Format-specific identity credentials requests

Introduction

Various identity credential formats exist today, each with its own encoding and data structure, like ISO/IEC 18013-5:2021 mdoc, IETF SD-JWT VC, and W3C VCDM. We expect more formats to emerge in the future, such as the CBOR-based identity credential format being explored by the upcoming IETF SPICE working group.

Attempts to create a universal query syntax, like Presentation Exchange, faced challenges due to the excessive optionality and its core features relying on JSON Path, which is not practical for 1) supporting CBOR-encoded identity credentials utilizing non-string claim identifiers and 2) creates a barrier for adoption for relying parties due to the significant complexity of JSON Path which must first be understood before being able to use presentation exchange.

This document introduces a request structure and query syntax with shared features across different identity credential formats, allowing for format-specific query components.

One of the core design principles of this approach is that the different identity credential formats available in the market today have entirely different underlying data models, representation technologies, securing formats and extensibility models and thus the query syntax should recognise this and provide developer friendly ways to query credentials of different formats discreetly.

Use cases

The core use case is to support protocols for the identity credential exchange between a verifier and a holder by defining a simple and developer-friendly JSON-based query syntax. The following sections describe required features for the query syntax.

Basic scenarios

The query syntax MUST allow the verifier to define which identity credentials and/or what claims are required in the response to an identity credential presentation request to satisfy the verifier.

The following scenarios are considered:

- As a verifier, I want to request the user's mobile driver's license.
- As a verifier, I want to request the birthdate AND the portrait from the user's mobile driver's license.

- As a verifier, I want to request the given name and family name from any identity credential the user has.

Scenarios with optionality

To minimize the number of unsuccessful identity credential presentation requests resulting from a mismatch between verifier requirements and available identity credentials in the wallet, the query syntax MUST allow the verifier to encode different options that would equally satisfy the identity credentials presentation request.

The following scenarios are considered:

- As a verifier, I want to request the user's driving license OR personal identification credential.
- As a verifier, I want to request the given name and family name from the user's driving license OR the firstname and lastname from the user's health card.
- As a verifier, I want to request the portrait AND birthdate from the user's mobile driving license, OR the portrait and age-over-18 attestation from the user's mobile driving license.
- As a verifier, I want to request the (portrait AND (zip code OR (city AND state)) from the user's mobile driving license OR personal identification credential.

Scenarios with criticality information

To facilitate data minimization and to reduce the risk of oversharing, the query syntax MUST be able to articulate the criticality of requested claims in identity credentials to provide the service to the user.

The following scenario is considered:

- As a verifier, I want to request a number of claims from the user's mobile driving license where all requested claims are required except the address claim is optional.

Indication of preferred options

The query syntax MUST support expressing the preferred order of provided query options, particularly in situations where the costs of the options vary in terms of UX, performance, security, monetary costs, or other implications affecting the verifier.

The following scenarios are considered:

- As a verifier, I want to request credential holder verification from the user's driving license to onboard the user, OR the portrait from the user's driving license to identify the user using facial recognition software. Since holder verification will result in a better UX and lower dropout rate, I prefer this over the portrait.

Scope

This document defines a query syntax and credential format-specific query parameters with the following requirements:

- Format-specific query syntaxes for ISO/IEC 18013-5:2021 mdoc, IETF SD-JWT VC, W3C VCDM.
- Ability to express query requirements for a presentation request that results in exactly one identity credential in the response.
- Ability to express the ordered preference of the verifier if multiple possible options are provided to satisfy the query.
- Ability to provide information about the verifier identifier and discover metadata about the verifier
- Supports different higher level protocols such as OID4VP

The following is explicitly out of scope:

- It is explicitly out of scope of this document to request multiple identity credentials in one response. To reduce complexity in the platform-level mediation process, this will be done using multiple distinct sequential queries, when needed.
- Protocol-specific holder or wallet (device) authentication.

Presentation Request

Specific identity presentation protocols MUST use a `PresentationRequest` object with the following parameters to represent the identity credential presentation request :

- `queries`: REQUIRED.

