.0 |
| | 4 FAPI 2.0 Security Profile Second & Message Signing | FAPI2SP private key + MTLs | Test Passed | 23.0.0 |
| | 5 FAPI 2.0 Security Profile Second & Message Signing | FAPI2SP OpenID Connect | Test Passed | 23.0.0 |
| | 6 FAPI 2.0 Security Profile Second & Message Signing | FAPI2MS JAR | Test Passed | 23.0.0 |
| | 7 FAPI 2.0 Security Profile Second & Message Signing | FAPI2MS JARM | Test Passed | 23.0.0 |
|  | 8 Brazil Open Finance (FAPI-BR v2) | BR-OF Adv. OP w/ Private Key, PAR | Test Passed | 23.0.1 |
|  | 9 Brazil Open Finance (FAPI-BR v2) | BR-OF Adv. OP DCR | Not Yet | - |

Standardization Body: Internet Engineering Task Force (IETF)

Passed Integration Tests

| # | Specification | Status |
|----|---|--------|
| 1 | RFC 6749: The OAuth 2.0 Authorization Framework | RFC |
| 2 | RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage | RFC |
| 3 | RFC 7009: OAuth 2.0 Token Revocation | RFC |
| 4 | RFC 7521: Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants | RFC |
| 5 | RFC 7523: JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants | RFC |
| 6 | RFC 7591: OAuth 2.0 Dynamic Client Registration Protocol | RFC |
| 7 | RFC 7592: OAuth 2.0 Dynamic Client Registration Management Protocol | RFC |
| 8 | RFC 7636: Proof Key for Code Exchange by OAuth Public Clients | RFC |
| 9 | RFC 7662: OAuth 2.0 Token Introspection | RFC |
| 10 | RFC 8414: OAuth 2.0 Authorization Server Metadata | RFC |
| 11 | RFC 8628: OAuth 2.0 Device Authorization Grant | RFC |
| 12 | RFC 8693: OAuth 2.0 Token Exchange | RFC |
| 13 | RFC 8705: OAuth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens | RFC |

Standardization Body: Internet Engineering Task Force (IETF)

Passed Integration Tests

| # | Specification | Status |
|----|--|----------------|
| 14 | RFC 9101: The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR) | RFC |
| 15 | RFC 9126: OAuth 2.0 Pushed Authorization Requests | RFC |
| 16 | RFC 9207: OAuth 2.0 Authorization Server Issuer Identification | RFC |
| 17 | RFC 9449: Demonstration of Proof-of-Possession at the Application Layer (DPoP) | RFC |
| 18 | The OAuth 2.1 Authorization Framework | Internet Draft |



“Supporting a specification” does **not** mean that Keycloak officially supports features of the specification. Some supports of specifications are treated as “**Preview**” or “**Experimental**” features.

Ex. support for #12 “RFC 8693: OAuth 2.0 Token Exchange” and #17 “RFC 9449: Demonstration of Proof-of-Possession at the Application Layer (DPoP)” are treated as “**Preview**” feature by the latest version of Keycloak (25.0.1).

Standardization Body: Kantara Initiative

Passed Integration Tests

| # | Specification | Status |
|---|---|----------------|
| 1 | User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization (version 2.0) | Recommendation |
| 2 | Federated Authorization for User-Managed Access (UMA) 2.0 | Recommendation |

Standardization Body: Organization for the Advancement of Structured Information Standards (OASIS)

Passed Integration Tests

| # | Specification | Status |
|---|---|-----------|
| 1 | Security Assertion Markup Language 2.0 (SAML 2.0) | Published |

Standardization Body: World Wide Web Consortium (W3C)

Passed Integration Tests

| # | Specification | Status |
|---|---|----------------|
| 1 | Web Authentication: An API for accessing Public Key Credentials Level 2 | Recommendation |

Standardization Body: U.S. National Institute of Standards and Technology (NIST)

Passed Integration Tests

| # | Specification | Status |
|---|--|-----------|
| 1 | The Federal Information Processing Standard Publication 140-2 (FIPS 140-2) | Published |

- Keycloak community activity : OAuth SIG (Special Interest Group) mainly supports OAuth and its related specifications to Keycloak.
GitHub repository : <https://github.com/keycloak/kc-sig-fapi> (*1)
CNCF slack channel : **#keycloak-oauth-sig**
OAuth SIG works on security standards in this talk
(Passkeys, OAuth 2.1, DPoP, OID4VCI, etc.)
- Whenever a new version of Keycloak is released, OAuth SIG runs conformance tests for all OIDF's specifications that Keycloak has already supported against it.
- OAuth SIG publishes the conformance test run results:
<https://github.com/keycloak/kc-sig-fapi?tab=readme-ov-file#passed-conformance-tests-per-keycloak-version>

- By supporting OID4VCI, Keycloak could be used as a Credential Issuer in EU Digital Identity Wallet ecosystem.
- Keycloak 25 implemented OID4VCI, but its feature is treated as an experimental feature and have many limitations.
- Keycloak community OAuth SIG continue working on refining OID4VCI support.
- Keycloak supported a lot of security specifications, but we need to take care that what “supporting a specification” means.
- As for specifications like OIDC and FAPI that their standardization body provides their conformance tests, Keycloak community OAuth SIG runs regression tests for them against newly released version of Keycloak.

