BIZA·IO

Response to Decision Proposal 209: Transition to FAPI 1.0 Advanced Profile

Introduction

Decision Proposal 209 focuses on the proposal to transition the existing CDR Information Security Profile (a modified FAPI ID2 derivation using a draft version of PAR) to be aligned with the FAPI 1.0 Advanced Final profile combined with RFC9126 (the final version of PAR).

As has been our position since the beginning of the CDR implementation Biza.io is strongly in favour of alignment to international standards. As a member of the OpenID Foundation and a contributing member of the FAPI Working Group we support the unmodified adoption of FAPI 1.0 Advanced Final.

Within this feedback we provide:

1. Comments related to the accuracy of the proposal
2. Feedback with regards to phasing schedules
3. Readiness with respect to installations we deliver on behalf of Holders

References

The following documents were used and are referenced during preparation of this analysis:

Document Name	Date (Version)	URL
Financial-grade API Security Profile 1.0 – Part 1: Baseline	March 12, 2021 (Final)	https://openid.net/specs/openid-financial-api-part-1-1_0-final.html
Financial-grade API Security Profile 1.0 – Part 2: Advanced	March 12, 2021 (Final)	https://openid.net/specs/openid-financial-api-part-2-1_0.html
Financial-grade API: JWT Secured Authorization Response Mode for OAuth 2.0 (JARM)	October 17 2018 (Draft-02)	https://openid.net/specs/openid-financial-api-jarm.html
OAuth 2.0 Pushed Authorization Requests	September 2021 (RFC9126)	https://www.rfc-editor.org/rfc/rfc9126.txt
Proof Key for Code Exchange by OAuth Public Clients	September 2015 (RFC7636)	https://datatracker.ietf.org/doc/html/rfc7636
Proposal - should we remove support for refresh token rotation from FAPI 2.0	November 11 2021 (Latest Comment)	https://bitbucket.org/openid/fapi/issues/456/proposal-should-we-remove-support-for

FAPI 2.0 Note

This document refers to “FAPI 2.0” in the broad context of the current approach being taken by the FAPI Working Group whereby it is more broadly referred to as the FAPI 2 Framework[1].

[1] https://bitbucket.org/openid/fapi/issues/432/fapi2-trust-framework-structure

Decision Proposal Clarifications

Biza.io wishes to note the following clarifications regarding the Decision Proposal.

PKCE regardless of Response Type

The phasing table (Page 12) contained within the Decision Proposal contains a reference regarding ***`response_type`*** stating *"code id_token (unless supporting PKCE)"*.

We note that according to FAPI 1.0, PKCE is *mandatory* for all PAR requests regardless of the ***`response_type`*** in use. As the CDR requires PAR support already, alignment would implicitly rely upon PKCE being supported at this endpoint to facilitate FAPI 1.0 Final alignment.

JARM Support

The phasing table (Page 11) specifies that JARM Support is not required *however:*

FAPI 1.0 states the following in *5.2.2 (2)*:

```
shall require
   1) the response_type value code id_token, or
   2) the response_type value code in conjunction with the
      response_mode value jwt;
```

While *5.2.2.2* states the following:

```
In addition, if the response_type value code is used in conjunction
with the response_mode value jwt, the authorization server
   1) shall create JWT-secured authorization responses as specified in
      JARM, Section 4.3.
```

This has the effect of indicating that response type must be either:

1) `code id_token` (Hybrid)
2) `code` with JARM based `response_mode` of ***`jwt`***

In essence this means that should the DSB incorporate `code` flow that JARM support is *required*.

Additional Considerations

In addition to the considerations outlined in the Decision Proposal Biza.io raises the following items for consideration.

Introduction of PKCE for PAR

As noted above PKCE is required for PAR requests to reach alignment with FAPI 1.0. The Decision Proposal suggests a ***response_type*** of ***code id_token*** transitioning to *code* which appears to tie PKCE adoption to this migration.