  The `queries` parameter is an array of independent format-specific `Query` objects ordered by the preference of the verifier. The `queries` parameter MAY contain multiple `Query` objects of the same format-specific profile. The wallet MUST select an identity credential that fulfills the requirements of one `Query` object contained in the presentation request. Supplying multiple `Query` objects is useful in situations where a verifier is interested in specific claims that could be included in identity credentials of different types and/or formats, and has no prior knowledge of which identity credentials are available in the user's wallet.

The following example shows an instance of a `PresentationRequest` object:

```
{
  // ordered (prioritized) list of queries
  "queries":[
    {
      // profile-specific parameters
```

```
    },
    {
      // profile-specific parameters
    }
  ]
}
```

Format-specific query profiles

The following parameters are defined for each `Query` object irrespective of the specific credential format-specific profile:

- `format`: REQUIRED. The `format` parameter contains the credential format identifier as defined in OID4VC and indicates which format-specific profile the `Query` object conforms to, e.g., `mso_mdoc` for ISO/IEC 18013-5:2021 mobile documents.

Each profile MUST define the parameters of the `Query` object used for querying identity credentials of the specific credential format.

ISO/IEC 18103-5:2021 mdoc

This section defines the credential-format specific profile for ISO/IEC 18013-5:2021 `Query` objects.

The `format` parameter MUST be set to `mdoc_mso` and the identity credential in the response MUST conform to ISO/IEC 18013-5:2021. `Query` objects conforming to this profile MUST use the following parameters:

- `docType`: OPTIONAL. If the `docType` parameter is present, the response MUST include an mdoc (as per ISO/IEC 18013-5) of that document type (as per ISO/IEC 18013-5:2021). If the `docType` parameter is omitted, the mdoc in the response MAY have any document type. If the `docType` parameter is omitted, the `nameSpaces` parameter MUST be present.
- `nameSpaces`: OPTIONAL. The `nameSpaces` parameter is a `Namespace` object describing that MUST be included in the mdoc in the response. The `Namespace` object is encoded as a map where each entry MUST use a namespace identifier (as defined in ISO/18013-5:2021) as the key for that entry. The corresponding value is be a map where each entry MUST use a data element identifier (as per ISO/IEC 18013-5:2021) as the key, and the associated value MUST be a `Constraint` object with the following entries:
    - `required`. OPTIONAL. The `required` parameter is a boolean value if set to `true`, the mdoc in the response MUST contain a data element with the data element identifier that identifies the `Constraint` object. If the `required` parameter is omitted or set to `false`, the mdoc in the response MAY have a data

element with the data element identifier of the corresponding `Constraint` object.

- `requiredIfPresent`. OPTIONAL. The `requiredIfPresent` parameter is a boolean value if set to `true`, the mdoc in the response SHOULD contain a data element with the data element identifier that identifies the `Constraint` object if the data element is included in the Mobile Security Object (MSO) (as per ISO/IEC 18013-5:2021). If the `requiredIfPresent` parameter is omitted or set to `false`, the mdoc in the response MAY have a data element with the data element identifier of the corresponding `Constraint` object.
- `intentToRetain`. OPTIONAL. The `intentToRetain` parameter is a boolean value if set to `true`, the wallet MUST indicate to the user that the verifier intends to retain the data element value (see ISO/IEC 18013-5:2021 for a definition) corresponding to the `Constraint` object. If omitted or set to `false`, the wallet MUST indicate to the user that the verifier does not intend to retain data associated with the data element corresponding to the `Constraint` object.

Note that identity credentials in the ISO/IEC 18013-5 mdoc format cannot selectively disclose fields in a data element that contains a CBOR data structure. In this case, the entire data structure is provided in the identity credential response.

The following is a non-normative example of a `Query` object that requires an mdoc with a document type of `org.iso.18013.5.1.mDL`, the data elements `birthdate`, `portrait` and `resident_postal_code` from the namespace `org.iso.18013.5.1`, and if present in the MSO, the data element `DHS_compliance` from the namespace `org.iso.18013.5.1.aamva` in the response. The data elements `birthdate` and `resident_postal_code` are retained by the verifier.

```
{
  "format":"mdoc_mso",
  "docType":"org.iso.18013.5.1.mDL",
  "nameSpaces":{
    "org.iso.18013.5.1":{
      "birthdate":{
        "required":true,
        "intentToRetain":true
      },
      "portrait":{
        "required":true
      },
      "resident_postal_code":{
        "intentToRetain":true
      }
    },
    "org.iso.18013.5.1.aamva":{
      "DHS_compliance":{
        "requiredIfPresent":true
```

```
      }
    }
  }
}
```

