Age verification system for access to online content

Age verification protocol

Version 1

June 30, 2024

Plan de Recuperación, Transformación y Resiliencia
Financiado por la Unión Europea NextGenerationEU
España | digital 2026

AUTHOR	Ministry for the Digital Transformation and Civil Service
PROJECT	Digital Wallet [BETA]
DOCUMENT NAME	Age verification protocol Age verification system for access to online content

Document Version Control

VERSION	AUTHOR	DATE	DESCRIPTION
V1	Ministry for the Digital Transformation and Civil Service	30-06-2024	Initial version

Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This technical specification is part of the functionalities of the age verification system for controlling the access of minors to content for adults, which the Ministry for the Digital Transformation and Civil Service (MTDFP) is defining and implementing.

The main condition for this specification is that in no case must the person who proves the age of majority provide information that allows their identification or tracking on the Internet, and content platforms must therefore obtain the minimum necessary information from users, following the principle of minimizing data disclosure.

This document does not cover the process of obtaining that credential, nor the process of consulting an authentic source before generation of the credential[1].

This document details **the communication protocol between the end-user mobile application** and the content provider **for verifying the age of majority**. It includes the **technical specification of the provider of adult-restricted content** to **ensure interoperability**, at national level, between **platforms with adult content**.

With this document, both adult content providers and mobile app developers must be able to implement the necessary information flows following the standards and protocols defined for this purpose.

This document assumes that there are mobile applications, of which Digital Wallet [BETA] is one, in which a user has securely stored a verifiable credential that proves that they are of legal age and, therefore, authorized to access adult content (in no case is the user's age shared). By means of the credential, the end user proves their age of majority in access to platforms with restricted adult content that will have the obligation to verify this attribute.

NOTE - This solution has been designed and proposed considering the current state of the art in different widely spread cryptographic technologies and the principles that are being developed in the eIDAS2[2] regulation. In any case, it is not a complete implementation of this regulation, which is still being developed.

Work will continue on all processes and improvements made as the eIDAS2 regulation and ZKP[3] technologies evolve.

[1] The protocol that details how Digital Wallet [BETA] manages the issuance of the credentials and their presentation is in the document "Specification for Use of the Credential of Age of Majority"

[2] Regulation (EU) 2024/1183 of the European Parliament and of the Council of 11 April 2024 amending Regulation (EU) No 910/2014 as regards the establishment of the European Digital Identity Framework)

[3] Zero Knowledge Protocol, highly secure cryptographic systems focused on the minimum disclosure of information.

1.1 Scope

This document includes everything necessary so that:

- **Platforms** containing adult content may **request, receive and verify age of majority credentials**.
- Adult content providers can develop the solution on both **web platforms** and **mobile apps**.
- **Mobile apps that have the "Age of Majority" credential can present it to** adult content platforms.

1.2 Dictionary

- **Authentic Information sources:** Sources from which the credential issuer extracts the attributes that are subsequently included in the Verifiable Credential.
- **Decentralized Identifier (DID):** Decentralized identifier defined in [DID Core]. The DIDs of this solution will be generated by the method specified in [DID-Key].
- **Request for evidence:** Request made by a verifier notifying the owner of the credentials which credentials they must submit to the verifier and the format they must use.
- **Evidence:** Response to the request for evidence made by a verifier in which the requested credential or credentials are included.
- **Whitelists:** This is a list of entities that are considered trusted to perform certain actions, such as requesting certain types of credentials for subsequent verification.
- **Deep Link:** This is a URL that takes a user directly to a specific location within an app or website, rather than just opening the homepage.
- **Universal Link:** This is a technology introduced by Apple for iOS, which allows web links to directly open relevant content in an app, if it is installed, or in the web browser, if the app is not installed.
- **App Link:** This is the equivalent of *Universal Links* but for the Android platform. Introduced by Google, *App Links* allow web URLs to be opened directly in a specific app if it is installed, and in the web browser if it is not.

1.3 Actors

The following diagram shows the actors in the ecosystem:

- **Credential Issuer:** Solution that generates the credential of the age of majority. First, it will extract the necessary data from the authentic sources of information to generate a credential of majority that certifies that the holder is over eighteen years old (according to the Spanish Constitution). This credential will be secured by the signature of the issuer, so that third parties can validate the identity of the issuing entity.

- **Digital Wallet [BETA] Mobile App:** mobile application developed by the Ministry for the Digital Transformation and Civil Service which, in the context of this document, will contain the pair of cryptographic keys, public and private, and the credential of the age of majority.
- **Content provider (Verifier):** Component that requests the adult credential from the end user and performs the corresponding verifications to allow or deny access to adult-restricted content.

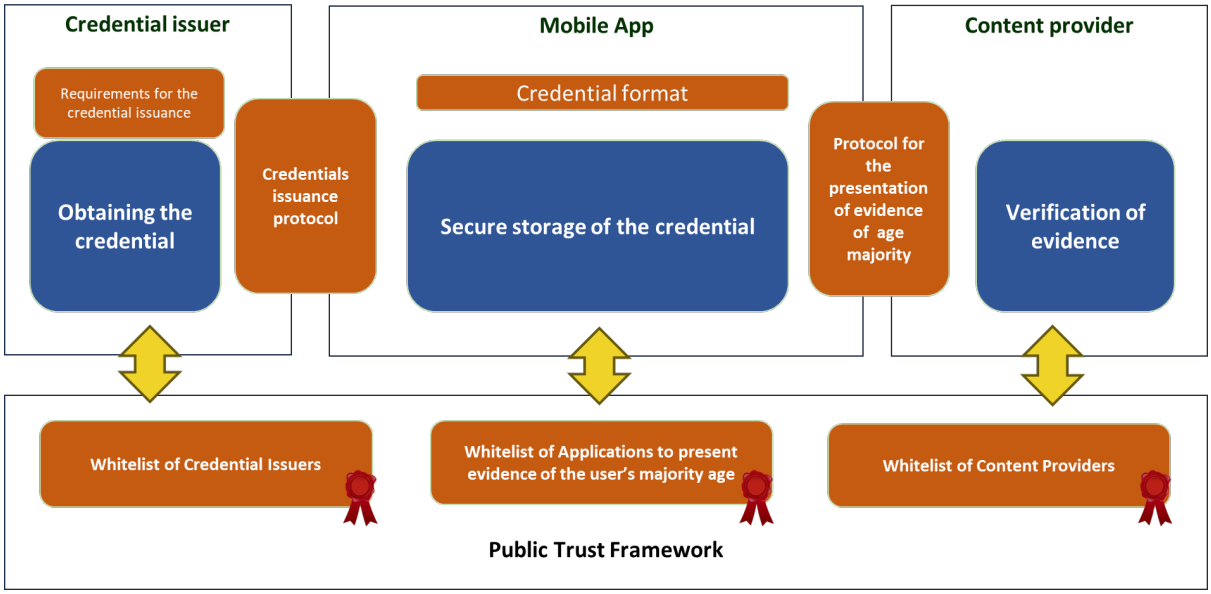


Figure 1. General solution components

The diagram shows the communication protocols of the general solution such as the credential issuance protocol by which the mobile application obtains the credential of majority and the presentation protocol of the credential of majority through which the mobile application submits the credential to the content provider to gain access to adult-restricted content. Since this document aims to define the technical solution for requesting and verifying the credential of majority, it will only describe the protocol for submitting the credential of majority, which will be carried out following the specification **OpenID For Verifiable Presentations** [OpenID4VP], as well as in the model of the data shared by both parties during this communication.

The trust framework is based on whitelists managed by the root certification authority. From the perspective of the content provider, it is only necessary to deal with the list of trusted content providers, where they must register in advance and the list of trusted issuers, which must be consulted to verify that the credential of majority has been issued by a trusted issuer.

1.4 General flow

The following diagram represents the high-level communication flow of the previously introduced actors, who make up the common technical solution. The green box highlights the technical solution addressed in this document.

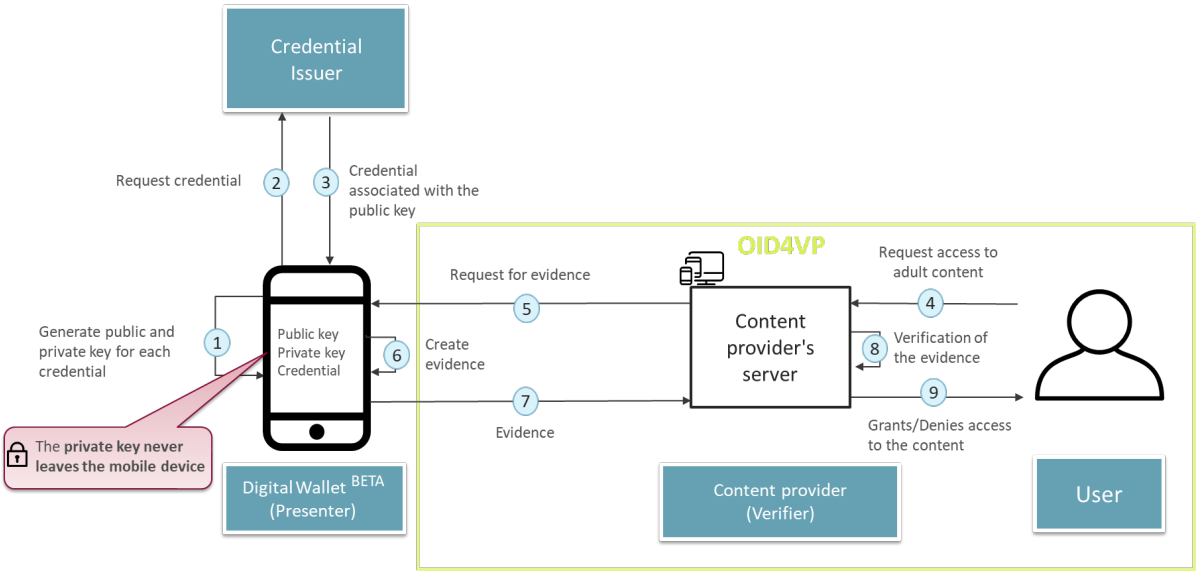


Figure 2. General flow: OID4VP

To undertake the part of the access flow to restricted content for adults, the end user must first have a verifiable credential of majority; that is, the following issue flow must have been previously carried out:

1. For each credential requested, the mobile device will generate a pair of keys, public and private, mathematically related to each other. The private key will never leave the device while the public key will be provided to the credential issuer in the credential request.
2. The credential is requested by providing the DID, a decentralized identifier generated from the previously generated public key.
3. The issuer issues the verifiable credential associated with the DID, which identifies the holder of the private key associated with that public key as the credential holder. The fact that the credentials are associated with the public keys provides traceability since all the credentials that are presented to content providers could be correlated using the public key to which they are associated. To decrease traceability, a pair of keys is generated for each credential so that traceability is reduced to a single credential, and it is impossible to correlate credentials between them. The document "Specification for use of the age of majority credential" explains the solution adopted in Digital Wallet BETA to minimize traceability.
4. Once the credential of majority is available, the end user can request access to the restricted adult content.
5. The request for access by the end user results in a request for evidence generated by the content provider that will receive the Digital Wallet BETA mobile application with

indications on the credential requested, format, algorithms and all the details necessary to access the content.