PKCE is *separate* from potential changes to ***response_type***. Instead, PKCE allows the Data Recipient (Client) to generate a ***code_verifier*** which is then transformed (using the ***code_challenge_method)*** into a ***code_challenge*** during the submission of the request (to the PAR endpoint).

After the authorisation flow has completed and during the subsequent submission to the token endpoint the Data Recipient then submits the previously undisclosed ***code_verifier*** and the Data Holder verifies this aligns with the ***code_challenge*** sent at the beginning of the request.

By implementing PKCE the Holder is able to verify that the Recipient has not had its code intercepted and used to illegitimately claim tokens – regardless of inadvertent exposure of PAR request submission or authorisation response.

JARM Support for Code Flow

As noted above FAPI 1.0 essentially makes JARM mandatory for ***response_type*** of *code*. Consequently, for FAPI 1.0 alignment, it would be mandatory for *code* flow adoption.

Biza.io notes that the **code** flow within the FAPI 2.0 Baseline Profile *does not* have a mandatory requirement for JARM based response mode. The *primary* reason for this is that the FAPI 2.0 Baseline is not intended to provide message integrity, which JARM provides to the authorisation code. In contrast the FAPI 2.0 Advanced Profile *does* intend to provide message integrity and this is in the form of JARM for ***code*** response type and signed introspection responses.

While signalling such features early would be beneficial it is *critical* that the DSB decides whether message integrity (and more broadly non-repudiation) is a continuing requirement for the CDR. For its part Biza.io recommends that message integrity *should* be considered a requirement, as is already the case with the use of **code id_token** hybrid flow, and retained in future iterations of the CDR Information Security Profile.

Dual Response Type Support

The Decision Proposal currently proposes to migrate from ***code id_token*** response type to ***code*** only response type. As noted above this would implicitly require JARM support to reach alignment.

While migrating away from ***code id_token*** sets the CDR Standards towards a path of FAPI 2.0 adoption we believe it may be premature at this stage. Instead, Biza.io suggests that the DSB consider whether dual `response_type` support may be worthwhile and have outlined such an approach in our revised phasing table.

Removal of Encrypted ID Tokens

Biza.io has always questioned the usefulness of encrypted ID Tokens particularly because the CDR Standards block the return of PI claims on front-channel (from authorisation endpoint) and the back-channel (token endpoint) is protected by MTLS.

In the live ecosystem Biza.io has experienced race conditions, during time sensitive operations (ie. time constrained authorisation codes), caused by non-performant JWKS endpoints (including those of the ACCC CTS) delaying the efficient retrieval of keys to be used for encrypting ID Tokens

While not explicitly required to achieve FAPI 1.0 alignment we note that this may be an opportunity to remove encrypted id tokens being required, aligning with essentially all other international ecosystems and increasing vendor choice for holders as the CDR enters additional sectors.

Signed Introspection Support

In addition to JARM we note that the other non-breaking change future alignment with FAPI 2.0 is the support for signed introspection responses. As JARM is now within scope as part of ***code*** we feel it is only natural to raise support for signed introspection response support as well.

Disable Refresh Token Cycling

Refresh Token Cycling within confidential clients being issued sender-constrained tokens is recognised as posing no significant security benefit *("security theatre")* while conversely creating numerous ecosystem interoperability issues during error conditions.

The FAPI Working Group currently has an issue open[2] to explicitly remove support for refresh token rotation from FAPI 2.0 security profiles. Within the CDR ecosystem disabling cycling would enable a simplification of the current method of communicating sharing expiration as the sharing duration could be tied to the refresh token expiration.

[2] https://bitbucket.org/openid/fapi/issues/456/proposal-should-we-remove-support-for

Impact Analysis

Biza.io thanks the DSB for providing a list of specific changes to achieve FAPI 1.0 alignment. In order to assist in the decisioning related to how quickly to migrate we provide the following high level impact analysis of each change.