The following is a non-normative example of a `Query` object that requires an mdoc with a document type of `org.iso.18013.5.1.mDL`, the data elements `age_over_18`, `portrait` and `resident_postal_code` from the namespace `org.iso.18013.5.1`, and if present in the MSO, the data element `DHS_compliance` from the namespace `org.iso.18013.5.1.aamva` in the response. The data elements `age_over18`, `resident_state` and `resident_city` are retained by the verifier.

```
{
  "format":"mdoc_mso",
  "nameSpaces":{
    "org.iso.18013.5.1":{
      "age_over_18":{
        "required":true,
        "intentToRetain":true
      },
      "portrait":{
        "required":true
      },
      "resident_state":{
        "intentToRetain":true
      },
      "resident_city":{
        "intentToRetain":true
      }
    },
    "org.iso.18013.5.1.aamva":{
      "DHS_compliance":{
        "requiredIfPresent":true
      }
    }
  }
}
```

The following is a non-normative example of a `PresentationRequest` object containing the queries from the previous examples as two possible options to satisfy the request:

```
{
  "queries":[
    {
      "format":"mdoc_mso",
      "docType":"org.iso.18013.5.1.mDL",
      "nameSpaces":{
        "org.iso.18013.5.1":{
          "birthdate":{
```

```
            "required":true,
            "intentToRetain":true
          },
          "portrait":{
            "required":true
          },
          "resident_postal_code":{
            "intentToRetain":true
          }
        },
        "org.iso.18013.5.1.aamva":{
          "DHS_compliance":{
            "requiredIfPresent":true
          }
        }
      }
    },
    {
      "format":"mdoc_mso",
      "nameSpaces":{
        "org.iso.18013.5.1":{
          "age_over_18":{
            "required":true,
            "intentToRetain":true
          },
          "portrait":{
            "required":true
          },
          "resident_state":{
            "intentToRetain":true
          },
          "resident_city":{
            "intentToRetain":true
          }
        },
        "org.iso.18013.5.1.aamva":{
          "DHS_compliance":{
            "requiredIfPresent":true
          }
        }
      }
    }
  ]
}
```

The following is a non-normative example of an `PresentationRequest` object containing four different queries where each one satisfies the request:

```
{
  "queries":[
    {
      "format":"mdoc_mso",
      "docType":"org.iso.18013.5.1.mDL",
```

```
      "nameSpaces":[
        {
          "org.iso.18013.5.1":{
            "birthdate":{
              "required":true
            },
            "portrait":{
              "required":true
            },
            "zip_code":{
              "required":true
            }
          }
        }
      ]
    },
    {
      "format":"mdoc_mso",
      "docType":"org.iso.18013.5.1.mDL",
      "nameSpaces":[
        {
          "org.iso.18013.5.1":{
            "birthdate":{
              "required":true
            },
            "portrait":{
              "required":true
            },
            "city":{
              "required":true
            },
            "state":{
              "required":true
            }
          }
        }
      ]
    },
    {
      "format":"mdoc_mso",
      "docType":"org.iso.18013.5.1.mDL",
      "nameSpaces":[
        {
          "org.iso.18013.5.1":{
            "age_over_18":{
              "required":true
            },
            "portrait":{
              "required":true
            },
            "zip_code":{
              "required":true
            }
          }
        }
      ]
```

```
    },
    {
      "format":"mdoc_mso",
      "docType":"org.iso.18013.5.1.mDL",
      "nameSpaces":[
        {
          "org.iso.18013.5.1":{
            "age_over_18":{
              "required":true
            },
            "portrait":{
              "required":true
            },
            "city":{
              "required":true
            },
            "state":{
              "required":true
            }
          }
        }
      ]
    }
  ]
}
```

IETF SD-JWT VC

This section defines the credential-format specific profile for IETF SD-JWT VC `Query` objects.

