FAPI-SIG Annex

OIDC Client’s Public Keys Management Support

PROPOSED DRAFT

Table of Contents

Motivation

Background

There is the case that a client app uses its keys of public-key cryptography.

<Signature>

- Application -
  - Client Authentication by JWS Client Assertion
  - Authorization Request with signed Request Object
- Algorithm (JWA “alg”) -

EC

ECDSA : ES256, ES384, ES512

RSA

RSASSA-PSS : PS256, PS384, PS512

RSASSA-PKCS1-v1_5 : RS256, RS384, RS512

<Encryption>

- Application -
  - CEK Encryption for ID Token Encryption
- Algorithm (JWA “alg”) -

Key Encryption

RSA

RSAES-OAEP : RSA_OAEP

RSAES-PKCS1-v1_5 : RSA1_5

RSA PKCS v1_5 based algorithms are yet considered not to be secure.

EC or RSA PSS,OAEP based algorithms are still considered to be secure.

→ EC or RSA PSS,OAEP based algorithms are preferable.

Background

The current keycloak(v12) gets and manages the client app's public key as follows.

<Key loading methods>

- By Reference Dynamically

  - Download client app's public keys from the endpoint whose url is specified as "jwks_uri" OAuth2 Client Metadata.

    Key Format : JWKS

- By Value Statically

  - Import the client app's certificate or public key in advance.

    Key Format : JKS, PKCS12, Certificate PEM, Public Key PEM, JWKS

  - Generate and register the client app's certificate in advance.

    Key Format : JKS, PKCS12

<Key loading preference>

1. By reference
2. By value (both methods are mutually exclusive which means only either one of them is set up)

Background

The current keycloak(v12)'s client app's public key management support is as follows.

Signature			Key Loading Method	
			By Reference	By Value
Signature Algorithm	ECDSA	ES256	X	
		ES384	X	
		ES512	X	
	RSASSA-PSS	PS256	X	
		PS384	X	
		PS512	X	
	RSASSA-PKCS1-v1_5	RS256	X	X
		RS384	X	
		RS512	X	

Encryption			Key Loading Method	
			By Reference	By Value
Key Encryption Algorithm	RSAES-OAEP	RSA-OAEP	X	
	RSAES-PKCS1-v1_5	RSA1_5	X	

Key Use (JWK "use")		Key Loading Method	
		By Reference	By Value
Key Use	"sig"	X	X
	"enc"	X	

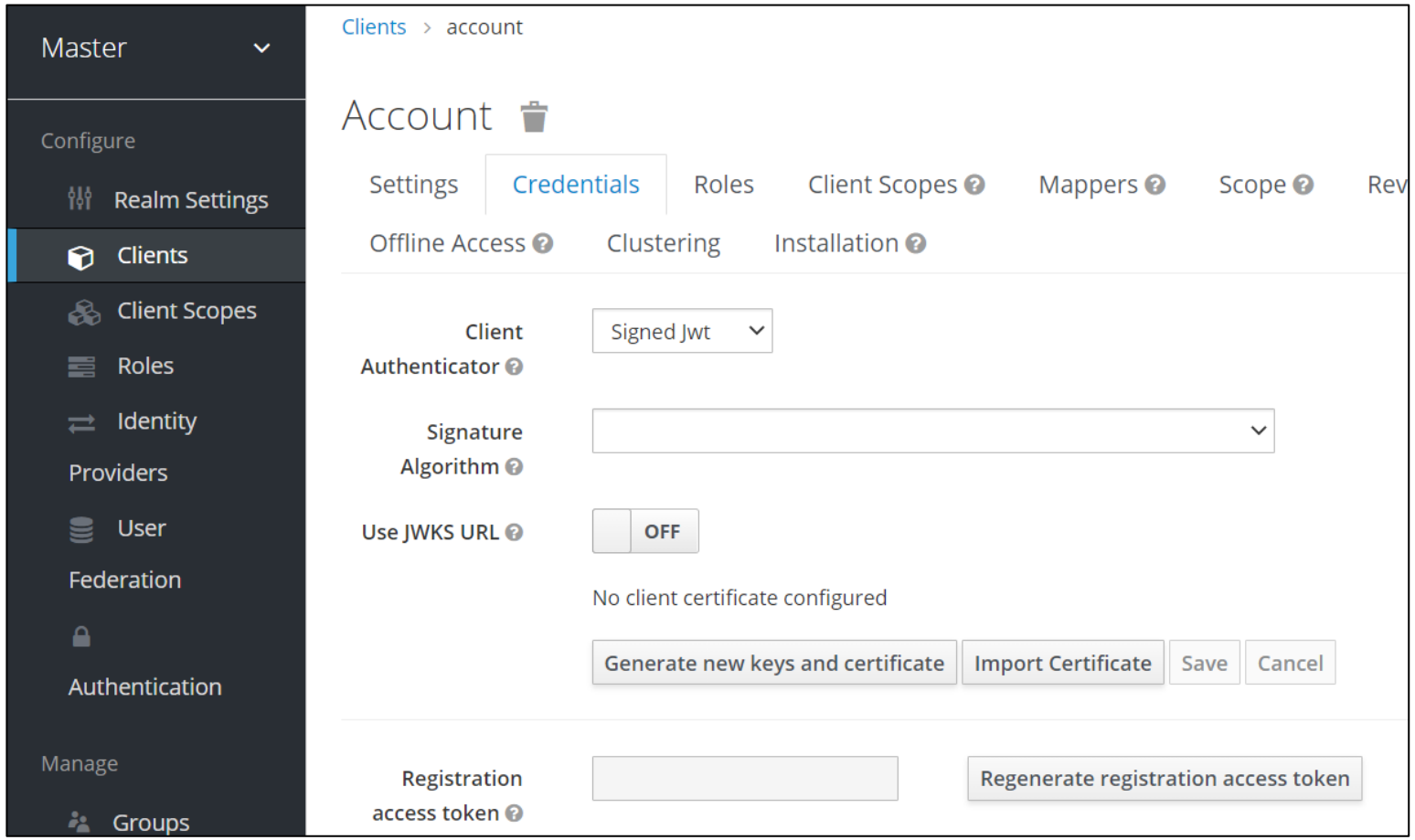
Key Type (JWK "kty")		Key Loading Method	
		By Reference	By Value
Key Type	EC	X	
	RSA	X	X

