What does Presentation Exchange do and what parts of it do we actually need?

Mike Jones

OSW August 2023 London – Unconference Discussion

Background

- Presentation Exchange (PE) is a Swiss Army Knife of a specification
  - It can do many things supporting many different use cases
- OpenID for Verifiable Presentations uses it
  - https://openid.bitbucket.io/connect/openid-4-verifiable-presentations-1_0.html
- High Assurance Interoperability Profile requires a subset of PE
  - https://vcstuff.github.io/oid4vc-haip-sd-jwt-vc/draft-oid4vc-haip-sd-jwt-vc.html

- See the Presentation Exchange spec at
  - https://identity.foundation/presentation-exchange/spec/v2.0.0/

Let’s create a feature inventory together

Presentation Exchange Feature Description	Feature Name	HAIP
Request claims in presentation	constraints/fields/path	yes
Request claims within nested structures	path w/ JSONPath	yes
Request minimal disclosure	limit_disclosure	yes
Request presentation format	format	yes
Request proof type	format/proof_type	no
Request multiple presentations	input_descriptors array	yes
Specify multiple choices from groups (3 of A and 2 of B or 1 of C)	submission_requirements	parts
Regular expressions	filter/pattern	no
Abstract query syntax across credential formats	(PE design assumption)	implicitly
Response contains inventory of multiple presentations	presentation_submission/ descriptor_map	yes

Notes from Wednesday Unconference Session

Brian: What use is the presentation_submission? You have to validate the received credentials anyway.

Someone observed that the descriptor_map can let you inventory what you received without looking inside the credentials.

Brian worried that developers could accept the map at face value and not validate the credentials.

Mike: Failures become opaque if the application just receives a "failed" error without saying what is missing

Couldn't we just ask for one credential at a time?

Kristina: We can request multiple presentations without using presentation_requirements just using an array.

Justin: Aren't you creating privacy-destroying oracle if you return detailed error information?

Isn't the wallet in a better position to give actionable error information than the application?

We can't assume good intentions in a query language.

Mike: We'll have a follow-up session tomorrow.

Which functionality do we need & why?

Presentation Exchange Feature Description	Feature Name	Functionality Needed & Why?
Request claims in presentation	constraints/fields/path	Yes – data minimization for verifier & privacy for end-user
Request claims within nested structures	path w/ JSONPath	Yes, for credentialSubject claims
Request minimal disclosure	limit_disclosure	Selective disclosure yes
Request presentation format	format	Requesting credentials in formats useful
Request proof type	format/proof_type	Negotiating algorithms needed – could be done with metadata
Request multiple presentations	input_descriptors array	Combined requests can have better UX or it can be worse
Specify multiple choices from groups (3 of A and 2 of B or 1 of C)	submission_requirements	Some use cases described
Regular expressions	filter/pattern	No: Needs to be blocked for security
Abstract query syntax across credential formats	(PE design assumption)	Alternative is format-specific query langs
Response contains inventory of multiple presentations	presentation_submission/ descriptor_map	(Brian questioned this on Wednesday)

Notes from Thursday Unconference Session (1 of 2)

Daniel: JSONPath should instead be JSON Pointer
Daniel: Can PE ask for everything in a structure or do you have to always request individual claims?
    How does that interact with limit_disclosure?
Yaron: Thinks it's useful to have the query language be format agnostic
    There will be multiple competing credential formats
    Mike: Credential formats are not self-describing
Brian: You have to understand a credential format to look inside of it
Daniel: You believe the issuer's statements about the credential format
Kristina: proof_type only makes sense for Linked Data Proofs
    Algorithms are separate
Mike: In OpenID and OAuth, we tend to do algorithm negotiation in metadata
    *_signing_algs_supported, etc.
Brian: The issuer's algorithm and the presentation algorithm are different
Brian: Supported credential formats could be in metadata, rather than the query language
Kristina: You could also pre-agree to supported features and simply pass short feature set handles
Brian: Is there a way to indicate which issuers you're willing to accept credentials from?
    Kristina: Yes - constraints on issuer values, for example - not just claims
Kristina: How you express issuer depends upon the credential format
    mDOCs express the issuer with the issuer certificate
Kristina: Being able to specify the issuer important

Notes from Thursday Unconference Session (2 of 2)

Mike: Do we need to be able to ask for multiple things at once or can we ask for things in different requests?
Kristina: Multiple consents or QR code scans result in a really bad user experience
Brian: Combined requests can have better UX or can have worse UX - especially in error cases
Kristina: We could imagine requesting multiple credentials and returning them one-by-one
    That could make the error conditions clearer
Kristina: Selecting one of mDL or another format useful
Kristina: Combinations of credential type and formats could be useful
Mark: Achieving a level assurance could require one strong or two medium
Mark: Transformed claims is a missing feature
    Brian: You can’t transform signed data
Kristina: Regular expressions need to be blocked for security reasons
Mike: We may need format-specific query languages under some circumstances
    For instance, for binary formats with binary claim names
    Some format-specific query languages already exist