The `format` parameter MUST be set to `vc+sd-jwt` and the identity credential in the response MUST conform to IETF SD-JWT VC. `Query` objects conforming to this profile MUST use the following parameters:

- `vct`: OPTIONAL. If the `vct` parameter is present, the response MUST include an SD-JWT VC (as per IETF SD-JWT VC) that contains a `vct` claim value matching the `vct` parameter value. If the `vct` parameter is omitted, the SD-JWT VC in the response MAY have any `vct` claim value. If the `vct` parameter is omitted, the `claims` parameter MUST be present.
- `claims`: OPTIONAL. The `claims` parameter contains a `Claims` object describing the claims that MUST be included in the SD-JWT VC in the response. The `Claims` object is encoded as a map where each entry MUST use a SD-JWT VC claim name (as defined in IETF SD-JWT VC) as the key for that entry. The corresponding value is a map where each entry MUST be a `Constraint` object with the following entries:
    - `required`. OPTIONAL. The `required` parameter is a boolean value if set to `true`, the SD-JWT VC in the response MUST contain the claim of this `Constraint` object. If the `required` parameter is omitted or set to `false`, the SD-JWT VC in the response MAY have the corresponding claim.

- `nested`. OPTIONAL. The `nested` parameter contains a `Claims` object (see `claims` parameter) that MUST contain only entries with keys corresponding to the embedding claim. The SD-JWT VC in the response MUST have nested claims for the claim enclosing this `Constraint` object.

The following is a non-normative example of a `Query` object that requires an SD-JWT VC with a `vct` claim value of `https://credentials.example.com/identity_credential`, the claims `given_name`, `family_name` and optionally the `address` claim in the response.

```
{
  "format":"vc+sd-jwt",
  "vct":"https://credentials.example.com/identity_credential",
  "claims":{
    "family_name":{
      "required":true
    },
    "given_name":{
      "required":true
    },
    "address":{
      "required":false
    }
  }
}
```

The following is a non-normative example of a `PresentationRequest` object containing the two queries where each `Query` object satisfies the request:

```
{
  "queries":[
    {
      "format":"vc+sd-jwt",
      "vct":"https://credentials.example.com/identity_credential",
      "claims":{
        "family_name":{
          "required":true
        },
        "given_name":{
          "required":true
        },
        "address":{
          "required":false
        }
      }
    },
    {
      "format":"vc+sd-jwt",
      "vct":"https://credentials.example.com/health_card_credential",
      "claims":{
        "last_name":{
```

```
          "required":true
        },
        "first_name":{
          "required":true
        },
        "address":{
          "required":false
        }
      }
    }
  ]
}
```

The following is a non-normative example of a `Query` object that requires an SD-JWT VC with a `vct` claim value of `https://credentials.example.com/identity_credential`, and the nested claim `zipcode` in the response.

```
{
  "format":"vc+sd-jwt",
  "vct":"https://credentials.example.com/identity_credential",
  "claims":{
    "address":{
      "nested":{
        "zipcode":{
          "required":true
        }
      }
    }
  }
}
```

IETF SD-JWT VC (alternative proposal)

This section defines the credential-format specific profile for IETF SD-JWT VC `Query` objects.

The `format` parameter MUST be set to `vc+sd-jwt` and the identity credential in the response MUST conform to IETF SD-JWT VC. `Query` objects conforming to this profile MUST use the following parameters:

- `vct`: OPTIONAL. If the `vct` parameter is present, the response MUST include an SD-JWT VC (as per IETF SD-JWT VC) that contains a `vct` claim value matching the `vct` parameter value. If the `vct` parameter is omitted, the SD-JWT VC in the response MAY have any `vct` claim value. If the `vct` parameter is omitted, the `claims` parameter MUST be present.
- `claims`: OPTIONAL. The `claims` parameter is an array of `Claim` objects each describing a claim and conditions under which the claim MUST be included in the SD-JWT VC in the response. The `Claim` object defines the following parameters:

- `path`. REQUIRED. The `path` parameter is an array of string values which describes the path of the claim within the unsecured payload of the SD-JWT VC.
- `required`. OPTIONAL. The `required` parameter is a boolean value if set to `true`, the SD-JWT VC in the response MUST contain the claim. If the `required` parameter is omitted or set to `false`, the SD-JWT VC in the response MAY have the corresponding claim.