PROPOSED DRAFT

Background

[client public key loading settings]

Key loading setting is tightly coupled with the one for “Signed Jwt” Client Authentication that should be independent.



1. On Admin Console, go to “Clients -> Credentials” tab, select Signed Jwt from “Client Authenticator” pulldown menu.
2. Set “ON” to “Use JWKS URL” switch.
3. Put the URL mentioned above onto “JWKS URL” textbox.

The client app can change the client authentication method set in “Client Authenticator” pulldown menu.

Issues and Goals

For Signature

[Issues and Goals]

<Signature>

Assume that the client app wants use algorithms other than RSASSA-PKCS1-v1_5 (considering security, or ECDSA choice for efficient computation) …

- [#01] The client app needs to take the option “by reference” as the key loading method which means it needs to provide its dedicated endpoint and publish its public key.

➡ The option “by value” also supports ECDSA, RSASSA-PSS.

For Encryption

[Issues and Goals]

<Encryption>

Assume that the client app wants encrypt ID token…

- [#02] The client app needs to take the option “by reference” as the key loading method which means it needs to provide its dedicated endpoint and publish its public key.

➡ The option “by value” also supports encryption usage.

For Key Loading Settings

[Issues and Goals]

<Key Loading Settings>

- [#03] Keys can not be registered as JWK by value.

➡ Support “jwks” OAuth2 Client Metadata to register keys.

- [#04] It is not straightforward to set up properly the keycloak’s public key loading feature.

Key loading setting is tightly coupled with the one for “Signed Jwt” Client Authentication.

➡ Make it more user-friendly, straightforward to use.

Goals in Details

Signature			Key Loading Method	
			By Reference	By Value
Signature Algorithm	ECDSA	ES256	X	X
		ES384	X	X
		ES512	X	X
	RSASSA-PSS	PS256	X	X
		PS384	X	X
		PS512	X	X
	RSASSA-PKCS1-v1_5	RS256	X	X
		RS384	X	X
		RS512	X	X

Key Type (JWK "kty")		Key Loading Method	
		By Reference	By Value
Key Type	EC	X	X
	RSA	X	X

[Goals in detail]

<Signature>

[#01] The client app needs to take the option "by reference" as the key loading method which means it needs to provide its dedicated endpoint and publish its public key.

The "by value" key loading method becomes to support

- EC key type
- ES256, ES284, ES512, PS256, PS384, PS512, RS384, RS512 signature algorithms.

Goals in Details

Encryption			Key Loading Method	
			By Reference	By Value
Key Encryption Algorithm	RSAES-OAEP	RSA-OAEP	X	X
	RSAES-PKCS1-v1_5	RSA1_5	X	X

Key Use (JWK "use")		Key Loading Method	
		By Reference	By Value
Key Use	"sig"	X	X
	"enc"	X	X

[Goals in detail]

<Encryption>

[#02] The client app needs to take the option "by reference" as the key loading method which means it needs to provide its dedicated endpoint and publish its public key.

The "by value" key loading method becomes to support

- "enc" key use
- RSA-OAEP, RSA1_5 key encryption algorithms.

Goals in Details

[Goals in detail]

<Encryption>

[#03] Keys can not be registered as JWK by value.

The “by value” key loading method becomes to support

- Register the client app’s public keys by “jwks” OAuth2 Client Metadata

  Registering Method :

  - Via Dynamic Client Registration (as “jwks” parameter)
  - Via Admin REST API (as realm attribute parameter)

  Key Format : JWK

Goals in Details

[Goals in detail]

<Encryption>

[#04] Keys can not be registered as JWK by value. It is not straightforward to set up properly the keycloak's public key loading feature.

