

FAPI-SIG Annex

OIDC Client's Public Keys Management Support

Table of Contents

Background

Issues and Goals

Working Tasks Broken Down

Reference

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	
Key Use (JWK “use”)		Key Loading Method		
		By Reference	By Value	
Key Use	“sig”	X	X	
	“enc”	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 Type (JWK "kty")		Key Loading Method	
		By Reference	By Value
Key Type	EC	X	
	RSA	X	X

Background

[client public key loading settings]

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

The screenshot shows the Keycloak Admin Console interface. On the left is a dark sidebar with navigation links: Master, Configure, Realm Settings, Clients (selected), Client Scopes, Roles, Identity, Providers, User, Federation, Authentication, Manage, and Groups. The main content area is titled 'Clients > account' and 'Account'. It has several tabs: Settings, Credentials (active), Roles, Client Scopes, Mappers, Scope, and Revocation. Below these are sub-tabs: Offline Access, Clustering, and Installation. The 'Credentials' tab shows a 'Client Authenticator' dropdown set to 'Signed Jwt'. Below it is a 'Signature Algorithm' dropdown. A 'Use JWKS URL' toggle is currently 'OFF'. A message states 'No client certificate configured'. At the bottom of this section are buttons: 'Generate new keys and certificate', 'Import Certificate', 'Save', and 'Cancel'. At the very bottom, there is a 'Registration access token' field and a 'Regenerate registration access token' button.

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

[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.

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 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.

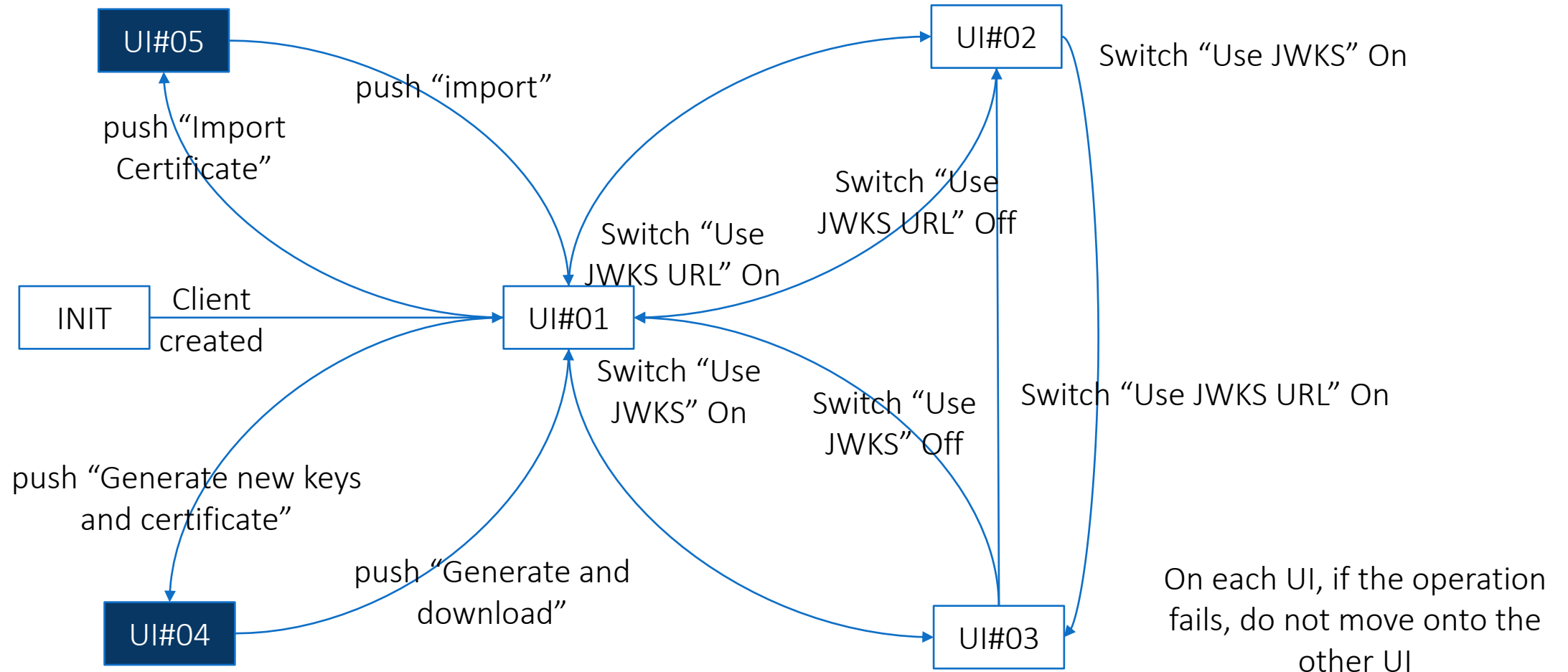
Key loading setting is tightly coupled with the one for “Signed Jwt” Client Authentication.