6. Based on the request, the Digital Wallet BETA, generates the evidence.
7. The evidence is sent to the content provider.
8. The content provider verifies the evidence.
9. If the evidence submitted successfully passes the verifications carried out by the content provider, the latter will give the end user access to the requested content. Otherwise, access will be denied.

2 DATA MODEL

2.1 Request for evidence

This section deals with the specification of the data model of the request for evidence made by the content provider to the Digital Wallet BETA mobile application once the end user requests access to the adult-restricted content.

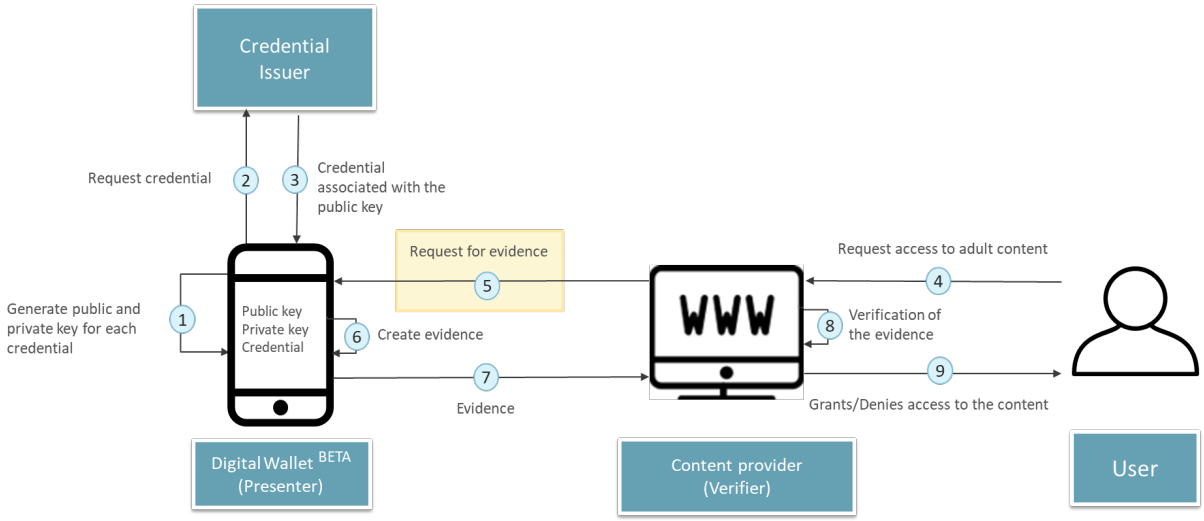


Figure 3. General flow: Request for evidence

The request for evidence will contain the URI (Uniform Resource Identifier) that the mobile application will use to obtain the parameters of the request for evidence. The URI that references the application data is constructed by adding the following parameters to the authorization *endpoint* URL using the *application/x-www-form-urlencoded* encoding:

- `request_uri`
  - The absolute URI is the URL that references the parameters of the request for evidence; that is, the URL to which the mobile application will make a request to obtain the object that contains the parameters of the request for evidence.
- `client_id`
  - The value must match that set in the `client_id` field of the object containing the parameters of the evidence request.

This URI references the parameters contained in the request for evidence, it may be either a deep link, universal link or app link and must not exceed 521 ASCII characters.

The protocol scheme configured in the Digital Wallet BETA application is *ageverification*, so the following is a non-normative example of a request for evidence generated as a *deep link*:

```
ageverification://authorize? client_id=https%3A%2F%2Fwwww.todoporno.es%2F
postpresvp&request_uri=https%3A%2F%2F
wwww.todoporno.es%2Frequest.json%2FGkurKxf5T0Y-
mnPFCHqWOMiZi4VS138cQO_V7PZHAdM
```

2.2 Object of the request for evidence

This section details the data model of the object that contains the parameters of the request for evidence, that is, the data model that will be returned when the Digital Wallet BETA mobile application performs a GET to the URL provided in the `request_uri` field of the request for evidence specified in the previous section.

```
{
    "response_type": "vp_token",
    "client_id_schema": "redirect_uri",
    "response_mode": "direct_post.jwt",
    "response_uri": "${URI de vuelta}",
    "client_id": "${response_uri}",
    "nonce": "07d54d63-7136-3ff1-11d8-f 9d17bdb0620",
    "presentation_definition": {
       "id": "32f54163-7166-48f1-93d8-f f217bdb0653",
        "format": {
                "jwt_vc": {
                        "alg": ["RS512"
                },
                 "jwt_vp": {
                        "alg": ["RS512"
                },
         },
         "input_descriptors": [{
                "id": "Age over 18",

               "format": {
                        "jwt_vc": {
                                "alg": ["RS512"]
                        }
                }
        }]
     }
}
```

It will contain the following fields given by the [OpenID4VP] protocol:

- `presentation_definition`:
  - This is a JSON Object defined in specification [DIF.PresentationExchange] on Presentation Exchanges. These are objects that define which tests are requested by a verifier, in this case, the content provider.
  - `id`:
    - Mandatory field. It is advisable to use a unique identifier, such as a Universal Unique Identifier (UUID) with *String* format.
  - `format`:

- Optional. Object that tells the mobile application the format configuration that the content provider can process. The content provider can accept various formats, within the formats set in [DIF.ClaimFormatRegistry] (*jwt*, *jwt_vc*, *jwt_vp*, *ldp*, *ldp_vc* or *ldp_vp*). The following is a non-normative example:

```
{
        "jwt": {
                "alg": ["EdDSA", "ES256K", "ES384"]
              },
        "jwt_vc": {
                "alg": ["EdDSA", "ES384"]
        },
        "jwt_vp": {
                "alg": ["EdDSA", "ES256K"]
        },
        "ldp_vc": {
                "proof_type": [
                  "JsonWebSignature2020",
                  "Ed25519Signature2018",
                  "EcdsaSecp256k1Signature2019",
                  "RsaSignature2018"
                ]
         },
        "ldp_vp": {
                "proof_type": ["Ed25519Signature2018"]
        },
        "ldp": {
                "proof_type": ["RsaSignature2018"]
        }
}
```

It is important to note that every service provider must support the RS512 signature algorithm, since it is the one supported by the Digital Wallet [BETA] mobile application.

- `input_descriptors`:
  - Mandatory field. Array of *Input Descriptors* type objects; objects containing the following fields:
    - `id`: Mandatory. *String* that identifies the *input descriptor*; it cannot match the `id` property of another *input descriptor* contained in the same presentation definition.
    - `format`: Optional. It is identical to the `format` property of the presentation definition but can be used if we want to restrict the format of a specific descriptor input, see `format` property for more detail.
    - `constraints`: Mandatory. This is an object with the following properties:
      - `fields`: Optional. JSON Object. The fields are processed in order, so if we want to reduce processing by checking the most relevant characteristics of the credential, the validations of these fields must be ordered first when implementing the solution. The `path` field must be present in the `fields` object; it is a list of one or more JSON*Paths*

string expressions. Additionally, optional fields described in the presentation exchange specification may be added.

        - `limit_disclosure`: Optional. It can be set as *required*, indicating that only the fields listed in `fields` can be presented or it can be set as *preferred* indicating that it is advisable to do so.

- `client_id_scheme`:
    - Value set to `redirect_uri`. Since the trust framework is based on a whitelist that will identify content providers based on the URI to which the evidence is sent, it will be this URI that will be used as `client_id` in the request for evidence.
- `nonce`:
    - Mandatory. *String* that is used to prevent replication attacks. An attacker could attempt to insert the verifiable presentation included in one submission of evidence into other; using `nonce` means these can be presented only once. It will also serve to link the request for evidence with the evidence presented by the Digital Wallet [BETA] mobile application.
- `state`:
    - Optional. Manages the session, allows the content provider to link the request to the evidence. The session will be linked to the cookie of the browser from which access was requested, so that, if the evidence is verified, that cookie will be authorized to view the content. Therefore, if a third party presents the evidence, the authorized cookie will be that of the browser of the user who requested it and not that of the user who presents the evidence.
- `response_mode`:
    - Value set to `direct_post`. This enables the mobile application to send the evidence through an HTTPS POST request, solves problems such as exceeding the URL size limit or not being able to send the response by redirection to the verifier because the content provider and the mobile application are on different devices. The parameters of the authorization response will be encoded in the body using the *application/x-www-form-urlencoded* format.
- `response_uri`: Required if the `response_mode` field is set to `direct_post`. The URI to which the mobile application must send the evidence, the URI that is used as the content provider identifier in the content provider whitelist.