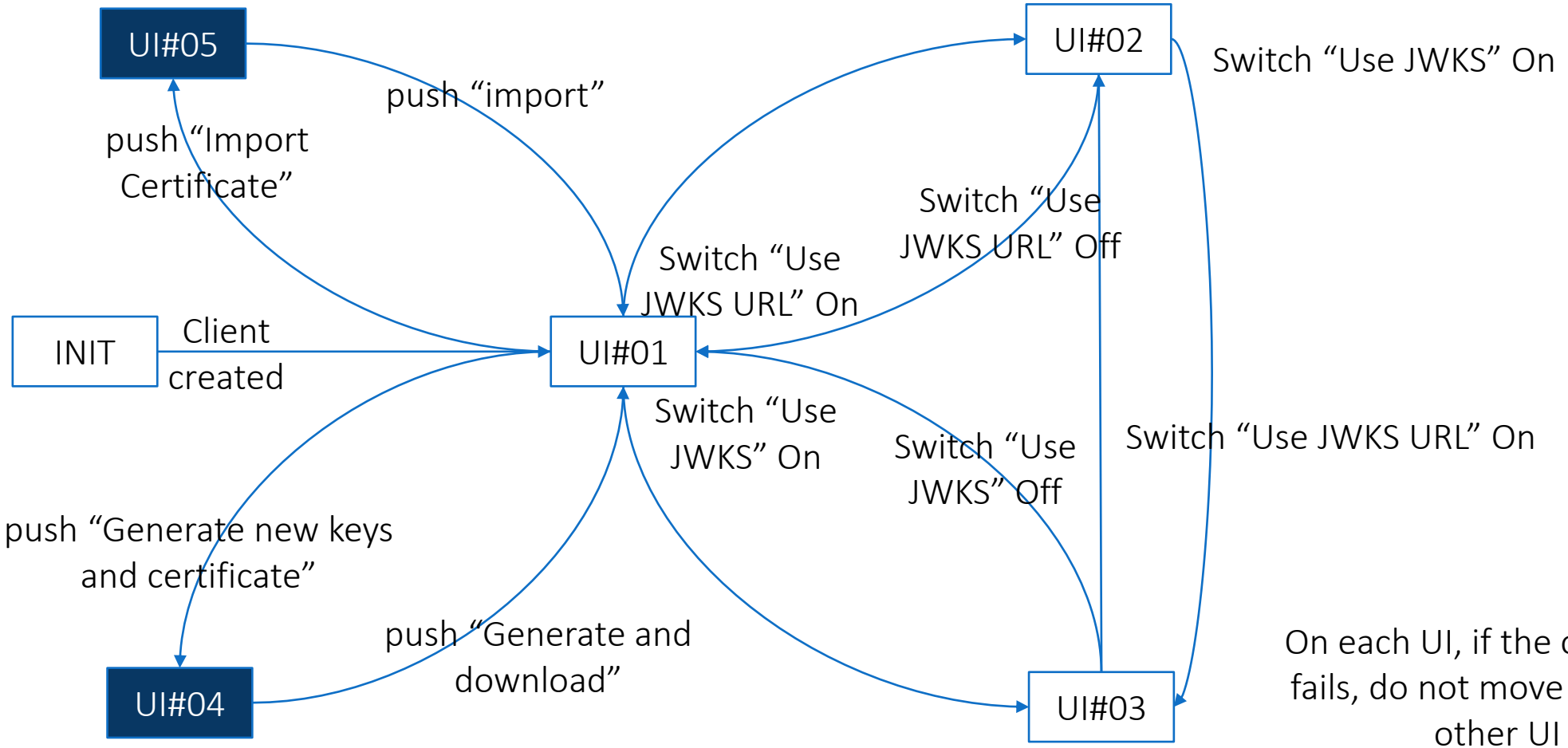
Key loading setting is tightly coupled with the one for "Signed Jwt" Client Authentication.

Prepare dedicated UI for managing client app's public keys as Clients->Keys tab.

Remove the client app's public key loading setting from current Clients->Confidential Tab.