The following is a non-normative example of a `Query` object that requires an SD-JWT VC with a `vct` claim value of `https://credentials.example.com/identity_credential`, the claims `given_name`, `family_name` and optionally the `address` claim in the response.

```
{
  "format":"vc+sd-jwt",
  "vct":"https://credentials.example.com/identity_credential",
  "claims":[
    {
      "path":["family_name"],
      "required":true
    },
    {
      "path":["given_name"],
      "required":true
    },
    {
      "path":["address"],
      "required":false
    }
  ]
}
```

The following is a non-normative example of a `PresentationRequest` object containing the two queries where each `Query` object satisfies the request:

```
{
  "queries":[
    {
      "format":"vc+sd-jwt",
      "vct":"https://credentials.example.com/identity_credential",
      "claims":[
        {
          "path":["family_name"],
          "required":true
        },
        {
          "path":["given_name"],
          "required":true
        },
        {
          "path":["address"],
```

```
          "required":false
        }
      ]
    },
    {
      "format":"vc+sd-jwt",
      "vct":"https://credentials.example.com/health_card_credential",
      "claims":[
        {
          "path":["first_name"],
          "required":true
        },
        {
          "path":["last_name"],
          "required":true
        },
        {
          "path":["address"],
          "required":false
        }
      ]
    }
  ]
}
```

The following is a non-normative example of a `Query` object that requires an SD-JWT VC with a `vct` claim value of `https://credentials.example.com/identity_credential`, and the nested claim `zipcode` in the response.

```
{
  "format":"vc+sd-jwt",
  "vct":"https://credentials.example.com/identity_credential",
  "claims":[
    {
      "path":["address", "zipcode"],
      "required":true
    }
  ]
}
```

W3C VCDM 2.0

This section defines the credential-format specific profile for W3C VCDM `Query` objects.

The `format` parameter MUST be set to `vp+ld+json` or, `vc+ld+json` and the identity credential in the response MUST conform to W3C VCDM 2.0. `Query` objects conforming to this profile MUST use the following parameters:

- `example`: REQUIRED. The verifiable presentation usually but optionally embedding a verifiable credential in the response MUST contain all properties provided by the `example` parameter value, i.e., query by example. The `example` parameter value is an object that MUST be a partial verifiable credential or a partial verifiable presentation as defined in W3C VCDM 2.0. If the `format` parameter value is set to `vp+ld+json` , the verifiable presentation in the response MUST contain the same structure and properties provided by the `example` parameter value. If the `format` parameter value is set to `vc+ld+json`, the verifiable presentation in the response MUST contain a verifiable credential with the structure and properties provided by the `example` parameter value.

The following is a non-normative example of a `Query` object that requires a verifiable presentation embedding a verifiable credential of type `https://example.com/ns/v2#IdentityCredential` and the subject claims `https://example.com/ns/v2#given_name`, `https://example.com/ns/v2#family_name` and `https://example.com/ns/v2#address`.

```
{
  "format":"vc+ld+json",
  "example":{
    "@context":[
      "https://www.w3.org/ns/credentials/v2",
      "https://example.com/ns/v2"
    ],
    "type":[
      "VerifiableCredential", "IdentityCredential"
    ],
    "credentialSubject":{
      "givenName":{

      },
      "familyName":{

      },
      "address":{

      }
    }
  }
}
```

The following is a non-normative example of a `PresentationRequest` object containing the two queries where each `Query` object satisfies the request:

```
{
  "queries":[
    {
```

```
      "format":"vc+ld+json",
      "example":{
        "@context":[
          "https://www.w3.org/ns/credentials/v2",
          "https://example.com/ns/v2"
        ],
        "type":[
          "VerifiableCredential", "IdentityCredential"
        ],
        "credentialSubject":{
          "givenName":{

          },
          "familyName":{

          },
          "address":{

          }
        }
      }
    },
    {
      "format":"vc+ld+json",
      "example":{
        "@context":[
          "https://www.w3.org/ns/credentials/v2",
          "https://example.com/ns/v2"
        ],
        "type":[
          "VerifiableCredential", "HealthCredential"
        ],
        "credentialSubject":{
          "firstName":{

          },
          "lastName":{

          },
          "address":{

          }
        }
      }
    }
  ]
}
```

CWT Example

This section defines the credential-format specific profile for … `Query` objects.