It will contain the following fields inherited from the OAuth2.0 standard:

- `response_type`
    - Mandatory field set to `vp_token`, as established by the [OpenID4VP] protocol.
- `client_id`
    - Mandatory field whose objective is to identify the OAuth2.0 client, that is, the content provider. Since the `client_id_schema` is set to `redirect_uri` and considering that a redirection URI will not be used to avoid possible problems, but the `direct_post` response mode will be used to send the response

through an HTTP POST request, the `client_id` field will be equal to the `response_uri` field.

2.3 Evidence

Once the mobile application has received the request for evidence, it proceeds to generate the evidence following the requirements specified in the request.

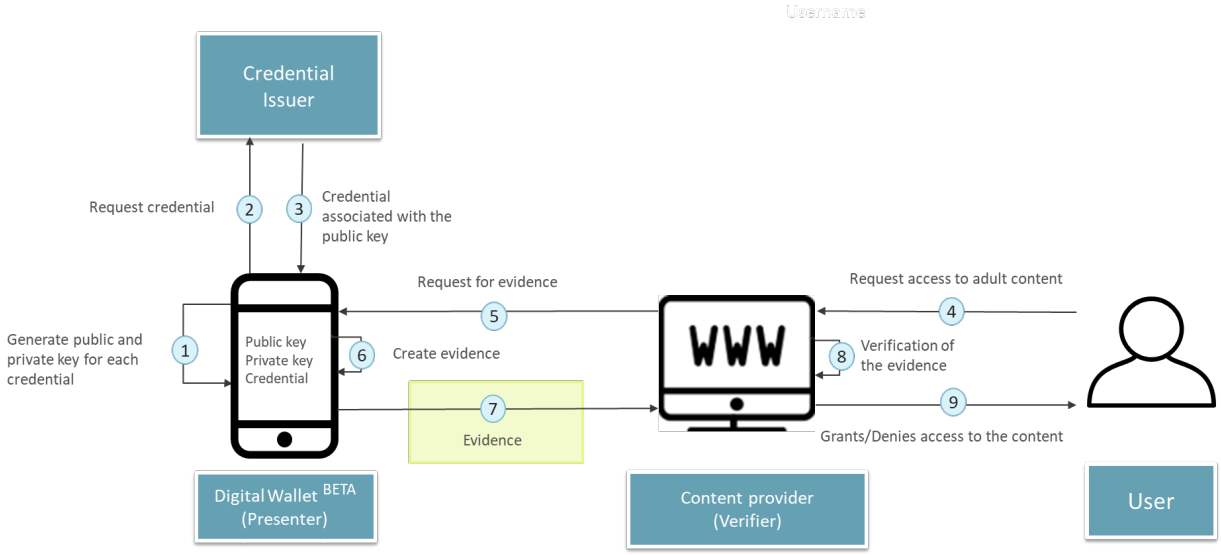


Figure 4. General flow: Evidence

The evidence is a JWT (JSON Web Token), in this case composed of a single verifiable presentation containing a verifiable credential of age of majority, as shown in the following non-normative example.

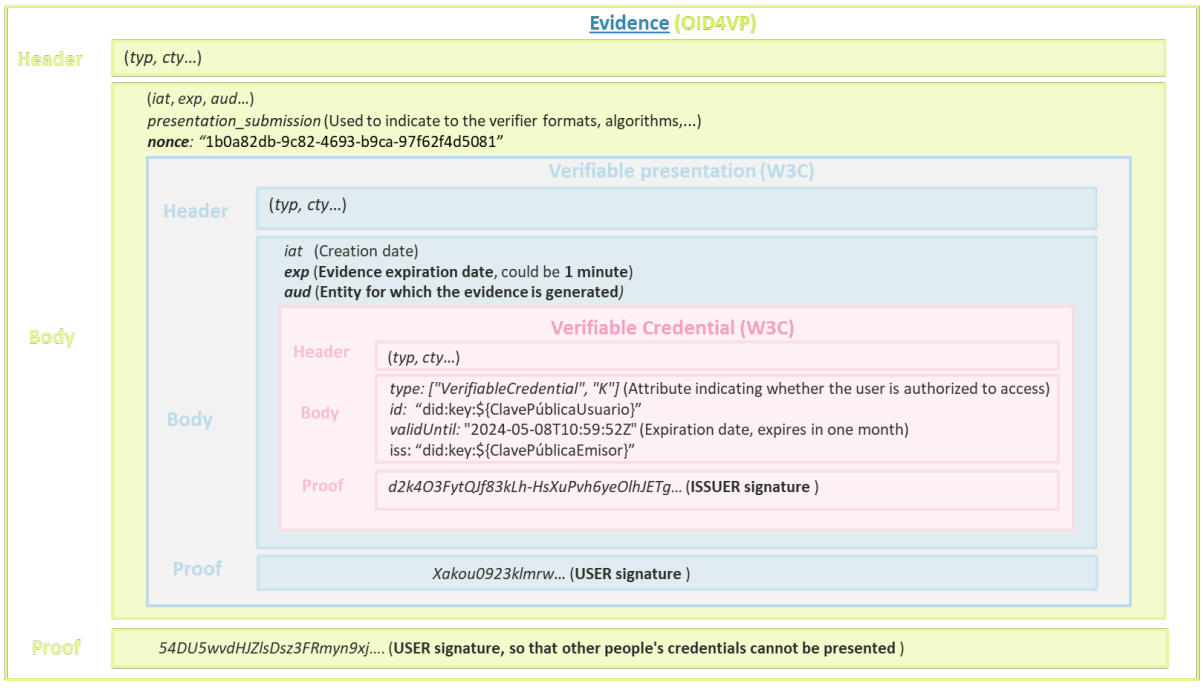


Figure 5. Evidence

In response to the request for the evidence that has been provided to the mobile application by the content provider, the evidence whose body follows the data model detailed below will be returned to the content provider.

```
{
        "vp_token": {
           "@context": "https://www.w3.org/ns/credentials/v2",
           "id":"data:application/vp+ld+json+jwt;${presentacionVerificableJWT}”,
           "type": "EnvelopedVerifiablePresentation"
        },
        "presentation_submission": {
                "id": "a30e3b91-fb77-4d22-95fa-871689c322e2",
                "definition_id": “32f54163-7166-48f1-93d8-f f217bdb0653”,
                "descriptor_map": [ {
                        "id": "Age over 18",
                        "format": "jwt_vc",
                        "path": "$.verifiableCredential[0]"
                }]
        },
        "nonce": “07d54d63-7136-3ff1-11d8-f 9d17bdb0620”
}
```

Since the `response_type` field set in the authorization request is `vp_token`, the authorization response will contain the following parameters:

- `vp_token`
  - Mandatory. Since a single Verifiable Presentation will be included, a JSON Object will include it. In the case of multiple Verifiable Presentations, it would be a list of JSON objects, each of which will include one of the Verifiable Presentations.
- `presentation_submission`
  - Mandatory. An object that contains the following fields:
    - `id`: Mandatory, it must be a unique identifier, such as a UUID.
    - `definition_id`: Mandatory. The `id` field of the presentation definition.
    - `descriptor_map`: List of *Input Descriptor Mapping* objects composed of the following fields:
      - `id`: Mandatory. String that matches the id field of the *Input Descriptor* property of the presentation definition on which the response is based.
      - `format`: Mandatory. Denotes the format of the statements and must be one of those proposed by DIF.
      - `path`: Mandatory. It must be an expression in *string* format of *JSONpath* type. Indicates the attribute with respect to the identified *Input Descriptor*.
- `nonce`:
  - *String*. It must match the `nonce` provided in the parameters of the request for evidence.
- `state`:

    - Optional. *String*. It must match the `state` field provided in the authorization request parameters. In this case, it will not be necessary since the `nonce` field will be used to link the request with the evidence.
- `exp`: Mandatory. *NumericDate*. Expiration date, after which the evidence must not be accepted.
- `aud`: Mandatory. *String*. Identifies the receiver for which the JWT has been generated. The value that comes in the `response_uri` field of the request will be used since this is the content provider identifier.

Although the `exp` and `aud` attributes are defined in RFC7519 as optional, they will be mandatory in this specification because of their relevance for the solution of the use case of majority.

The evidence received by the content provider is a JWT whose body corresponds to that described and secured by a signature made with the private key that resides on the mobile device of the end user. This is a test that demonstrates control over the private key associated with the public key of the credential holder. The signature algorithm will be ECDSA with SHA256 (ecdsaSignatureMessageX962SHA256). The P256 curve. The JWT header will contain the following properties:

- `alg`: *String*. Mandatory. Value set to ES256.

2.4 Verifiable presentation

The evidence previously detailed contains a Verifiable Presentation, in the `vp_token field`, generated by the Digital Wallet [BETA] mobile application following the W3C data model:

This is a JSON with the following properties:

```
{
   "@context": "https://www.w3.org/ns/credentials/v2",
   "id":"data:application/vp+ld+json+jwt;${presentacionVerificableJWT}”,
   "type": "EnvelopedVerifiablePresentation"
}
```

- `@context`
  - Value set to *https://www.w3.org/ns/credentials/v2*.
- `id`
  - It represents a verifiable presentation secured by the *JOSE* mechanism. The format of the field is defined in data URL schema and will be completed as follows:

```
data:application/vp+ld+json+jwt;${presentaciónVerificableJWT}
```

- `type`
  - Value set to *EnvelopedVerifiablePresentation* .