Goals : UI

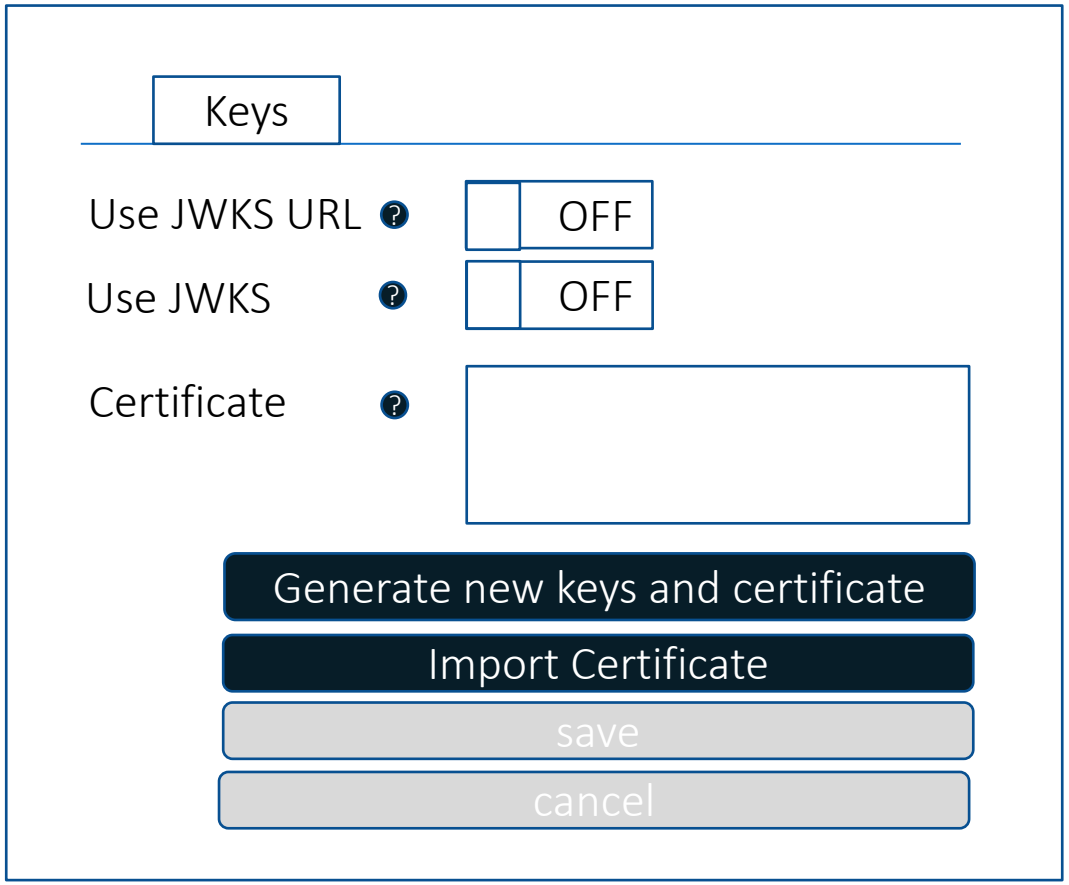
[UI action and transition for new Clients->Keys Tab]



On each UI, if the operation fails, do not move onto the other UI

Goals : UI

[UI#01]

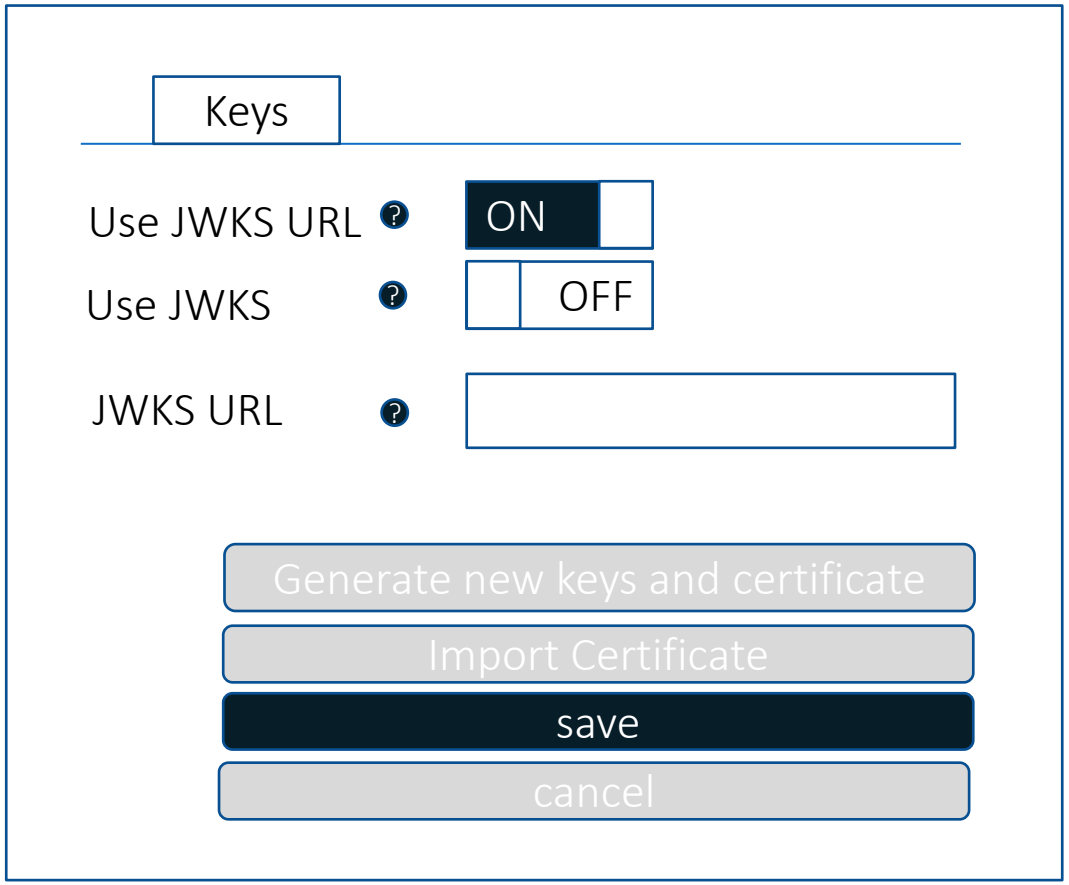


This UI is for the following existing key loading setting.