The `format` parameter MUST be set to `tbd` and the identity credential in the response MUST conform to IETF CWT. `Query` objects conforming to this profile MUST use the following parameters:

- `TBD`: OPTIONAL.

The following is a non-normative example of a `Query` object that requires a CWT with a ….

```
{
  "format":"tbd",
  "tbd":"tbd",
}
```

Example protocols

OID4VP

This document defines the following new parameters for OID4VP authorization requests:

- `presentation_request`: OPTIONAL. The value for the `presentation_request` parameter is a `PresentationRequest` object and is used to specify requirements for credentials included in the `vp_token`. If the `presentation_definition` parameter is present, the `presentation_request` parameter MUST NOT be present. If the `presentation_request` parameter is present, the `presentation_definition` MUST NOT be present.

The following is a non-normative example of an OID4VP authorization request using the `presentation_request` parameter:

```
  GET /authorize?
    response_type=vp_token
    &client_id=https%3A%2F%2Fclient.example.org%2Fcb
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &presentation_request=%7B%22format%22%3A%22vc%2Bsd-jwt
        %22%2C%22vct%22%3A%22https%3A%2F%2Fcredentials.example.com
        %2Fidentity_credential%22%2C%22claims%22%3A%7B
        %22family_name%22%3A%7B%22required%22%3Atrue%7D%2C
        %22given_name%22%3A%7B%22required
        %22%3Atrue%7D%2C%22address%22%3A%7B%22required%22%3Afalse%7D%7D%7D
    &nonce=n-0S6_WzA2Mj HTTP/1.1
```

Additional verifier capability discovery

TBD: well-known based on verifier identifier

- Supported encryption algorithms
- Terms of use
- Public keys???
- Credential formats
- …

DISCUSS

Scope

- Discuss whether it should be in scope:
    - (DISCUSS) Ability to express conditionality for query-components.
    - (DISCUSS) Ability to express criticality such as required, required if present and optional for selectors.
    - (DISCUSS) Ability to express the reason why certain selectors and claims are required.

Use cases

(DISCUSS) Scenarios with conditional criticality information

To support cross-ecosystem use cases where specific identity credentials have additional claims for users within that ecosystem, the query syntax MUST support conditional criticality based on the presence of claims in the identity credential.

The following scenarios are considered:

- As a verifier, I want to request a number of claims from the user's mobile driving license AND the real ID if the claim is present in the user's mobile driving license.
- As a verifier, I want to request a number of claims from the user's mobile driving license AND the DHS compliance claim if the claim is present in the user's mobile driving license.

(DISCUSS) Educated consent

To ask for educated consent and to enforce privacy-by-design best practices, the verifier MUST be able to express the reason why specific identity credentials or specific claims or set of claims is required, as well as their intent to retain the disclosed information.

The following scenarios are considered:

- As a verifier, I want to request the motorcycle club membership card without any claims to get access to the members club area.
- As a verifier, I want to request a number of claims from the user's mobile driving license where some claims are required to rent a car and the address claim is optional in case the user wants to get their invoice sent to their home address where no shared information will be retained.

Request parameters

- `rule`: OPTIONAL. The `rule` parameter value is a String that indicates how to logically combine the selectors contained in the `selectors` parameter. If `rule` parameter is omitted, a default value of `allOf` MUST be assumed and applied to the `selectors` parameter. The `rule` parameter MUST NOT be present if the `selectors` parameter is omitted. This document defines the following values for the `rule` parameter:
  - `allOf`. If the `rule` parameter is set to `allOf`, the identity credential in the response MUST satisfy all of the selectors in the `selectors` parameter.
  - `oneOf`. If the `rule` parameter is set to `oneOf`, the identity credential in the response MUST satisfy at least one of the selectors in the `selectors` parameter.

Tobias Proposal

```
{
  // web origin of verifier
  "verifier":"https://verifier.example.com",
  "requests":[ {
        "format":"mso_mdoc”,
        “docType”: “....”,
        "rule": “oneOf”,
        "queries":[
        {
                "nameSpaces":{
                        "org.iso.18013.5.1":{
                                "document_number":{
                                        // General annotations
                                        "required":true
                                },
                                "resident_address":{
                                        // General annotations
                                        "requiredIfPresent":true,
                                        "intentToRetain":true
                                }
                        }
                }

        },
      ]
    }
  ]
}
}
```