The JWT included in the `id` field of the Verifiable Presentation, JSON that contains the properties detailed below and is secured by encryption with the private key of the holder of the Verifiable Credential that is included, must be extracted and decoded, so that only the holder of the credentials can present them.

```
{
        "id": "urn:uuid:00000000-0000-0000-0000-000000000000",

"type":  [
                "VerifiablePresentation"
        ],
        "verifiableCredential":[{
                "@context" : "https://www.w3.org/ns/credentials/v2",
                "id":"data:application/vc+ld+json+jwt;${credencialVerificableJWT}”,
                "type": "EnvelopedVerifiableCredential"
        }],
        "holder":"did:key:z2dmzD81cgPx8Vki7JbuuMmFYrWPgYoytykUZ3eyqht1j9KbrSNto1XXZFRD5StnZ
        PJ1tLKTc39AJ3Ae1EW99bJhMpXJgEq8BaqpX2UCrbsxG9fDpXKLFswiEdJisHwMqhTWrMUTe7pHH8Vo3Zkt
        nujZVd7HuTCwjrvEv4m1r8yTKQt35e"
}
```

It consists of the following fields:

- `id`
  - Field reserved for future use. Value set to *urn:uuid:00000000-0000-0000-0000-000000000000*.
- `type`
  - *Strings* list. Value set to *VerifiablePresentation.*
- `verifiableCredential`

- `@context`
  - Value set to *https://www.w3.org/ns/credentials/v2*.
- `type`
  - Value set to *EnvelopedVerifiableCredential.*
- `id`
  - It represents a verifiable presentation secured by the JOSE (JavaScript Object Signing and Encryption) mechanism. The format of the field is defined in the standard [RFC2397] and will be completed as follows:

```
data:application/vc+ld+json+sd-jwt;${verifiableCredentialJWT}
```

- `holder`
  - DID of the holder of the Verifiable Credential.

2.5 Verifiable Credential

The Verifiable Presentation detailed in the previous section contains in `the verifiableCredential` field the Verifiable Credential of majority that the content provider must verify generated by the credential issuer following the W3C data model.

The JWT included in the `id` field of the Verifiable Credential; it is a JSON which contains the properties listed below and is secured by the issuer signature of the Verifiable Credential must be extracted and decoded, so that the adult-restricted content provider can verify the issuing entity of the adult-restricted content provider, as shown in the following non-normative example.

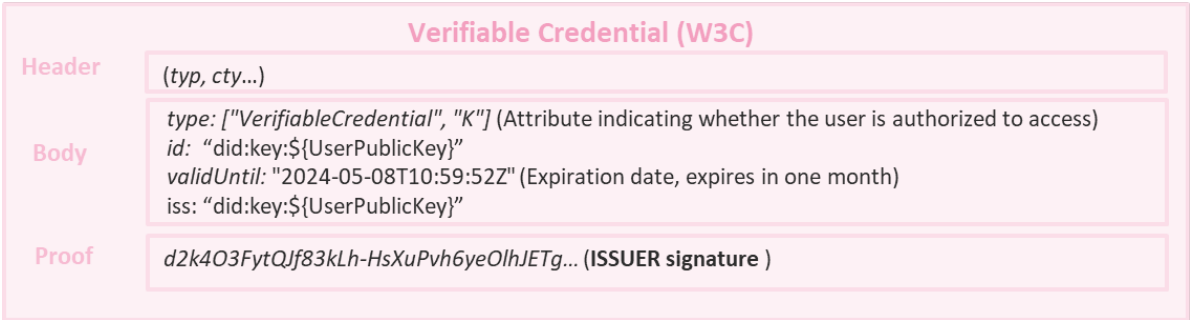


Figure 6. Type K verifiable credential

The header will include at least the following fields:


```
{
        "x5c":[ "MIIHrjCCBpagAwIBAgIQft7+0deAk5JhFQlRBPK9njANBgkqhkiG9w0BAQsFADBHMQswCQYDVQ
        QGEwJFUzERMA8GA1UECgwIRk5NVC1SQ00xJTAjBgNVBAsMHEFDIENvbXBvbmVudGVzIEluZm9ybcOhdGljb
        3MwHhcNMjEwODEyMTE0MzEzWhcNMjQwODEyMTE0MzEyWjCB7DELMAkGA1UEBhMCRVMxDzANBgNVBAcMBk1B
        RFJJRDFCMEAGA1UECgw5TUlOSVNURVJJTyBERSBBU1VOVE9TIEVDT05PTUlDT1MgWSBUUkFOU0ZPUk1BQ0l
        PTiBESUdJVEFMMTUwMwYDVQQLDCxTRUNSRVRBUklBIEdFTkVSQUwgREUgQURNSU5JU1RSQUNJT04gRElHSV
        RBTDESMBAGA1UEBRMJUzI4MDA1NjhEMRgwFgYDVQRhDA9WQVRFUy1TMjgwMDU2OEQxIzAhBgNVBAMMGlNFT
        ExPIEVOVElEQUQgU0dBRCBQUlVFQkFTMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvdOB/mRK
        zFJSZKbDrBvZoomo+Yuc+IKr9uiYIRvTgz+yEqucVRejRhMjsIh1MiT1GlRb+V9iST6pRj7t/aS8H6Sqziz
        Iy756TgsJz8GVRbOfX2A1XCN5QK0Fo96HmcADVj01M18ze+VQz7YG0Q/onbdx5IZwNncxOn3e0fGw2TEb85
        wuynhBND3ci2341+zh/zhcHEd0rMXv6NJkTi2DiS5aVx8/ou4LjFum9HmFBrIOfbVv8r+Q5W1q494HlRqGq
        /rnTyGisq3YAC2iEE/ctJJf86dza/b08lb9yFT+WBmW6Zs5Aa3CvI5+dnGrEJk/O3v+JiBMaDaMzkwFZ8Nk
        CQIDAQABo4ID7jCCA+owDAYDVR0TAQH/BAIwADCBgQYIKwYBBQUHAQEEdTBzMDsGCCsGAQUFBzABhi9odHR
        wOi8vb2NzcGNvbXAuY2VydC5mbm10LmVzL29jc3AvT2NzcFJ1c3BvbmRlcjA0BggrBgEFBQcwAoYoaHR0cD
        ovL3d3dy5jZXJ0LmZubXQuZXMvY2VydHMvQUNDT01QLmNydDCCATQGA1UdIASCASswggEnMIIBGAYKKwYBB
        AGsZgMJEzCCAQgwKQYIKwYBBQUHAgEWHWh0dHA6Ly93d3cuY2VydC5mbm10LmVzL2RwY3MvMIHaBggrBgEF
        BQcCAjCBzQyBykNlcnRpZmljYWRvIGN1YWxpZmljYWRvIGRlIHNlbGxvIGVsZWN0csOzbmljbyBzZWfDum4
        gcmVnbGFtZW50byBldXJvcGVvIGVJREFTLiBTdWpldG8gYSBsYXMgY29uZGljaW9uZXMgZGUgdXNvIGV4cH
        Vlc3RhcyBlbiBsYSBEUEMgZGUgRk5NVC1SQ00gY29uIE5JRjogUTI4MjYwMDQtSiAoQy9Kb3JnZSBKdWFuI
        DEwNi0yODAwOS1NYWRyaWQtRXNwYcOxYSkwCQYHBACL7EABATA4BgNVHREEMTAvpC0wKzEpMCcGCSsGAQQB
        rGYBCAwaU0VMTE8gRU5USURBRCBTR0FEIFBSVUVCQVMwDgYDVR0PAQH/BAQDAgXgMB0GA1UdDgQWBBR/WVm
        pRLawlHxewC4lJsqtEgQENzCBsAYIKwYBBQUHAQMEgaMwgaAwCAYGBACORgEBMAsGBgQAjkYBAwIBDzATBg
        YEAI5GAQYwCQYHBACORgEGAjByBgYEAI5GAQUwaDAyFixodHRwczovL3d3dy5jZXJ0LmZubXQuZXMvcGRzL
        1BEU19DT01QX2VzLnBkZhMCZXMwMhYsaHR0cHM6Ly93d3cuY2VydC5mbm10LmVzL3Bkcy9QRFNfQ09NUF9l
        bi5wZGYTAmVuMB8GA1UdIwQYMBaAFBn4WC8U1qbMmwSYCA1M16sAp4NlMIHgBgNVHR8EgdgwgdUwgdKggc+
        ggcyGgZ5sZGFwOi8vbGRhcGNvbXAuY2VydC5mbm10LmVzL0NOPUNSTDEsT1U9QUMlMjBDb21wb25lbnRlcy
        UyMEluZm9ybWF0aWNvcyxPPUZOTVQtUkNNLEM9RVM/Y2VydGlmaWNhdGVSZXZvY2F0aW9uTGlzdDtiaW5hc
        nk/YmFzZT9vYmplY3RjbGFzcz1jUkxEaXN0cmlidXRpb25Qb2ludIYpaHR0cDovL3d3dy5jZXJ0LmZubXQu
        ZXMvY3Jsc2NvbXAvQ1JMMS5jcmwwDQYJKoZIhvcNAQELBQADggEBACSyNnj3Kd05oB7fScFj+2sy5CKtHry
        iCIwrlLAGHWS0rik20WSrx1+hgkRJxgsKvqH5mxDzddWfg93+NwEP6JCHPMtPqfWMaLHGuHQfRYxKJx00ak
        ufhgH2knpAEFJhB5SGEU0gQuRSijvF5+a+VhX9evLt3RwnGYIwcPo34MTlr+PHR7PTgzjGxYn/8g8etrOeG
        e3eZAIp0YNTDSUA1BLSdhDjM6zP7DdTOFJGcfu3pJAx3EpuEBaSQlyxc2FWDqWrP6xPKgz5VdqsJ4Zlekka
        0SRI+ub/dNQ4I44Pj+qs15FWSUF3Mt+VBNizg2omHpdcccB2xpgBsBN5k0GzT50="],
        "kid":
        "MIIBCgKCAQEAvdOB/mRKzFJSZKbDrBvZoomo+Yuc+IKr9uiYIRvTgz+yEqucVRejRhMjsIh1MiT1GlRb+V
        9iST6pRj7t/aS8H6SqzizIy756TgsJz8GVRbOfX2A1XCN5QK0Fo96HmcADVj01M18ze+VQz7YG0Q/onbdx5
        IZwNncxOn3e0fGw2TEb85wuynhBND3ci2341+zh/zhcHEd0rMXv6NJkTi2DiS5aVx8/ou4LjFum9HmFBrIO
        fbVv8r+Q5W1q494HlRqGq/rnTyGisq3YAC2iEE/ctJJf86dza/b08lb9yFT+WBmW6Zs5Aa3CvI5+dnGrEJk
        /O3v+JiBMaDaMzkwFZ8NkCQIDAQAB",
        "alg": "RS512"
}
```


- `x5c`
  - Contains the X.509 public key certificate or the chain of certificates corresponding to the key used to sign the body of the credential. Field defined in section 4.1.6 of the standard [RFC7515].
- `kid`
  - Field defined in section 4.1.4 of standard [RFC7515]. Content providers will ignore this field, the public key will be extracted from the certificate present in the `x5c` field.
- `alg`
  - The signature algorithm used by the credential issuer. Value set to *RS512*. Field described in section 3.1 of standard [RFC7518].