Item	Biza.io Customers	Ecosystem Holder	Ecosystem Recipient
Request URI Replay (PAR)	None. Request URI Replay is already enforced.	Low. Most Holders already enforcing.	None. No recipient is reusing request_uris.
Require PAR	None. We prefer PAR only.	None. Holders already required to accept PAR so making it exclusive is unlikely to be an issue.	Low. Biza.io believes all Active Recipients already utilise PAR exclusively.
PKCE Support	None. Biza.io Platform already supports PKCE.	Medium. PKCE support depends on Vendor support.	Medium. PKCE support is untested by Recipients however if they don't support it their existing implementation will continue functioning until it is Mandatory.
Authorisation Code Reuse	None. Biza.io Platform already blocks authorisation code reuse.	Medium. A reasonable proportion of Holders have challenges enforcing code reuse protections (especially instant code reuse versus 30 second delay use)	None. No recipient is reusing authorisation codes.
Scope Request Support	None. Biza.io Platform will continue to provide scopes regardless.	None. Existing implementations can remain unchanged.	Low. Introducing conditional check of scope presence is likely to be a low impact change to recipients.

Multi-Brand Support	None. Biza.io Platform already uses separate issuers per brand.	Medium. Biza.io is aware of some installations which are recycling key material across brands and/or using duplicate issuers.	Low. Recipients are unlikely to be impacted by this alignment as they (should) rely upon discovery documents anyway.
x-fapi-customer-ip-address support IPv4 and IPv6	None. Biza.io Platform accepts these headers already.	None. Biza.io is not aware of holders rejecting these requests.	Low. Recipients are supplying well-formed IP addresses already.
Request Object Expiry	Low. Biza.io Platform already enforces request object expirations in alignment with FAPI	Low. Biza.io believes most holders are already enforcing expiry.	Low. Limited reasons why a Recipient would require a request to not expire for extended periods.
Content-Type Header Requirement	None. Biza.io Platform accepts both formats.	Low. We believe there a small number of holders using string-based header parsing and rejecting these requests.	None. Existing Recipients would be issuing the specified Content-Type already to ensure functionality. Existing implementations would be unaffected by this change.
Cipher Support	None. Biza.io is comfortable with the existing ciphers.	None. Existing Holders already comply with existing ciphers.	Low. Recipients would need to support new ciphers however this is likely to already be the case.
Ignore Claims outside the Request Object	None. Biza.io already ignores such claims.	Low. Most existing holders already ignore claims.	None. Biza.io believes Recipients are already in compliance.
(Additional) Remove Encrypted ID Tokens	None. Biza.io can alter its discovery document and	None.	Low.

	await DCR updates from Recipients.	Holders can alter their discovery document and await DCR updates from Recipients.	Recipients can perform a DCR Update to disable ID Token encryption.
(Additional) Require PKCE	None. Biza.io Platform already supports PKCE.	Medium. PKCE support depends on Vendor support.	High. PKCE support is untested by Recipients and one holder mandating it will require all recipients to uplift.
(Additional) Support `code` flow with `response_mode` of *`jwt`* using JARM.	Medium. Biza.io Platform does not currently support JARM but would do so if it was supported in the CDR ecosystem.	Medium. JARM is not supported by many vendors.	Medium. Recipients can continue using hybrid flow however introducing JARM is likely a body of work should they choose to do so.
(Additional) Support signed introspection responses.	None. Biza.io Platform already supports signed introspection responses.	Medium. Signed introspection responses have varying vendor support	None. Existing implementations would continue to receive unsigned introspection responses.
(Additional) Disable Refresh Token Cycling	None. Biza.io Platform does not currently utilise refresh token cycling.	Low. Primarily driven by numerous issues within the live ecosystem Refresh Token Cycling has been disabled in a majority of Holder deployments.	None. Existing implementations would remain unchanged (they already handle both scenarios).

Question Responses

We provide the following feedback regarding the specific questions posed by the DSB.

1(a) Should Refresh token expiry time be pegged to consent duration?