- By Value Statically
  - Import the client app's certificate or public key in advance.
  - Generate and register the client app's certificate in advance.

Goals : UI

[UI#02]

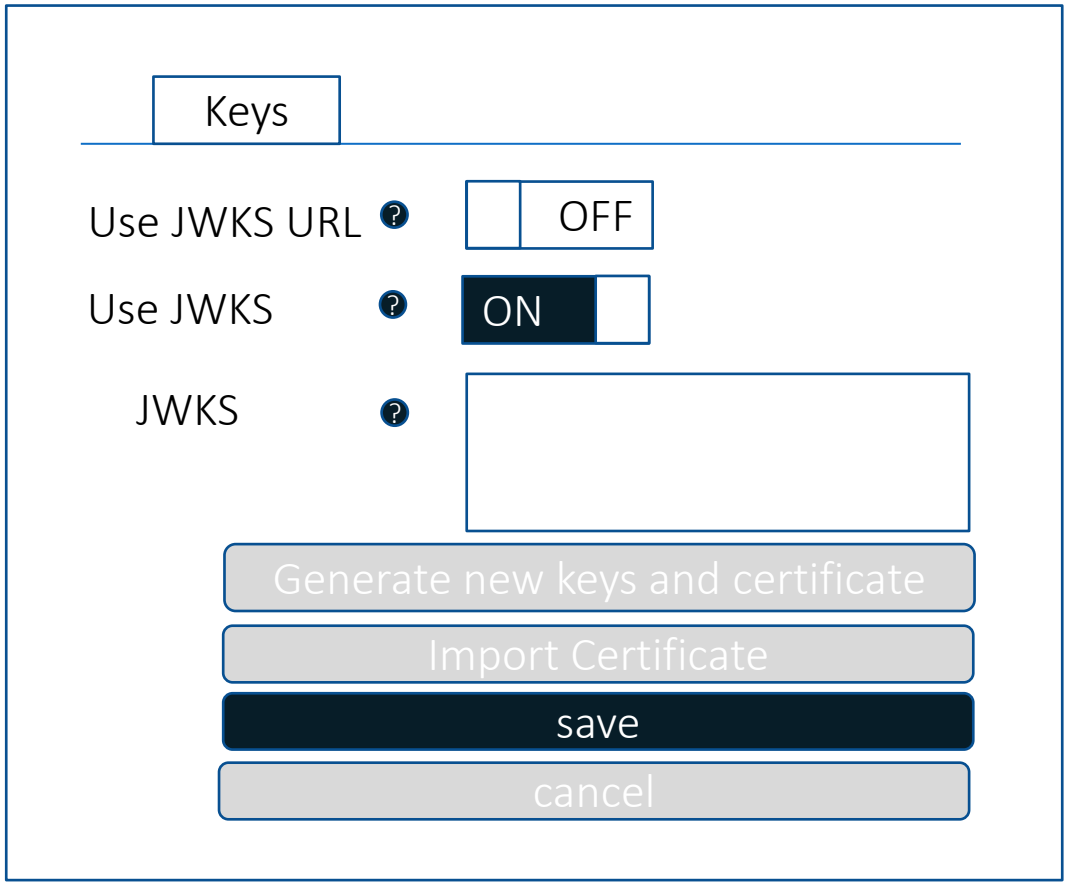


This UI is for the following existing key loading setting.

- By Reference Dynamically
  - Download client app's public keys from the endpoint whose url is specified as "jwks_uri" OAuth2 Client Metadata.

Goals : UI

[UI#03]