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]



Goals : UI

[UI#01]

Keys

Use JWKS URL ? ☐ OFF

Use JWKS ? ☐ OFF

Certificate ?

Generate new keys and certificate

Import Certificate

save

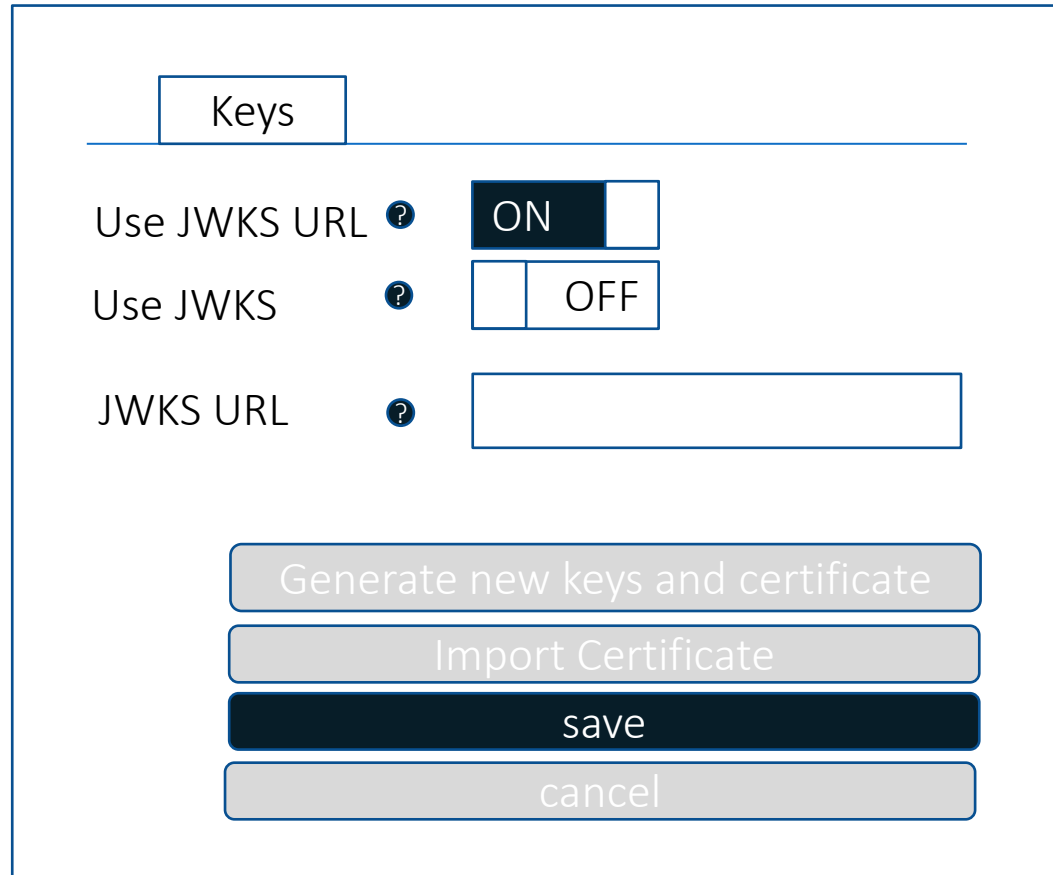
cancel

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]