Biza.io sees our answer to this question being somewhat tied to the future of Refresh Token Cycling. We see essentially nil value in rotating Refresh Token's within the CDR security context and as such believe refresh token expiration before consent expiration should not occur – therefore expiration time should, at a minimum, be the consent duration expiration time.

On the other hand, issuing refresh tokens with expiration *more than* the sharing duration *may* have value in the future, for instance to find consent status after it has expired, however we feel this is a use case best considered in the broader context of rich authorisation.

As it stands right now Holders are either issuing refresh tokens shorter than sharing duration (and cycling) or issuing refresh tokens with an expiration matching the consent duration. In summary, Biza.io supports the pegging of Refresh Token Expiry time to the consent duration.

1(b) Should CDR authorisation input parameters be registered or otherwise moved out of the authorisation request object?

We find it unlikely that ***`cdr_arrangement_id`*** and ***`sharing_duration`*** would pass the requirements for international registration and that doing so would likely be a time consuming exercise with possible zero success. Additionally, moving these parameters to another place *now* would appear to be a case of *too little too late* as implementations have already adapted to support these requirements.

Biza.io agrees with the alternative which is to leave the input parameters as-is and instead focus on the next steps associated within Decision Proposal 210 and the adoption of Rich Authorization Requests.

1(c) Should CDR token response parameters be registered or otherwise moved out of the parent token endpoint response JSON / ID token JWT?

As per our answer to 1(b) we find it unlikely any of the response parameters would pass the requirements for international registration.

With this stated we *do* see opportunity to simplify the claims within the ID Token and as such suggest, assuming that refresh token expiration time is pegged to sharing duration, that both the ***`sharing_expires_at`*** and ***`refresh_token_expires_at`*** be removed entirely as they would both be the same values as that described within the *exp* of the refresh token either as a JWT or via an introspection request. For once-off consents the absence of a refresh token would indicate a single access token or an ***`exp`*** within 24 hours of the ***`iat`*** would be sufficient.

Fundamentally our strategy in this regard is focused on streamlining the existing process and removing potential future blockers toward FAPI 2.0 and RAR adoption. With the above proposal *only* the ***`cdr_arrangement_id`*** would be present - within the token endpoint response – smoothing the pathway for eventual removal of ID Token entirely.

2(a) Should the CDR explicitly define the `request_uri` must only be used once and cannot be replayed

Biza.io supports the statement that a `request_uri` must only be used once and cannot be replayed.

While Biza.io currently explicitly enforces this we note there is the potential for user agents to trigger reuse (refreshing browser, double clicking a generated link etc) and as such recommend the DSB consider defining error behaviour should a `request_uri` be detected as utilised more than once and define recommendations for Data Recipients with respect to avoiding this behaviour.

2(b) Should the CDS explicitly define the upper lifetime of the PAR `request_uri`?

Biza.io supports defining the upper lifetime of a PAR generated `request_uri` however we believe 60 minutes would be far more than the recommendation within RFC9126 2.2 which states:

```
The request URI lifetime is at the discretion of the authorization server
but will typically be relatively short (e.g., between 5 and 600 seconds).
```

We suspect the DSB has given 60 minutes as the example based on the FAPI specification but this is related to the Request Object submitted not the PAR `request_uri` generated and realistically user agents are typically immediately redirected.

At this point Biza.io utilises PAR requests with a 90 second lifespan.

2(c) Should the Data Standards make requiring PAR mandatory for all Data Holders and Data Recipients?

Biza.io strongly supports migrating exclusively to PAR as the request object submission method as it significantly reduces the front-channel attack surface, simplifies implementations and allows broader approaches to solutions for scaling.

3(a) Should JARM be supported when `response_type` is code?

As stated in Decision Proposal Clarifications we believe JARM is *required* when `response_type` is code because the FAPI specification states `response_type` of `code` requires `response_mode` of `jwt` for which JARM is specified as the method for

doing so.