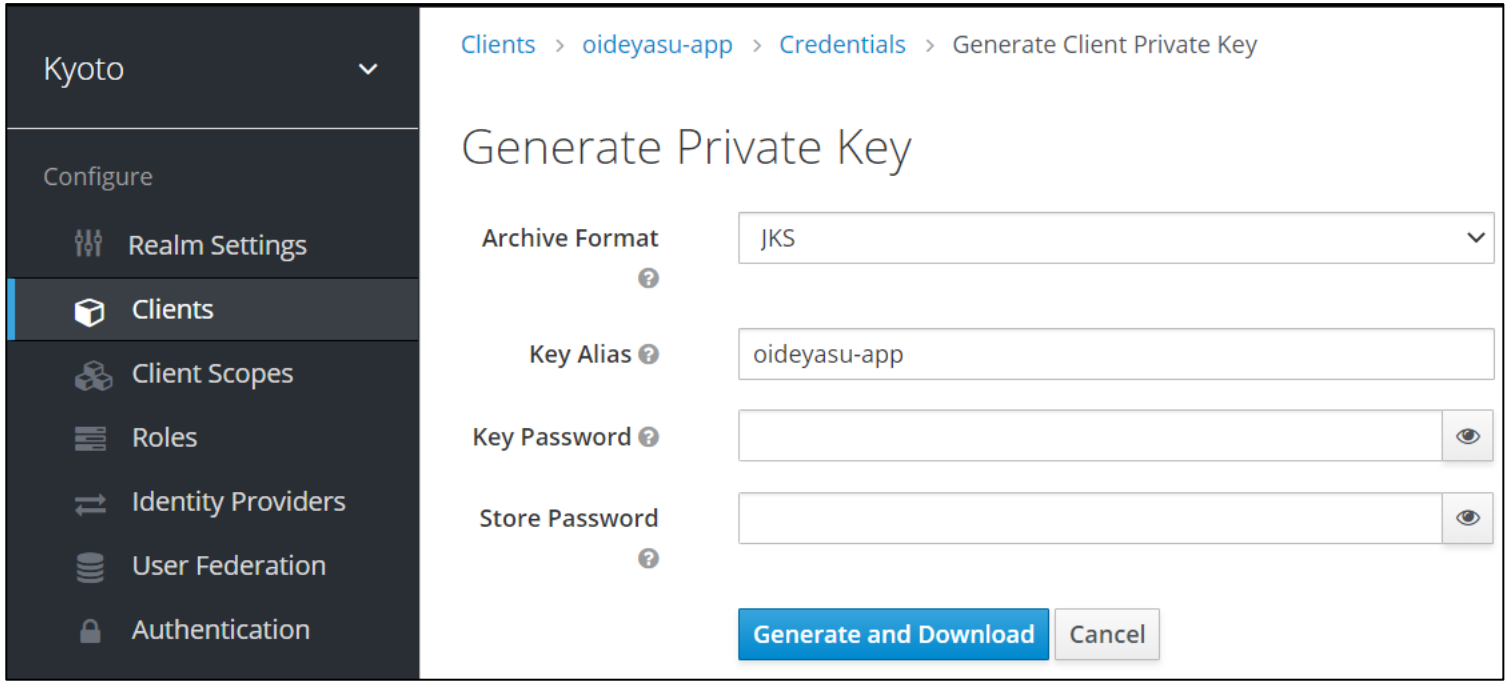
This UI is for the following existing key loading setting.

- By Value Statically

- Register the client app's public keys by "jwks" OAuth2 Client Metadata
  Registering Method :
  - Via Dynamic Client Registration (as "jwks" parameter)
  - Via Admin REST API (as realm attribute parameter)
  Key Format : JWK

Goals : UI

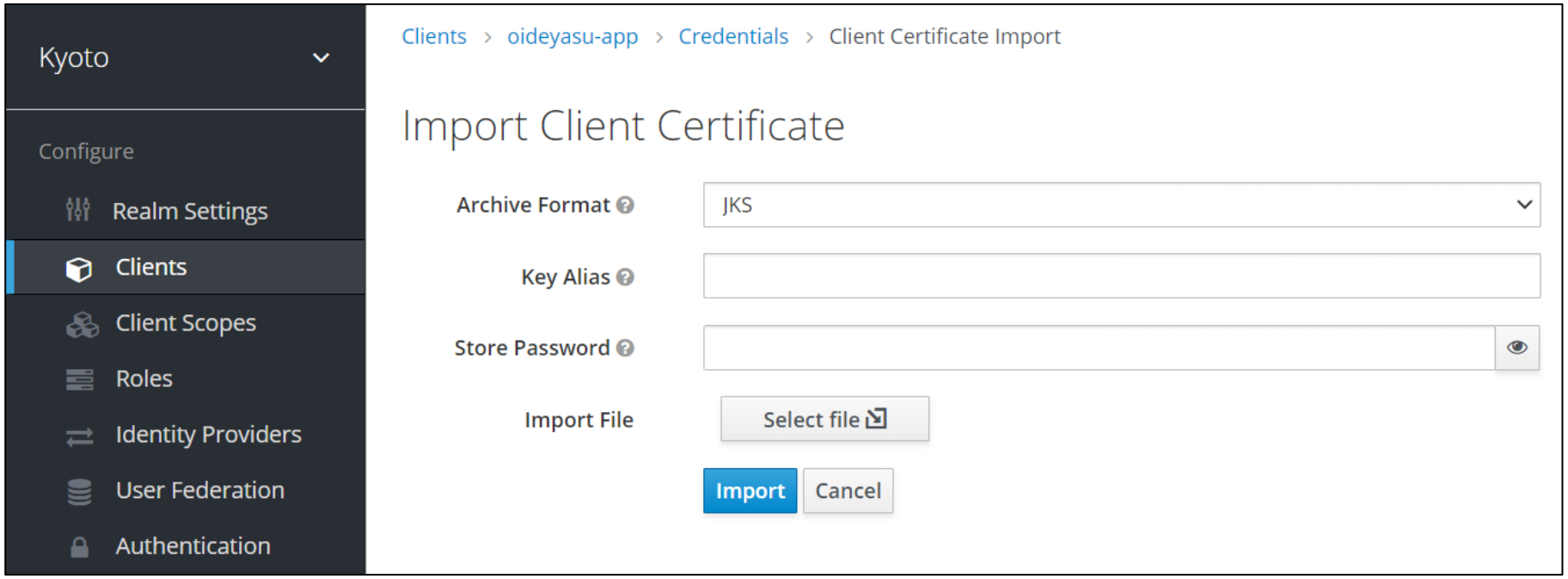
[UI#04]



