From an IPR perspective, we are granting only the view rights to this public document. Please send a note to yasudakristina[@]gmail.com if you would like to make a comment on this document. You will be granted comment rights once OIDF can confirm you have signed Contribution Agreement.

—

Extension to OpenID4VP to support transaction data

This extension enables using EUDIW for user identification and obtaining authorization, for example to complete a payment transaction, or to sign specific document(s) using QES (Qualified Electronic Signatures).

Requirements / Assumptions / Motivations

- **Payment confirmation:** Verifier (acting as a Relying Party to the Wallet) needs to authenticate the user and require the user (identified by a digital credential) to confirm the payment amount and beneficiary.
- **QES signing:** Verifier needs confirmation which user (identified by a digital credential depending on specific obligations and/or policies put in place by the Qualified Trusted Service Providers (QTSP)) has authorized signing of which document using a digital credential and, if needed, has authorized issuing a qualified certificate to him/her. In this use-case, strong identification (LoA High) and strong authentication (SCAL 2) are required for the end user.
- **Terms of Services Confirmation:** Verifier needs confirmation that the user has read terms of services.

Binding between the user identification/authentication and user's authorization is achieved by signing the document data with the user-controlled key used for proof of possession of the digital credential being presented as a means for user identification/authentication.

Technical Proposal Overview

When requesting presentation of a digital credential for the purpose of identifying/authenticating the user, the Verifier will include data the user is confirming/authorizing in the same request. Such data can include payment transaction related information, or the document(s) that the user is authorizing to sign using QES when consenting to present requested digital credential. For the QES use-case, during such a transaction, the Verifier may also determine the need to simultaneously issue a qualified certificate and gather user consent to do so.
Upon user consent / authorization, the exact data received in the request is included in the response, inside the presentation that is signed by the key material that is used to prove possession of the digital credential used for user identification/authentication. When the credential format used is SD-JWT VC, transaction data is included in the Key Binding JWT.

Note: This proposal uses an extended "presentation_definition" to request presentation of the Credential instead of "scope", to allow the Verifier to specify which digital credential to use to include the authorization-related data during presentation (in the Key Binding JWT in case of SD-JWT VC credential format). For example, if a PID and an additional QEAA is being requested, the Verifier can convey that the wallet shall include the data about what user is authorizing in Key Binding JWT of the PID, because it is PID that is used for user identification, and not QEAA.

Also note that the examples given in this document use the SD-JWT VC credential format. The same pattern could also be implemented with any other credential format as long as the credential presentation is performed with a proof of possession of key material (typically referred to as "credential presentation").

Request (to the Wallet)

The Credential Presentation Request syntax as defined in OpenID4VP is extended with the following parameter that is included in the `presentation_definition` object in the Presentation Definition:

- `transaction_data`: REQUIRED. Array of objects that contains typed parameter sets that provide details about the transaction that the Verifier is requesting the End-User to authorize. It consists of the following parameters:
    - `type`: REQUIRED. String that is the Identifier of the transaction data type. The values defined in this document are `payment_confirmation`, `qes_authorization` and `tos_acceptance`. The first type is used to authorize a QES, and the second can be used to gather the user's consent to the terms of service of the Verifier (e.g. the QTSP). The same request may contain two objects with both type values at the same time, when the transaction authorizes the creation of QES and the simultaneous issuance of a qualified certificate (one-shot or short term qualified certificate use case).
    - `input_descriptor_ids`: REQUIRED. Array of strings each pointing to an input_descriptor that the verifier is requesting transaction data to be bound to.

transaction data of type `payment_confirmation` (WIP)

The objects in the `transaction_data` array consists of the following parameters when `type` parameter value is `payment_confirmation`:

Field	Presence	Value	Description
creditorName	REQUIRED	String	
creditorAccount	REQUIRED	Object	contains the following entries: - "bic" - "iban" - "credit_card_issuer"

instructedAmount	REQUIRED	Object	contains the following entries: - “currency”: string - “amount”: string

transaction data of type `qes_authorization`

The objects in the `transaction_data` array consists of the following parameters when `type` parameter value is `qes_authorization`:

Field	Presence	Value	Description
signatureQualifier	REQUIRED when `CredentialID` parameter is not present	String	Identifier of the signature type to be created. A set of such identifiers are defined in [CSC-API v2] section 11.11. Both `signatureQualifier` and `CredentialID` values MAY be present.
credentialID	REQUIRED when `signatureQualifier` parameter is not present	String	The unique identifier associated with the credential. Both `signatureQualifier` and `CredentialID` values MAY be present.

documentDigests	REQUIRED	JSON Array	An array composed of entries for every document to be signed (SD). This applies for both cases, where a document is signed or a digest is signed. Every entry is composed of the following elements:
			- “hash”: REQUIRED. String containing the actual base64url-encoded octet-representation of the hash of the document.
			- “hashAlgorithmOID”: REQUIRED. String containing the OID of the hash algorithm used to generate the hashes listed in documentDigests.
			- “documentLocations”: REQUIRED. An array composed of entries for every document to be signed. Every element contains the following entries:
			  - “uri”: URL to document to be signed (SD)
			  - “method”: An object with information about access requirements to url that contains the following entries:
			  - “type”: Defines the access mode . Valid values include “public”, “OTP” and “da_rp”. Other values might be defined in the future versions of this extension. When the method is “da_rp”, Driving Application RP will authenticate the user trying to access the URL.
			  - “oneTimePassword”: REQUIRED when the access-type is “OTP”. String that is a one time password.
			- “label”: OPTIONAL. String containing a human-readable description of the respective document. The Wallet will use the label element in the user consent to designate the document.
			- “dtbsr”: OPTIONAL. String containing data to be signed representation as defined in CEN EN 419241-1 and ETSI/TR 119 001:2016(base64-encoded octet).