The credential body comprises the following fields:

```
{
    "@context": ["https://www.w3.org/ns/credentials/v2"],
    "id": "urn:uuid:00000000-0000-0000-0000-000000000000",
    "type": ["VerifiableCredential", "K"],
    "credentialSubject": {
        "id": "did:key:${ClavePúblicaUsuario}"
    },
    "validFrom": "2023-01-01T00:00:00Z",
    "validUntil": "2024-05-08T10:59:52Z",
    "issuer": "did:key:${ClavePúblicaEmisor}",
}
```

- `@context`
  - Value set to *https://www.w3.org/ns/credentials/v2* .
- `type`
  - Value set to *["VerifiableCredential", "K"].* Type K indicates that it is a credential that accredits the user as of legal age, and therefore, access to adult-restricted content must be granted.
- `credentialSubject`
  - JSON that expresses attributes of the user formed by the following property:
    - `id`: DID of the credential holder.
- `validFrom`
  - It is *an* XMLSCHEMA11-2 *dateTimeStamp string* that represents the date and time the credential becomes valid.
- `validUntil`
  - It is a XMLSCHEMA11-2 *dateTimeStamp string* that represents the date and time when the credential is no longer valid.
- `issuer`
  - Key type DID of the credential issuer specified in the [DID-Key] standard.

Finally, the body of the credential will be secured following the JOSE signature format specified in standard [RFC7515] with the *RS512* signature algorithm.

2.6 Whitelists

The trust framework of this solution is based on whitelists in which all the entities considered trustworthy are registered to perform an action, such as issuing certain types of credentials. There will be two whitelists, one of trusted issuers and another of content providers in which they will be accredited to issue and request credentials of a certain type respectively. These two whitelists are automatically consulted and verified by the actors involved in the process and will reside on public servers.

Both whitelists are an electronically signed JSON file following the JOSE signature format specified in standard [RFC7515] using the Certificate of Seal of the whitelist manager. The header will contain the following fields:

- `x5c`: defined in section 4.1.6 of the standard [RFC7515].
- `kid`: Public key identifier defined in section 4.1.4 [RFC7515].

- `alg`: The signature algorithm used. Value set to RS512, as described in standard [RFC7518], point 3.1.