The same as the current Clients->Credentials Tab

Goals : UI

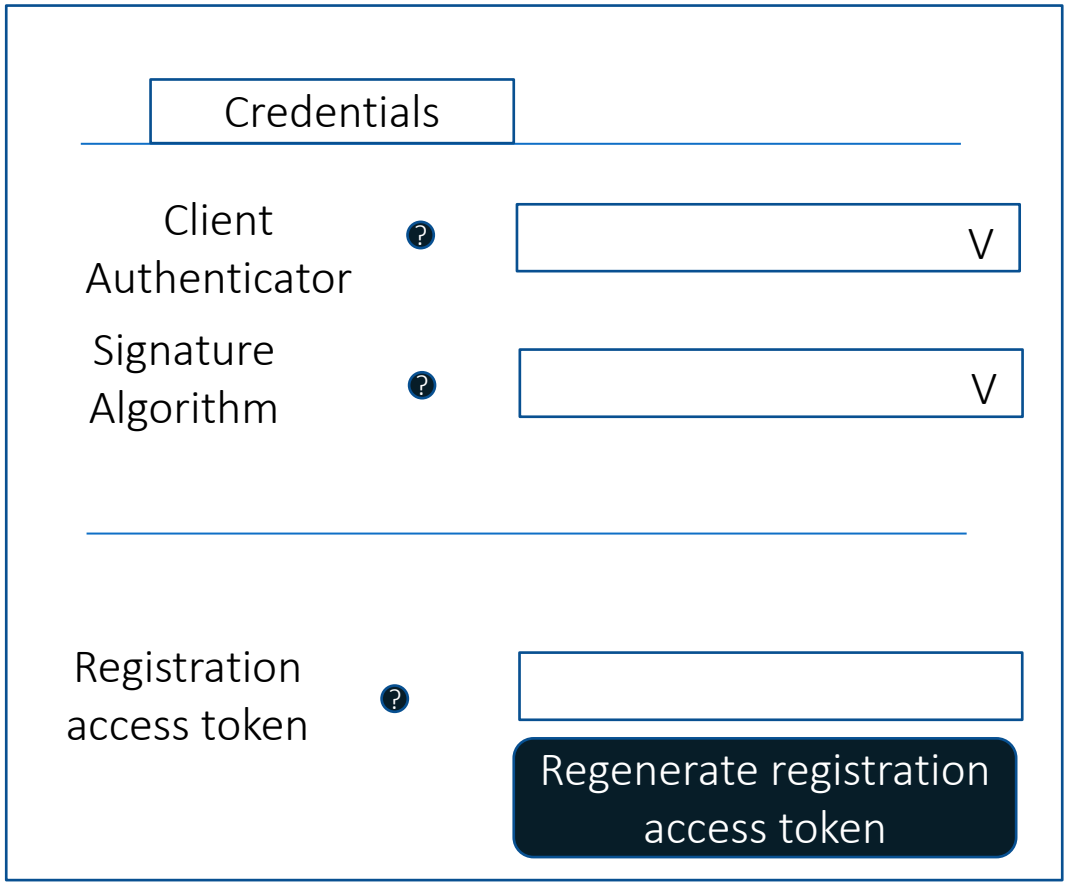
[UI#05]



