OpenID-SSI_Login Variants

I see three ways we could go about this but there could be more:

1. 1-Step Proof-Exchange
2. 2-Step Did-Auth + Proof-Exchange
3. 3-Step Password + Did-Auth + Proof-Exchange
4. DID-Auth vs no DID-Auth

> Note: DID-Auth in this case describes the process of proving ownership/control of a public or pairwise DID.

General

- user initiates openid-connect at client website
- user is sent to oidc-provider
- if previous login exists: user may choose account or new login
- oidc consent form is provided at the end

1-Step Proof-Exchange

Flow: OIDC Init -> Proof-Exchange -> Back to OIDC

- oidc-provider either / and
  - for app: provides proof-request as QR-Code or through app deep-link
  - for web-based agent: provides list of supported agents or allows copy-pasting proof-request as JSON
- proof request contains `~service` information containing response return address
- requested attributes in proof-request may change with requested claims / scopes for openid-connect
- example proof request from https://github.com/bcgov/vc-authn-oidc/blob/master/docs/README.md (https://github.com/bcgov/vc-authn-oidc/blob/master/docs/README.md)

```
{
    "@type": "did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/present-proof/1.0/request-presentation",
    "@id": "<uuid-request>",
    "comment": "some comment",
    "~service": {
        "recipientKeys": ["8HH5gYEeNc3z7PYXmd54d4x6qAfCNrqQqEB3nS7Zfu7K"],
        "routingKeys": ["8HH5gYEeNc3z7PYXmd54d4x6qAfCNrqQqEB3nS7Zfu7K"]
        "serviceEndpoint": "https://example.com/endpoint"
    },
    "request_presentations~attach": [
        {
            "@id": "request-presentation-0",
            "mime-type": "application/json",
            "data":  {
                "base64": "<bytes for base64>"
            }
        }
    ]
}
```

- very similar to https://github.com/bcgov/vc-authn-oidc (https://github.com/bcgov/vc-authn-oidc)
- very similar to first poc (https://git.snet.tu-berlin.de/blockchain/dims/openid-ssi_login/tree/poc), also see `docs` there
- this is more vc-auth than did-auth

requires:

- support of new message formats and decorators / fields
- mobile app: scanning and processing proof-request qr-code, app deep-link
- backend api: processing proof-request through custom frontend or `POST` proof-request

does not require:

- did-exchange / connection
- did-auth

2-Step Did-Auth + Proof-Exchange

Flow: OIDC Init -> DID-Exchange -> DID-Auth -> Proof-Exchange -> Back to OIDC

- oidc-provider either / and
  - for app: provides did-auth request as QR-Code or per app deep-link
  - for web-based agent: provides list of supported agents or allows copy-pasting did-auth request as JSON
  - all: provides way to initiate did-exchange first if did is not public
- did-auth:
  - request not yet specified in aries-rfcs, examples can be found here https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/did-auth.md (https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/did-auth.md)
  - request may include `~service` response address information to enable encryption
  - response should provide did whose ownership to prove and signature to verify
  - did-auth would replace username-password authentication
- proof-exchange:
  - if new attributes are required
  - request is sent directly to endpoint either
  - request is anoncrypted if did-auth was with public-did and no pairwise connection exists, this requires a valid did-doc on the ledger, or authcrypted if pairwise connection exists
- oidc-provider would keep track of some sort of `account`, which also holds previous proofs and consents

requires:

- support of custom did-auth-exchange protocol
- support of `~service` field
- did-exchange/connection or public did

possibility for simplification:

- make did-exchange mandatory
- remove support for public-did with requirement for did-doc on ledger
- remove support for `~service` field in did-auth request as it is no longer required
- also simplifies proof-exchange

3-Step Password + Did-Auth + Proof-Exchange

Flow: OIDC Init -> Username+Password -> DID-Exchange -> DID-Auth -> Proof-Exchange -> Back to OIDC

- oidc-provider shows login prompt with username and password and possibility to register
- after successful login: oidc-provider shows did-auth request and/or connectionOffer
- after successful did-auth: proof-exchange if new attributes are required
- did-auth is essentially 2nd factor in authentication
- proof-exchange provides attributes for oidc-scopes and claims

requires:

- username and password
- did-exchange/connection or public did

becomes simpler if public-did support is removed, just like 2-Step version

DID-Auth vs no DID-Auth

did-auth:

- did-auth may replace traditional username / password schemes
- allows for `account` like storing of information and reuse even if session expires or on logout/login
- `proof-exchange` only to refresh information (may be done periodically) or when additional attributes are required by oidc scopes/claims

no did-auth:

- allows for having no account at oidc-provider and reduce tracking, data can be deleted on logout or session expiration
- stored information can only be reused as long as session exists
- requires `proof-exchange` on session expiration or logout/login