2.6.1 Issuer Whitelist

Below is a model with all the enabled labels of the whitelist against which the issuers of the verifiable credentials must be validated:

```
{
  "trustIssuersStatusList": {
    "id": "TISL20240326",
    "nextUpdate": {
      "dateTime": "2024-09-22T00:00:00Z"
    },
    "distributionPoints": {
      "uri": [
        "URI de publicación de la lista blanca"
      ]
    },
    "schemeInformation": {
      "tislVersionIdentifier": "5",
      "schemeName": [
        {
          "lang": "en",
          "text": "EN:Trusted list including information related to trusted issuers."
        },
        {
          "lang": "es",
          "text": "ES:Lista de confianza que incluye información relacionada con los emisores de confianza."
        }
      ]
    },
    "trustIssuerList": [
      {
        "issuerName": [
          {
            "lang": "es",
            "text": "Organismo responsable de la emisión"
          },
          {
            "lang": "en",
            "text": "Issuing authority"
          }
        ],
        "issuerAddress": {
          "postalAddress": [
            {
              "streetAddress": "Registered address of the issuer",
              "locality": "Madrid",
              "stateOrProvince": "Madrid",
              "postalCode": "28020",
              "countryName": "ES",
              "lang": "en"
            },
            {
              "streetAddress": "Domicilio social del emisor",
              "locality": "Madrid",
              "stateOrProvince": "Madrid",
              "postalCode": "28020",
              "countryName": "ES",
              "lang": "es"
            }
          ],
          "electronicAddress": [
            {
              "lang": "es",
              "text": "Portal web del emisor"
            },
            {
              "lang": "es",
              "text": "Dirección de correo del emisor"
            }
          ]
        },
        "schemeTerritory": "ES",
        "authorizedToIssue": [
          "K",
          "UD"
        ],
```

```
    "policyOrLegalNotice": {
      "tislLegalNotice": [
        {
          "lang": "en",
          "text": "The applicable legal framework for the present trusted list"
        },
        {
          "lang": "es",
          "text": "El marco jurídico aplicable a la presente lista de confianza"
        }
      ]
    },
    "serviceDigitalIdentities": [
      {
        "digitalId": {
          "did": "did:key:Z2xvYmFsLmNvbS9xdmJlY2FnMi5jcnQwLQYIKwYBBQUHMAGGIWh0dHA6Ly91dy",
          "x509Certificate":
            "MIIHLjCCBRagAwIBAgIUP4a8WPhZN++7VkfYyuAcDGmTglEwDQYJKoZIhvcNAQELBQAwczELMAkGA1UEBhMCQkUxGTAXBgNVBGEMEE5UUkJFLTA1Mzc2OTgzMTgx
            IDAeBgNVBAoMF1F1b1ZhZGlzIFRydXN0bGluayBCVkJBMScwJQYDVQQDDB5RdW9WYWRpcyBCZWxnaXVtIElzc3VpbmcgQ0EgRzIwHhcNMjEwMTIxMDg0ODM4WhcNM
            jQwMTIxMDg1ODAwWjCBwDELMAkGA1UEBhMCQkUxLDAqBgNVBAsMI0RpcmVjdG9yYXRlLUdlbmVyYWwgZm9yIEluZm9ybWF0aWNzMRkwFwYDVQRhDBBOVFJCRS0wOT
            Q5MzgzMzQyMUowSAYDVQQKDEFDT01NSVNTSU9OIERFIEwnVU5JT04gRVVST1BFRU5ORSAtIENPTU1JU1NJRSBWQU4gREUgRVVST1BFU0UgVU5JRTEcMBoGA1UEAww
            TRXVyb3BlYW4gQ29tbWlzc2lvbjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMCq40SQNaqRkbc5nsdPjLKyxghYzBLJCmeN3/ndD5mMzhy9tsP8WOps
            fi8CnMD5tasEEmS5Fj73DQ+sL9PIyfmLzBJVhuCyzwKpDiFAhAa+ec/LTyVRa6+oCh918ZUcPDVXpCtGA3t6WS0pCgrIJNkZequ2yIwUZxiyVOACYmkxmWhWnGe7L
            RaLv80PCI0QERL2BzV+mKb2tQylUphPPI8hd9LG9EwCvSSXv0tb4rK1C+lIcxuYsui7dNyfw9UHDBnuI5452c3O18N5+GSI9uiEmOI8s++fbubn61RskePhk3ISUR
            PGIAeDKV9pCZezBGysq5IlNkC1C4AtYS2eKFcCAwEAAaOCAmowggJmMB8GA1UdIwQYMBaAFIfJvDGXEnpzu37APUVRtAEllVGrMHcGCCsGAQUFBwEBBGswaTA4Bgg
            rBgEFBQcwAoYsaHR0cDovL3RydXN0LnF1b3ZhZGlzZ2xvYmFsLmNvbS9xdmJlY2FnMi5jcnQwLQYIKwYBBQUHMAGGIWh0dHA6Ly91dy5vY3NwLnF1b3ZhZGlzZ2xv
            YmFsLmNvbTBaBgNVHSAEUzBRMEQGCisGAQQBvlgBgxAwNjA0BggrBgEFBQcCARYoaHR0cDovL3d3dy5xdW92YWRpc2dsb2JhbC5jb20vcmVwb3NpdG9yeTAJBgcEA
            IvsQAEDMCkGA1UdJQQiMCAGCCsGAQUFBwMCBggrBgEFBQcDBAYKKwYBBAGCNwoDDDCBiwYIKwYBBQUHAQMEfzB9MBUGCCsGAQUFBwsCMAkGBwQAi+xJAQIwCAYGBA
            CORgEBMAgGBgQAjkYBBDATBgYEAI5GAQYwCQYHBACORgEGAjA7BgYEAI5GAQUwMTAvFilodHRwczovL3d3dy5xdW92YWRpc2dsb2JhbC5jb20vcmVwb3NpdG9yeRM
            CZW4wOwYDVR0fBDQwMjAwoC6gLIYqaHR0cDovL2NybC5xdW92YWRpc2dsb2JhbC5jb20vcXZiZWNhZzIuY3JsMB0GA1UdDgQWBBQspnrB2Vq7SM99egtZxs2jZv1Q
            OjAOBgNVHQ8BAf8EBAMCBsAwEwYKKoZIhvcvAQEJAgQFMAMCAQEwNAYKKoZIhvcvAQEJAQQmMCQCAQGGH2h0dHA6Ly90cy5xdW92YWRpc2dsb2JhbC5jb20vYmUwD
            QYJKoZIhvcNAQELBQADggIBACGQDYsjrM3nsKSpiW1SqzBlZOkvuiphSLNhv5o5V1W+maYLtXYH/fMadrP0+K2X5XB5e+8yMw4Sbrvxv4LRR/CQ2WX03KjYGcqdcI
            UAH+A/8phwA7eLyKd3ip8k8ukT9kUKMJxbqVR5O8x2vOlAM9VFirOenOI+MufFDSaNKIJKt+PDhcSxDyzjs2MRKHAJcTCs23T8fB6+Qvw3uKUgniVg4MTLFvWpL8b
            vMXX7QUyvsScskUCns918hHA4Ett7fxdWXJFFJuuculChJIf8jbXZQ/8SAoecPyADnJw0C4uGNI/zc9/DWqMSMMytFwGeOpQb0/pukdmNe3sai8SUQj4+vyhkCHKM
            D27cak9buDO7cu5yhKEuLCERTga2QvbGSeiFAa6KuyPGY1Ad9AOuz5KPyyfVG0h31FvNL0MnAib1IpZr7mPQgWOsEX9ipgQXoE2Rh11yydTiKwmyTTrp/uMOJp+6U
            D06IdgPkHmqnqaKkqzGu6RKiHLqYWX5qhWXt0EatHQ0YY+92B03cDTLMgjPHKY+wN6qPfetz0oQNmort+G2HIXb9P+Q43sqMjXpQY/I3YmM0VBmcYWUon11BOavnG
            GOCVVPBQFKU6Kt7r69li1CmfA7KgK7Jn2o/jb8BupWUPGej++MkkrFDS0T6tmNH9ZDteyO3dSyOcl1IHcs"
        }
      },
      {
        "digitalId": {
          "did": "did:key:SmgAwIBAgIUCowT7Q98oPLpaLF0FnUX3eobcgYwDQYJKoZIhvcNAQELBQAwczE",
          "x509Certificate":
            "MIIHQTCCBSmgAwIBAgIUCowT7Q98oPLpaLF0FnUX3eobcgYwDQYJKoZIhvcNAQELBQAwczELMAkGA1UEBhMCQkUxGTAXBgNVBGEMEE5UUkJFLTA1Mzc2OTgzMTgx
            IDAeBgNVBAoMF1F1b1ZhZGlzIFRydXN0bGluayBCVkJBMScwJQYDVQQDDB5RdW9WYWRpcyBCZWxnaXVtIElzc3VpbmcgQ0EgRzIwHhcNMjEwMTIxMDg0ODIzWhcNM
            jQwMTIxMDg1ODAwWjCB0zELMAkGA1UEBhMCQkUxPzA9BgNVBAsMNkRHIGZvciBDb21tdW5pY2F0aW9ucyBOZXR3b3JrcywgQ29udGVudCBhbmQgVGVjaG5vbG9neT
            EZMBcGA1UEYQwQTlRSQkUtMDk0OTM4MzM0MjFKMEgGA1UECgxBQ09NTUlTU0lPTiBERSBMJ1VOSU9OIEVVUk9QRUVOTkUgLSBDT01NSVNTSUUgVkFOIERFIEVVUk9
            QRVNFIFVOSUUxHDAaBgNVBAMME0V1cm9wZWFuIENvbW1pc3Npb24wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDtD1bnPz0vyI0gxH+Ye/UEyssx1TWy
            XmT2SjX8T0oR36yu+FvEI1YB7q7CGA2JNUxKdOPgFi4MZxj9vOoMwqR1+cbPDNzXsdzINLZCDy3mrPt3hfqmijhmqmwwXSQnO2+uDce+BYGXL74oWnqJTc+GT6SjY
            QMyeXvyff1JU4pC+QL53tcpcNf7dO5EJzHAfLDp8HILebtafvWAQA9+Psf3BT/0lHQZ1nqnyMfjBntCBG6/Cj2mtM44vldCKtRDdGno7MT5lxTvwoIcEA5Lupbdu9
            sudHsJT4iOub8QshI3bjHqP7n7q+OsN5RX5X/bmIG033j7Yn+DfwCKwE4eKsZHAgMBAAGjggJqMIICZjAfBgNVHSMEGDAWgBSHybwxlxJ6c7t+wD1FUbQBJZVRqzB
            3BggrBgEFBQcBAQRrMGkwOAYIKwYBBQUHMAKGLGh0dHA6Ly90cnVzdC5xdW92YWRpc2dsb2JhbC5jb20vcXZiZWNhZzIuY3J0MC0GCCsGAQUFBzABhiFodHRwOi8v
            dXcub2NzcC5xdW92YWRpc2dsb2JhbC5jb20wWgYDVR0gBFMwUTBEBgorBgEEAb5YAYMQMDYwNAYIKwYBBQUHAgEWKGh0dHA6Ly93d3cucXVvdmFkaXNnbG9iYWwuY
            29tL3JlcG9zaXRvcnkwCQYHBACL7EABAzApBgNVHSUEIjAgBggrBgEFBQcDAgYIKwYBBQUHAwQGCisGAQQBgjcKAwwwgYsGCCsGAQUFBwEDBH8wfTAVBggrBgEFBQ
            cLAjAJBgcEAIvsSQECMAgGBgQAjkYBATAIBgYEAI5GAQQwEwYGBACORgEGMAkGBwQAjkYBBgIwOwYGBACORgEFMDEwLxYpaHR0cHM6Ly93d3cucXVvdmFkaXNnbG9
            iYWwuY29tL3JlcG9zaXRvcnkTAmVuMDsGA1UdHwQ0MDIwMKAuoCyGKmh0dHA6Ly9jcmwucXVvdmFkaXNnbG9iYWwuY29tL3F2YmVjYWcyLmNybDAdBgNVHQ4EFgQU
            SiKJPwxZvli51U75EfhdERrDf98wDgYDVR0PAQH/BAQDAgbAMBMGCiqGSIb3LwEBCQIEBTADAgEBMDQGCiqGSIb3LwEBCQEEJjAkAgEBhh9odHRwOi8vdHMucXVvd
            mFkaXNnbG9iYWwuY29tL2JlMA0GCSqGSIb3DQEBCwUAA4ICAQCWLVCxzkKy/s+8kRvxCPvTlExlWkNGsrhCO+PXsh1UgWMdJCpc33OzL1GegNMSL9f3Hw6gqjXV+m
            JqtTmvrP1ZvXCagv2LRLeyxSYAjlVjwYkaraddydS8Or86OyfA0OIs1UruPe0lClb6rBKwqqx7LvXotk7XvPYQo0HrP57pCqlp4MIiNEpGeW3gRwFq8IBQcecO3k0
            09ad+tuTSB5I+o4t/7kEd//6voViy7d/EmlOgUZzXsJ24NV4pFAmyc2FlkEvYOqw8YvEsOV2hluvcG3wmkMEF/SY4X+kvpw+Ph7JQdWpWGlozzAFVppkIIrSM2Sxw
            2QXFQZWwah210HiVY0Ax4VkQbRJQNYmFWVizixPONWtc3PIfMz3/TvD9ZbExFfoQzVYXZPmtK2aQjG3CH2bJcefvS18YdR169au8yNKsPvlLaRa1k7Qv6Hv8iE3Gt
            QmszDwnQHlGrbxdft306gOFl8/U0ctyIf2YkG6J/uTnYiiibeDoHK6wBo3BfCJQ2X7/Iab3971k1wSCwe39w4x3Xw72HL9E3czRkcfpg+uTxt5jSJ7HL7BRdMD34y
            sjQ4gOV8h6NU4z+QQ3W4sZTQkFqHGHKZKt2dTrCjhE+I1hULvo4274heBr/5F0DBAXRi9lHBCBUV5iJDm90bxjHjKuvj893WLEFW3WPkWlRw=="
        }
      }
    ]
  }
 ]
 }
}
```


Figure 7. Whitelist of verifiable credential issuers

- `trustIssuersStatusList`: Root element that contains the entire list of trusted issuers.
- `id`: Unique identifier of the list of trusted issuers.
- `nextUpdate`: Date and time when the list of trusted issuers is expected to be updated.
- `distributionPoints`: Distribution points where the list of trusted issuers can be obtained.
    - URI: URL where the list of trusted issuers can be obtained.
- `schemeInformation`: Information associated with the scheme of the whitelist of trusted issuers.

    - `tislVersionIdentifier`: Version of the scheme of the whitelist of trusted issuers.
    - `schemeName`: Name, in the different languages, of the scheme of the whitelist of trusted issuers.
        - `lang: Language`
        - `text: Scheme name`
- `trustIssuerList`: List of trusted issuers.
    - `issuerName`: Name of issuer in different languages.
        - `lang: Language`
        - `text: Issuer name`
    - `issuerAddress`: Issuer address.
    - `postalAddress`: Postal addresses of the issuer in different languages.
        - `lang`: Language
        - `streetAddress`: Street address.
        - `locality`: City or locality.
        - `stateOrProvince`: State or province.
        - `postalCode`: Post code.
        - `countryName`: Country name.
    - `electronicAddress`: Email address of the issuer in different languages.
        - `lang`: Language.
        - `text`: Issuer email address
    - `schemeTerritory`: Territory to which the issuer's scheme belongs.
    - `authorizedToIssue`: List of credentials that the issuer is authorized to issue.
    - `policyOrLegalNotice`: Legal or policy notices associated with the issuer.
        - `tislLegalNotice`: Legal notice in a specific language.
            - `Lang`: Language.
            - `Text`: Contents of the Legal Notice.
    - `serviceDigitalIdentities`: Digital identities of the services provided by the issuer.
        - `digitalId`: Digital identifier of the service.
            - `did`: Decentralized identifier of the issuer, generated from the public key.
            - `x509Certificate`: X.509 certificate associated with the digital identity of the service.