Note: There has been a suggestion, to define additional values “EUDIW” and “eID” as methods to authorize access to the URL in “documentLocations”. When the value is “EUDIW”, the EUDIW will be asked to present a credential to gain access to the URL. When the value is “eID”, the user will be asked to use eID to gain access to the URL.

transaction data of type `tos_uri`

Object in the `transaction_data` array consists of the following parameters when `type` parameter value is `tos_acceptance`.

Field	Presence	Value	Description
tos_uri	REQUIRED	String	URL that points to a human-readable terms of service document for the end user that describes a contractual relationship between the end-user and the Verifier. This parameter is used when the qualified certificate issuance is required (i.e. The end user doesn’t have yet a QC to use for signing). The end-user accepts such ToS when consenting to the release of a digital credential and signing the specific document using QES. The Wallet SHALL display this URL to the end-user if it is provided. The value of this field MUST point to a valid web page.

Below is the non-normative example of a Presentation Request with `transaction_data` array containing an object, of a type `payment_confimation`:

```
{
  "presentation_definition": {
    "id": "d76c51b7-ea90-49bb-8368-6b3d194fc131",
```

```
    "transaction_data": [
     "<base64url encoded string of an object, decodes to below>"
     ///
     {
          "type": "payment_confirmation",
          "input_descriptor_ids": [ "german_PID", "italian_PID" ]
          "instructedAmount": {
             "currency": "EUR",
             "amount": "123.50"
          },
          "creditorName": "Merchant A",
          "creditorAccount": {
             "bic":"ABCIDEFFXXX",
             "iban": "DE02100100109307118603"
          },
          "remittanceInformationUnstructured": "Ref Number Merchant"
       }
     ///
      ],
   "input_descriptors"
  }
}
```

Below is the non-normative example of a Presentation Request with `transaction_data` array containing two objects, of types `qes_authorization` and `tos_acceptance`:

```
{
   "presentation_definition": {
          "id": "d76c51b7-ea90-49bb-8368-6b3d194fc131",
          "transaction_data": [
                   {
                       "type": "qes_authorization",
                       "input_descriptor_ids": [ "german_PID", "italian_PID" ]
                       "signatureQualifier": "eu_eidas_qes",
                       "documentDigests": [
                           {
                               "hash": "sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI=",
                               "label": "Example Contract",
                               "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1",
                               "documentLocations": [
                                   {
                                       "uri":
"https://protected.rp.example/contract-01.pdf?token=HS9naJKWwp901hBcK348IUHiuH8374",
                                       "method": {
                                           "type": "public"
                                       }
```

```
                            },
                        ],
                        "dtbsr": "VYDl4oTeJ5TmIPCXKdTX1MSWRLI9CKYcyMRz6xlaGg"
                    }
                ]
            },
            {
                "type": "tos_acceptance",
                "input_descriptor_ids": [ "german_PID", "italian_PID" ]
                "tos_uri": "<>"
            }
        ],
    "input_descriptors": [
    {
        "id": "german_PID",
        "format": {
            "vc+sd-jwt": {}
        },
        "constraints": {
           <...>
                }
            ]
        },
```

```
            {
                "id": "italian_PID",
                "format": {
                    "vc+sd-jwt": {}
                },
                "constraints": {
                   <...>
                        }
                    ]
                }
            }
        ]
    }
}
```

Response from the Wallet:

Credential Format SD-JWT VC

In credential format SD-JWT VC, the Key Binding JWT that is used to proof cryptographic key binding of a presented credential is extended with the following parameter that is included at the top level:

- `transaction_data`: REQUIRED. Array of objects that contains parameters that provide details about the transaction that the End-User authorizes. MUST be the exact value received in the Request. The Verifier shall validate whether `transaction_data` is included in the presentation of a credential that was requested using an `input_descriptor_ids` array. If `transaction_data` is included in the presentation of a credential requested using an `input_descriptor` not contained in the `transaction_data_ids`, the verifier shall reject the response.

User's authorization is documented by including a `transaction_data` parameter in the Key Binding JWT that is signed by the key material that is used to prove possession of the digital credential used for user identification/authentication.

Processing by the Wallet when type is `qes_authorization`

When the type of transaction data is `qes_authorization`, the Wallet MUST calculate the hash of the document to be signed and compare it with the hash received in the request. If the values do not match, the Wallet MUST abort the processing and return the error.

Below is a non-normative example of a Key Binding JWT when a digital credential of a credential format SD-JWT VC is returned in the VP Token (Key Binding JWT is signed using the user-controlled key that proofs possession of the digital credential):


```
{
   "nonce": "1234567890",
   "aud": "https://verifier.example.org",
   "iat": 1709573255,
   "sd_hash": "UqAzPP5Xy1ip2II2c0E4x6U1yHL7_wI5x6VBoe4S1Sk",
   "transaction_data": [
       "db7031926f79ae41106bc8b50c3e290aa94ea730b8d4fa46a64bb678321272d0"
   ]
}
```

Extending section 6.4 Error response

The Wallet MUST refuse to process any unknown transaction data type or transaction data not conforming to the respective type definition. The Wallet MUST abort processing and respond with an error invalid_transaction_data to the Verifier if any of the following are true of the objects in the transaction_data structure:

- contains an unknown transaction data type value,
- is an object of known type but containing unknown fields,
- contains fields of the wrong type for the transaction data type,
- contains fields with invalid values for the transaction data type, or
- is missing required fields for the transaction data type.