The same as the current Clients->Credentials Tab

Goals : UI

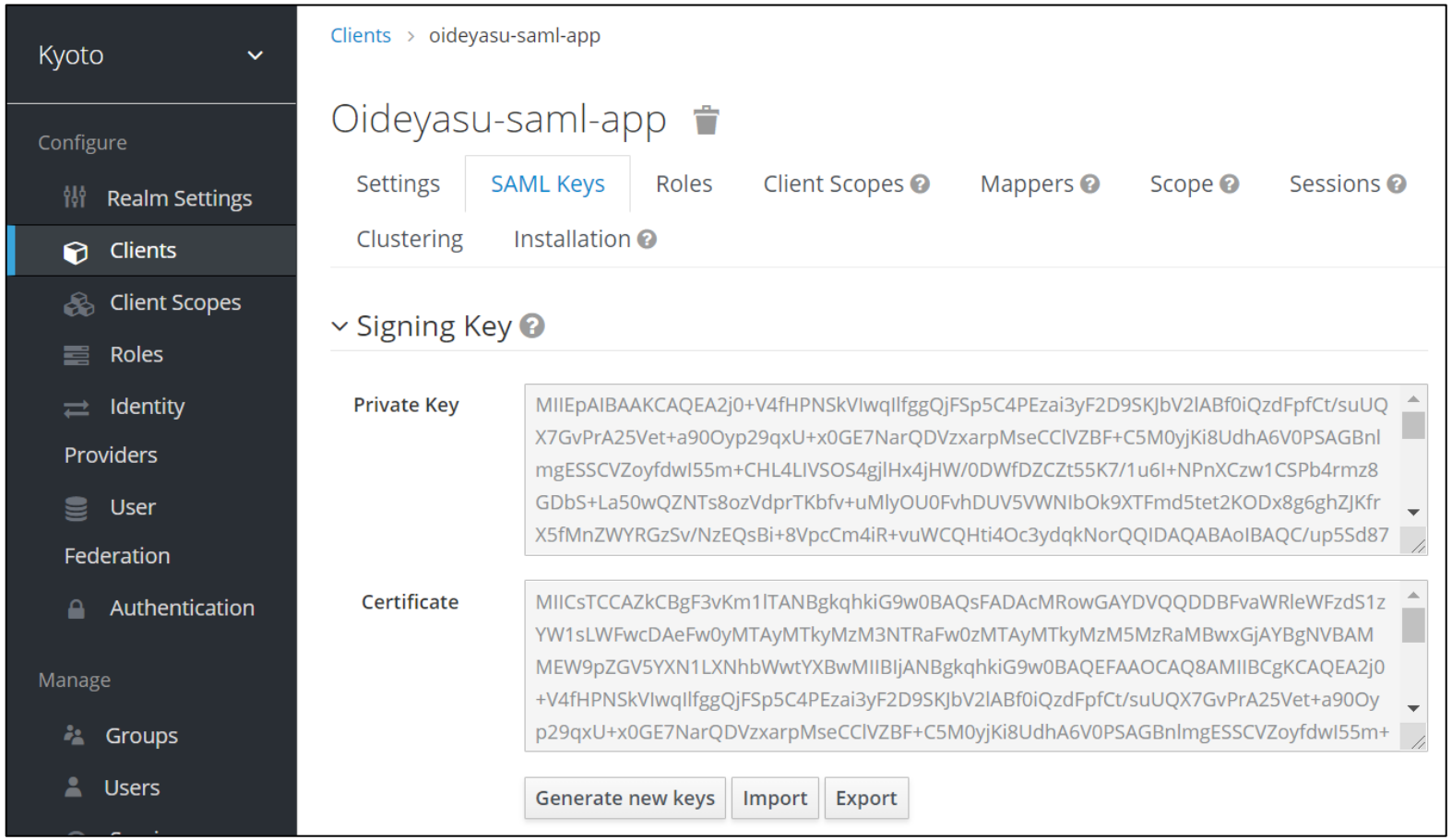
[UI Change on the existing Clients->Credentials tab]



The tooltip shows descriptions to guide the admin to access to Clients->Keys tab to set up key loading settings.

Goals : UI

[UI Change on the existing SAML Clients->SAML Keys tab]



Rename “SAML Keys” to “Keys” to align with OIDC’s client.

Working Tasks Broken Down

Issue #01 and #02

- By Value Statically

  - Import the client app’s certificate or public key in advance.

    The current keycloak implementation hardcoded RSA as key type, “sig” as key use, and RS256 as key algorithm. (ClientPublicyKeyLoader.getSignatureValidationKey)

    -> From imported public key data itself, those should be determined.

  - Generate and register the client app’s certificate in advance.

    The current keycloak only generate the certificate whose specification are RSA as key type, “sig” as key use, and RS256 as key algorithm.

    -> Nothing can be done.

  - Register the client app’s public keys by “jwks” OAuth2 Client Metadata

    Relating to [#04], key type, key use and key algorithm should be determined.

Issue #03

- Implement “jwks” client metadata on Client
- Create/read/update/delete keys via Dynamic Client Registration

  use “jwks” client metadata
- Create/read/update/delete keys via Admin REST API

  use client attribute key
- Persist received keys

  use client attribute
- Load/parse persisted received keys

  parse JWKS and determine key type, key use and key algorithm

Issue #04

- Newly implement UI

  Clients->Keys Tab

- Modify existing UI

  OIDC : Clients->Credentials Tab

  SAML : Clients->SAML Keys

Reference

Reference

KEYCLOAK-10462 Improve support for setting keys for OIDC

https://issues.redhat.com/browse/KEYCLOAK-10462

KEYCLOAK-11251 ES256 or PS256 support for Client Authentication by Signed JWT

https://issues.redhat.com/browse/KEYCLOAK-11251

KEYCLOAK-16702 Keycloak does not respect the algorithm defined in uploaded JWKS for signature verification

https://issues.redhat.com/browse/KEYCLOAK-10462

END