2.6.2 Content provider whitelist

Below is a model with all the tags enabled from the whitelist in which trusted content providers must be registered before they can request the presentation of credentials:


```
{
  "trustContentProviderStatusList": {
    "id": "TCPSL20240326",
    "nextUpdate": {
      "dateTime": "2024-09-22T00:00:00Z"
    },
    "distributionPoints": {
      "uri": [
        "URI de publicación de la lista blanca"
      ]
    },
    "schemeInformation": {
      "tcpslVersionIdentifier": "5",
      "schemeName": [
        {
          "lang": "en",
          "text": "EN:Trusted list including information related to the content providers"
        },
        {
          "lang": "es",
          "text": "ES:Lista de confianza que incluye informacion relacionada con los proveedores de contenido"
        }
      ]
    },
    "trustContentProviderList": [
      {
        "contentProviderName": [
          {
            "lang": "es",
            "text": "Todo Porno España"
          },
          {
            "lang": "en",
            "text": "Todo Porno España"
          }
        ],
        "clientUri": "https://wwww.todoporno.es/postpresvc",
        "responseUri": "https://wwww.todoporno.es/postpresvc",
        "requestUri": "https://wwww.todoporno.es/request.json?id=001",
        "contentProviderAddress": {
          "postalAddress": [
            {
              "lang": "en",
              "streetAddress": "Registered address of the content provider",
              "locality": "Madrid",
              "stateOrProvince": "Madrid",
              "postalCode": "28020",
              "countryName": "ES"
            },
            {
              "lang": "es",
              "streetAddress": "Domicilio social del proveedor de contenido",
              "locality": "Madrid",
              "stateOrProvince": "Madrid",
              "postalCode": "28020",
              "countryName": "ES"
            }
          ],
```

```
        "electronicAddress": [
          {
            "lang": "es",
            "text": "https://wwww.todoporno.es"
          },
          {
            "lang": "es",
            "text": "mailto:todoporno@todoporno.es"
          }
        ]
      },
      "schemeTerritory": "ES",
      "authorizedToRequest": [
        "K",
        "UD"
      ],
      "policyOrLegalNotice": {
        "tcpslLegalNotice": [
          {
            "lang": "en",
            "text": "The applicable legal framework for the present trusted list"
          },
          {
            "lang": "es",
            "text": "El marco juridico aplicable a la presente lista de confianza"
          }
        ]
      },
      "serviceDigitalIdentities": [
        {
          "clientId": "https://wwww.todoporno.es/request.json?id=001"
        }
      ]
    }
  ]
 }
}
```


Figure 8. Whitelist of providers of adult-restricted content

- `trustContentProviderStatusList`: Root element that contains the entire list of trusted content providers.
    - id: Unique identifier of the list of trusted content providers.
- `nextUpdate`: Date and time when the list of trusted content providers is expected to be updated.
- `distributionPoints`: Distribution points where the whitelist of trusted content providers can be obtained.
    - uri: URL where the whitelist of trusted content providers can be obtained.
- `schemeInformation`: Information associated with the scheme of the whitelist of trusted content providers.
    - `tcpslVersionIdentifier`: Version of the scheme of the whitelist of trusted content providers.
    - `schemeName`: Schema name of the whitelist of trusted content providers.
        - `lang`: Language
        - `text`: Scheme name
- `trustContentProviderList`: List of trusted content providers.
    - `contentProviderName`: The name of the content provider in different languages.
        - `lang`: Language

        - `text`: Content provider's name
    - `clientURI`: Content provider’s identifying URI.
    - `responseURI`: Content provider’s response URI.
    - `requestURI`: Content provider’s request uri.
    - `contentProviderAddress`: Content provider's address.
        - `postalAddresses`: Container for the content provider’s postal addresses in different languages.
            - `streetAddress`: Street address.
            - `locality`: City or locality.
            - `stateOrProvince`: State or province.
            - `postalCode`: Post code.
            - `countryName`: Country name.
            - `lang`: Language
        - `electronicAddress`: The content provider’s email address in different languages.
            - lang: Language
            - lext: The supplier’s email address
    - `schemeTerritory`: Territory to which the content provider schema belongs.
    - `authorizedToRequest`: List of credentials that the issuer is authorized to request.
    - `policyOrLegalNotice`: Legal notices or policies associated with the content provider.
        - `tcpslLegalNotice`: Legal notice in a specific language.
            - `lang`: Language.
            - `text`: Contents of the Legal Notice.
    - `serviceDigitalIdentities`: Digital identities of the services provided by the content provider.
        - `serviceDigitalIdentity`: Digital identity of a service provided by the content provider.
            - `clientId`: Customer identifier of the service provided by the content provider.

3 INTERFACE AGREEMENT

The following table describes the agreement of interfaces of the evidence presentation protocol:

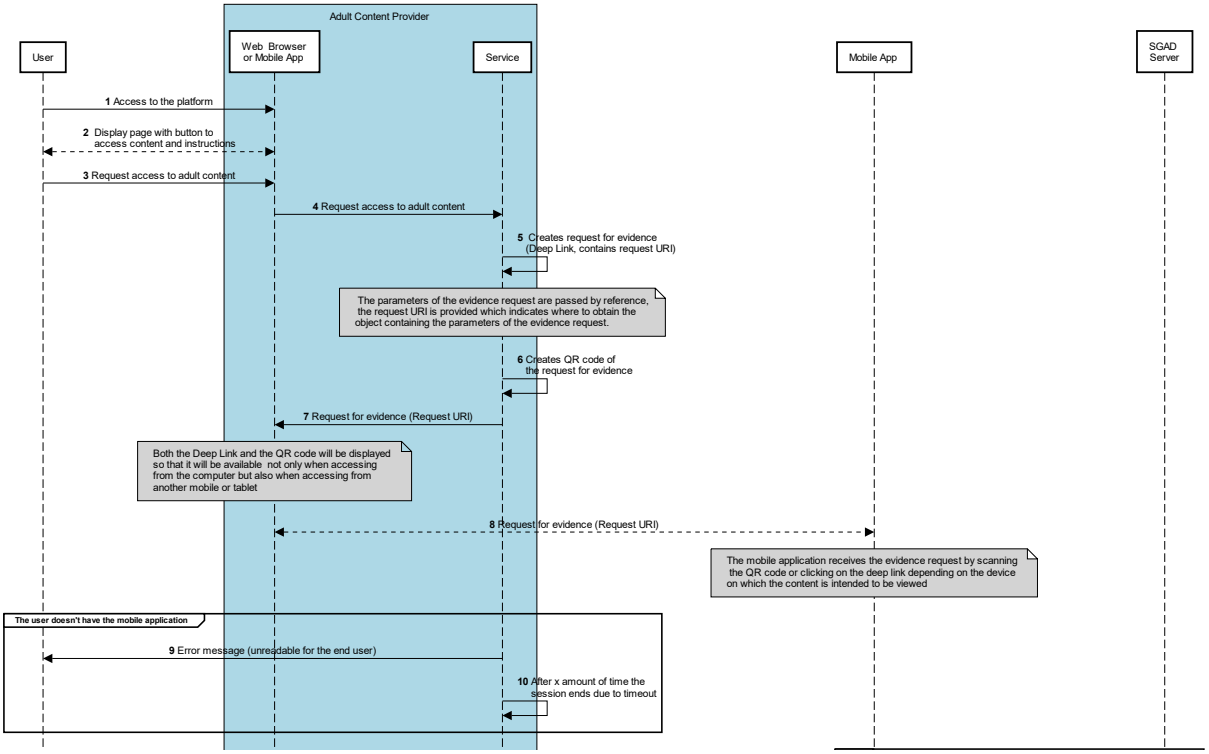
Service	Service type	URL Schema	Request		Response
			Format	Query Parameters/Body	
Evidence request	HTTP GET	ageverification://authorize	application/x-www-form-urlencoded	**Query Parameters:** •*client_id:* Response URL to which the evidence will be sent to •*request_uri:* URL referencing evidence request parameters	Not applicable
Evidence object request	HTTPS GET	*request_uri* *Obtained in the evidence request*	Not applicable	Not applicable	JSON. Object of the evidence request *See data model*
Evidence submission	HTTPS POST	*client_id* *The client_id in the evidence request object is obtained from the client_id field and must match the response_uri field of that object and the client_id field of the evidence request*	application/x-www-form-urlencoded	**Body:** • response: JWT. Evidence *See section 2.3, Evidence data model* Example: response = eyJra...9thuie	• Request processed: HTTP Status Code 200 • Bad request: HTTP Status Code 400 • Internal Error: HTTP Status Code 500