| # | Specification | Standardization Body | Status | Support Lv. by KC | Conformance Test exist? | Certificate Program exist? | Self-Integration Test Passed? | Conformance Test Passed? | Certified? |
|----|---|----------------------|----------------|-------------------|-------------------------|----------------------------|-------------------------------|--------------------------|------------|
| 1 | RFC 6749: The OAuth 2.0 Authorization Framework | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 2 | RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 3 | RFC 7009: OAuth 2.0 Token Revocation | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 4 | RFC 7521: Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 5 | RFC 7523: JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 6 | RFC 7591: OAuth 2.0 Dynamic Client Registration Protocol | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 7 | RFC 7592: OAuth 2.0 Dynamic Client Registration Management Protocol | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 8 | RFC 7636: Proof Key for Code Exchange by OAuth Public Clients | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 9 | RFC 7662: OAuth 2.0 Token Introspection | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 10 | RFC 8414: OAuth 2.0 Authorization Server Metadata | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 11 | RFC 8628: OAuth 2.0 Device Authorization Grant | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 12 | RFC 8693: OAuth 2.0 Token Exchange | IETF (OAuth WG) | RFC | Preview | - | - | ✓ | - | - |
| 13 | RFC 8705: OAuth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 14 | RFC 9101: The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR) | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 15 | RFC 9126: OAuth 2.0 Pushed Authorization Requests | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 16 | RFC 9207: OAuth 2.0 Authorization Server Issuer Identification | IETF (OAuth WG) | RFC | Supported | - | - | ✓ | - | - |
| 17 | RFC 9449: Demonstration of Proof-of-Possession at the Application Layer (DPoP) | IETF (OAuth WG) | RFC | Preview | - | - | ✓ | - | - |
| 18 | The OAuth 2.1 Authorization Framework | IETF (OAuth WG) | Internet Draft | Supported | - | - | ✓ | - | - |

| # | Specification | Standardization Body | Status | Support Lv. by KC | Conformance Test exist? | Certificate Program exist? | Self-Integration Test Passed? | Conformance Test Passed? | Certified? |
|----|---|----------------------|---------------------|-------------------|-------------------------|----------------------------|-------------------------------|--------------------------|------------|
| 19 | OpenID Connect Core 1.0 | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 20 | OpenID Connect Discovery 1.0 | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 21 | OpenID Connect Dynamic Client Registration 1.0 | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 22 | OAuth 2.0 Multiple Response Type Encoding Practices | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 23 | OAuth 2.0 Form Post Response Mode | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 24 | OpenID Connect RP-Initiated Logout 1.0 | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 25 | OpenID Connect Session Management 1.0 | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 26 | OpenID Connect Front-Channel Logout 1.0 | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 27 | OpenID Connect Back-Channel Logout 1.0 | OIDF (OIDC WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 28 | OpenID for Verifiable Credential Issuance | OIDF (OIDC WG) | Implementer's Draft | Experimental | - (coming soon?) | - (coming soon?) | ✓ | - | - |
| 29 | OpenID Connect Client Initiated Backchannel Authentication Flow - Core 1.0 | OIDF (MODRNA WG) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 30 | Financial-grade API Security Profile 1.0 - Part 1: Baseline | OIDF (OIDC FAPI) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 31 | Financial-grade API Security Profile 1.0 - Part 2: Advanced | OIDF (OIDC FAPI) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 32 | JWT Secured Authorization Response Mode for OAuth 2.0 (JARM) | OIDF (OIDC FAPI) | Final | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 33 | Financial-grade API: Client Initiated Backchannel Authentication Profile | OIDF (OIDC FAPI) | Implementer's Draft | Supported | ✓ | ✓ | ✓ | ✓ | ✓ |
| 34 | FAPI 2.0 Security Profile | OIDF (OIDC FAPI) | Implementer's Draft | Supported | ✓ | ✓ | ✓ | ✓ | ✗ |
| 35 | FAPI 2.0 Message Signing | OIDF (OIDC FAPI) | Draft | Supported | ✓ | ✓ | ✓ | ✓ | ✗ |
| 36 | The Federal Information Processing Standard Publication 140-2 (FIPS 140-2) | NIST | Published | Supported | ✓ | ✓ | ✓ | ? | ? |
| 37 | Web Authentication: An API for accessing Public Key Credentials Level 2 | W3C | Recommendation | Supported | ✓ | ✓ | ✓ | ✗ | ✗ |
| 38 | User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization (version 2.0) | Kantara Initiative | Recommendation | Supported | ? | ? | ✓ | ? | ? |
| 39 | Federated Authorization for User-Managed Access (UMA) 2.0 | Kantara Initiative | Recommendation | Supported | ? | ? | ✓ | ? | ? |
| 40 | Security Assertion Markup Language 2.0 (SAML 2.0) | OASIS | Published | Supported | ? | ? | ✓ | ? | ? |

- OpenID is a trademark or registered trademark of OpenID Foundation in the United States and other countries.
- GitHub is a trademark or registered trademark of GitHub, Inc. in the United States and other countries.
- Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and other countries.
- X is a trademark or registered trademark of X CORP. in the United States and other countries.
- Facebook is a trademark or registered trademark of Meta Platforms, Inc. in the United States and other countries.
- Other brand names and product names used in this material are trademarks, registered trademarks, or trade names of their respective holders.

END

Keycloak's Updates on Emerging Paradigm of Identity and Compliance with Security Specifications

September 19, 2024

Takashi Norimatsu

OSS Solution Center

Hitachi, Ltd.



Hitachi Social Innovation is
POWERING GOOD