The UI mockup is titled 'Keys' and contains the following elements:

- Use JWKS URL**: A toggle switch currently set to 'ON'.
- Use JWKS**: A toggle switch currently set to 'OFF'.
- JWKS URL**: An empty text input field.
- Buttons**: Four buttons stacked vertically: 'Generate new keys and certificate', 'Import Certificate', 'save' (highlighted in dark blue), and 'cancel'.

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]

Keys

Use JWKS URL ? ☐ OFF

Use JWKS ? ☒ ON

JWKS ?

Generate new keys and certificate

Import Certificate

save

cancel

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 screenshot shows the 'Generate Private Key' interface within the Kyto console. The left sidebar contains a navigation menu with the following items: 'Kyoto' (selected), 'Configure', 'Realm Settings', 'Clients' (highlighted), 'Client Scopes', 'Roles', 'Identity Providers', 'User Federation', and 'Authentication'. The main content area has a breadcrumb trail: 'Clients > oideyasu-app > Credentials > Generate Client Private Key'. The title 'Generate Private Key' is displayed. Below the title, there are four form fields: 'Archive Format' (a dropdown menu set to 'JKS'), 'Key Alias' (a text input field containing 'oideyasu-app'), 'Key Password' (a text input field with a toggle icon), and 'Store Password' (a text input field with a toggle icon). At the bottom of the form are two buttons: 'Generate and Download' (in blue) and 'Cancel' (in grey).

The same as the current Clients->Credentials Tab

Goals : UI

[UI#05]

The screenshot shows the 'Import Client Certificate' page in the Kyoto application. The left sidebar contains a navigation menu with the following items: 'Configure', 'Realm Settings', 'Clients' (highlighted), 'Client Scopes', 'Roles', 'Identity Providers', 'User Federation', and 'Authentication'. The main content area has a breadcrumb trail: 'Clients > oideyasu-app > Credentials > Client Certificate Import'. Below the breadcrumb, the title 'Import Client Certificate' is displayed. The form includes four fields: 'Archive Format' (a dropdown menu set to 'JKS'), 'Key Alias' (a text input field), 'Store Password' (a text input field with a toggle icon on the right), and 'Import File' (a button labeled 'Select file' with a file icon). At the bottom of the form are two buttons: 'Import' (in blue) and 'Cancel' (in grey).

The same as the current Clients->Credentials Tab

Goals : UI

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

The mockup shows a 'Credentials' tab with three input fields. Each field has a tooltip icon (a question mark in a circle) and a validation icon (a 'V' in a circle). The fields are: 'Client Authenticator', 'Signature Algorithm', and 'Registration access token'. Below the 'Registration access token' field is a button labeled 'Regenerate registration access token'.

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]

The screenshot displays the Keycloak administration console. On the left is a dark sidebar with a 'Kyoto' dropdown and a 'Configure' section containing 'Realm Settings', 'Clients' (highlighted), 'Client Scopes', 'Roles', 'Identity', 'Providers', 'User', 'Federation', and 'Authentication'. Below this is a 'Manage' section with 'Groups' and 'Users'. The main panel shows the configuration for 'Oideyasu-saml-app' under the 'Clients' tab. The 'SAML Keys' sub-tab is active, showing a 'Signing Key' section. It contains two text areas: 'Private Key' and 'Certificate', both filled with long, complex base64-encoded strings. At the bottom of the section are three buttons: 'Generate new keys', 'Import', and 'Export'.

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.
(ClientPublicKeyLoader.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.

- 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