3(b) Should the Data Standards require JARM when response_type is code?

We believe this is the case for FAPI 1.0 alignment.

3(c) Should the CDS mandate that the same kid is not allowed to be used by multiple keys within a JWKS?

FAPI 1.0 recommends this behaviour but does not mandate it. We note that, due to encrypted id tokens, a duplicate `kid` within the CDR has a higher probability of causing significant issues.

Biza.io supports the explicit requirement to ensure all JWKS kid values are unique.

Alternate Phasing Schedule

Biza.io believes that the migration to FAPI 1.0 is an opportunity to realign the CDR with international standards and in so doing benefit from the experience of multiple ecosystems globally. We also believe that where there is opportunity for implementers to start aligning with the emerging FAPI 2.0 profiles that they be enabled to do so.

Biza.io feels that acceleration of this re-alignment wherever possible would be advantageous to the ecosystem so that more time can be spent on developments for which Australia can lead on (for instance complex consents enabled by FAPI 2.0 Trust Framework) rather than lagging international alignment. By doing so the CDR ecosystem can also benefit from early adopters on both sides who can provide feedback in future proposals regarding learned experiences.

We recommend an acceleration towards FAPI 1.0 alignment and instead propose the following phasing table with the first occurrence of alignment coloured in green:

	Current State	Phase 1	Phase 2
		1st March 2022	1st May 2022
FAPI 1.0 Baseline (Final) Support	Implementer's Draft 2 (Draft 06)	Partial	Fully Supported
Scope Request Support	Always	FAPI 1.0	FAPI 1.0
Authorization Code Reuse	SHOULD refuse	MUST refuse	MUST refuse
Content-type Header Requirement	SHOULD support	SHOULD support	MUST support
FAPI 1.0: Advanced (Final) support	Implementer's Draft 2 (Draft 06)	Partial	Fully Supported
Cipher Support	Draft 06	Draft 06	FAPI 1.0
JARM Support & JWT Response Mode	No	MUST with `code` response type and `jwt` response mode	MUST with `code` response type and `jwt` response mode

PAR version	Draft 01	RFC 9126	RFC 9126
Require Pushed Authorization Requests	Not Supported	Optional	Mandatory
Request Object Submission	Authorisation endpoint and PAR	Authorisation endpoint and PAR	PAR only
PKCE Support (RFC 7636)	Not Specified	Optional	Mandatory
Response Type	`code id_token` (MUST)	`code id_token` (MUST) `code` (MAY)	`code id_token` (MUST) `code` (MAY)
Request URI Replay	SHOULD refuse	MUST refuse (Not Allowed)	MUST refuse (Not Allowed)
Multi-Brand Support (Separate Issuers for Data Holder Brands)	Separate issuer	Separate issuer	Separate issuer
Access Token Revocation	Mandatory	Mandatory	Mandatory
Additional Proposals			
Signed Introspection Request/Responses	No	Optional	Optional
Unencrypted ID Token Support	No	Optional	Mandatory
Disable Refresh Token Cycling	No	SHOULD NOT	SHALL NOT

About Biza.io

Biza.io are the market leaders in Data Holder solutions to the Consumer Data Right and are the only *pure-play* CDR vendor in Australia. Founded by the former Engineering Lead of the Data Standards Body (DSB), Biza.io has been involved in the Data Standards setting process since the very beginning and its personnel remain the largest non-government contributors to the consultations. In addition to its participation within the CDR, Biza.io is also a contributing member of the Financial-grade API (FAPI) Working Group, contributors to the FAPI 1.0 information security profile and co-authors of the Grant Management for OAuth 2.0 specification.

About Our Customers

As of November 2021, Biza.io is directly responsible for delivering, or verifying solutions used by, ***over 60%*** of active Data Holders servicing more than ***4 million Australians***. Beyond just a contractual engagement Biza.io considers all its customers partners in the journey toward open data. Our customers choose us to not only achieve compliance but to compete then command the consumer data ecosystem.