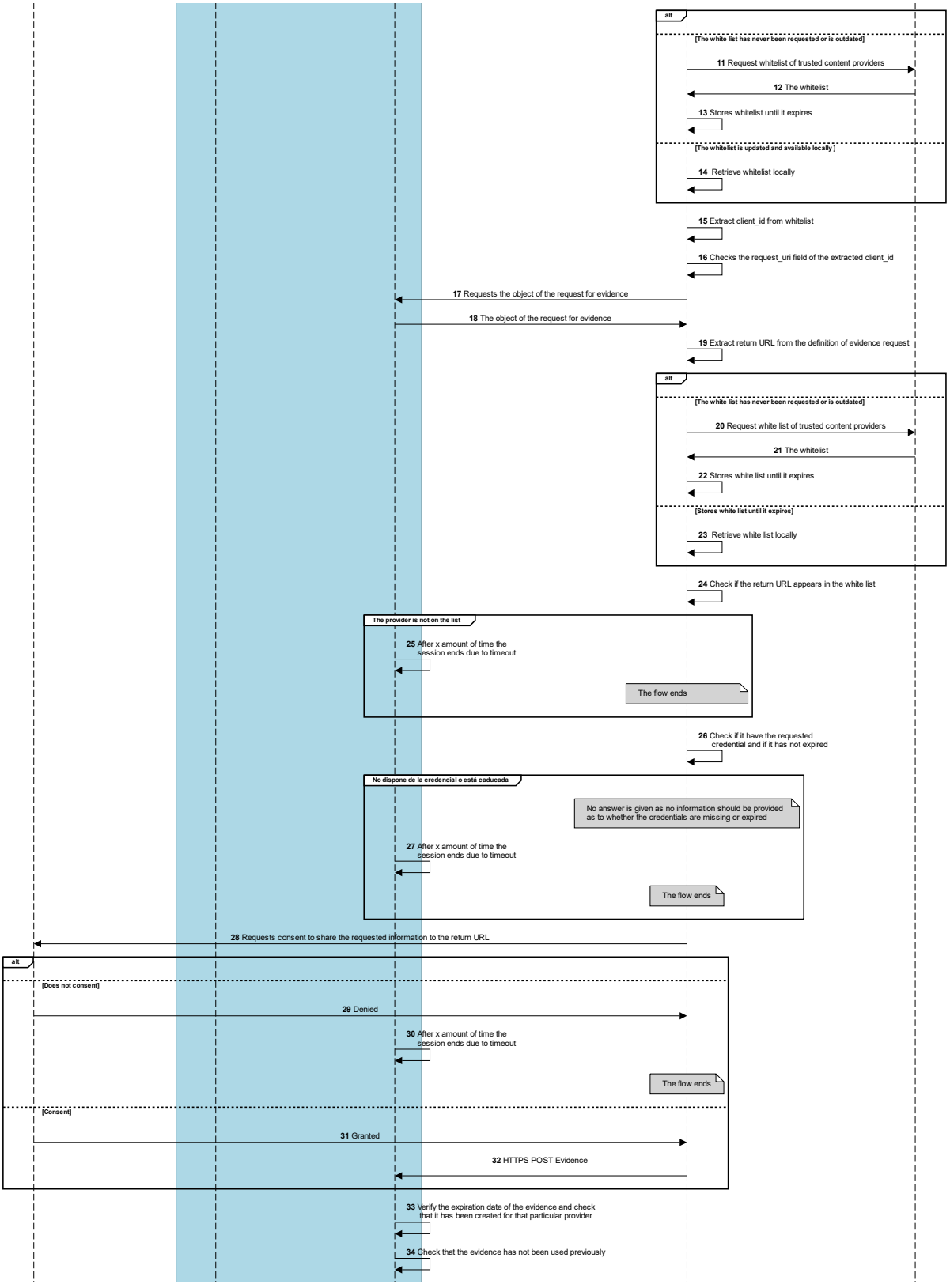
Table 1. Interface agreement

The evidence request is a redirection to the mobile application that facilitates the `client_id` and `request_uri` fields described in the previous data model. By means of an HTTPS GET request to the URI provided in the `request_uri` field, the object of the request for evidence is obtained as a response to the request. The last service is that of sending the evidence to which the evidence will be sent in JWT format within the response field in the body of the request using the *application/x-www-form-urlencoded* encoding through an HTTPS POST request to the URL indicated in the *client_id* field of the previously obtained request object. Note that the `client_id` field must match the `response_uri` field of said object and the `client_id` field of the evidence request, and that the mobile application must previously verify that the content provider with `client_id` identifier is on the whitelist of content providers.

4 FLOW OF ACCESS TO ADULT-RESTRICTED CONTENT

When the end user wants to access the adult-restricted content, either from a web browser or from a mobile application of the content provider, the communication flow between the content provider and the mobile application Digital Wallet [BETA] begins using the OpenID For Verifiable Presentations protocol [OpenID4VP] as shown in [Figure-OpenID4VP].


Adult Content Provider
User
Web Browser or Mobile App
Service
Mobile App
SGAD Server
1 Access to the platform
2 Display page with button to access content and instructions
3 Request access to adult content
4 Request access to adult content
5 Creates request for evidence (Deep Link, contains request URI)
The parameters of the evidence request are passed by reference, the request URI is provided which indicates where to obtain the object containing the parameters of the evidence request.
6 Creates QR code of the request for evidence
7 Request for evidence (Request URI)
Both the Deep Link and the QR code will be displayed so that it will be available not only when accessing from the computer but also when accessing from another mobile or tablet
8 Request for evidence (Request URI)
The mobile application receives the evidence request by scanning the QR code or clicking on the deep link depending on the device on which the content is intended to be viewed
The user doesn't have the mobile application
9 Error message (unreadable for the end user)
10 After x amount of time the session ends due to timeout


alt
[The white list has never been requested or is outdated]
11 Request whitelist of trusted content providers
12 The whitelist
13 Stores whitelist until it expires
[The whitelist is updated and available locally]
14 Retrieve whitelist locally
15 Extract client_id from whitelist
16 Checks the request_uri field of the extracted client_id
17 Requests the object of the request for evidence
18 The object of the request for evidence
19 Extract return URL from the definition of evidence request
alt
[The white list has never been requested or is outdated]
20 Request white list of trusted content providers
21 The whitelist
22 Stores white list until it expires
[Stores white list until it expires]
23 Retrieve white list locally
24 Check if the return URL appears in the white list
The provider is not on the list
25 After x amount of time the session ends due to timeout
The flow ends
26 Check if it have the requested credential and if it has not expired
No dispone de la credencial o está caducada
No answer is given as no information should be provided as to whether the credentials are missing or expired
27 After x amount of time the session ends due to timeout
The flow ends
28 Requests consent to share the requested information to the return URL
alt
[Does not consent]
29 Denied
30 After x amount of time the session ends due to timeout
The flow ends
[Consent]
31 Granted
32 HTTPS POST Evidence
33 Verify the expiration date of the evidence and check that it has been created for that particular provider
34 Check that the evidence has not been used previously

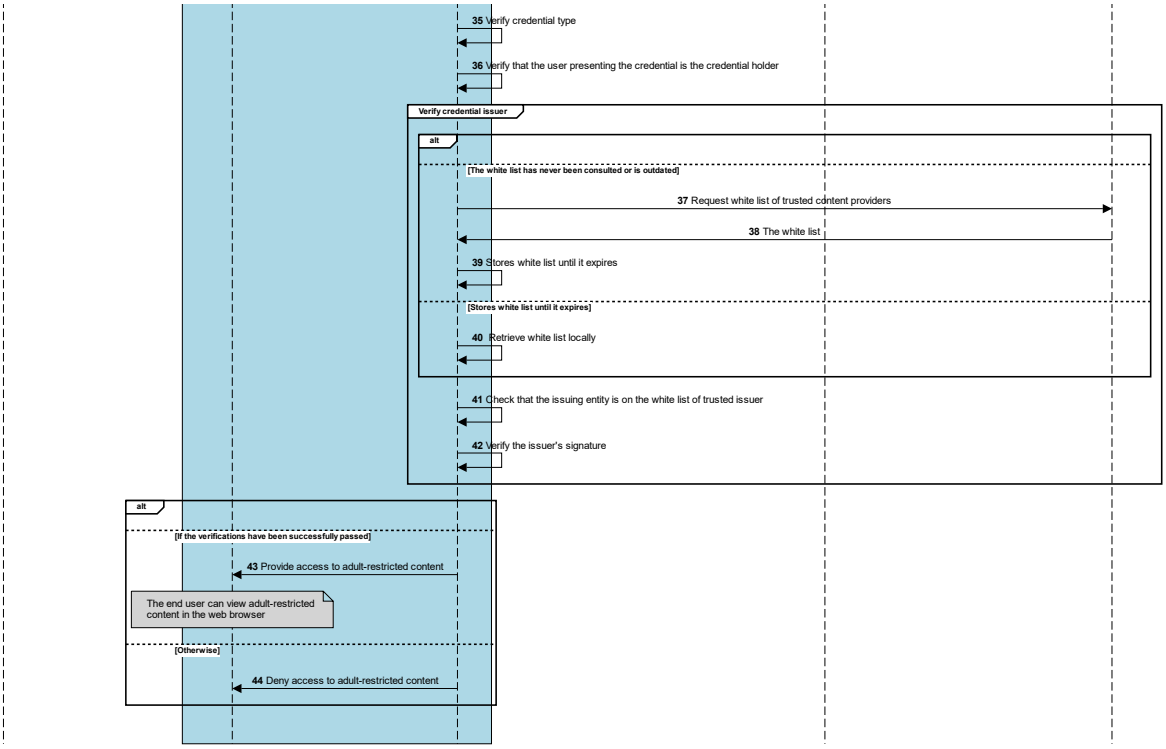


Figure 9. OID4VP flow

The flow by which the content provider verifies whether the end user is of legal age to access to the adult-restricted content consists of the following steps:

1. The end user requests access to adult-restricted content from the content provider's platform.
2. The platform displays a button to access the content next to the instructions of the following prerequisites:
   - Have the Digital Wallet [BETA] mobile application installed. The link indicating where it can be downloaded is alongside.
   - Have the credential of majority.
3. The end user clicks on the button to access the adult-restricted content.
4. The platform requests access to the content from the service.
5. The service of the content provider generates the request for evidence.
6. A QR code is generated from the request for evidence, see ISO/IEC 18004:2015.
7. The content provider service returns the request for evidence to the corresponding platform.

The content provider will display both the QR code, and the *Deep Link* of the evidence request on the platform so that the end user can access the Digital Wallet [BETA] mobile application regardless of whether it is installed on the same device or on another they want to use to view the content.

8. The Digital Wallet [BETA] mobile application obtains the request for evidence, either by scanning the QR code or by clicking on the URI of the request for evidence that will redirect the end user to the mobile application.

If the end user does not have the Digital Wallet [BETA] app installed on the mobile device:

9. They will receive an error with a message that will be unreadable to the end user.
10. The content provider will use the `nonce` field of the evidence request to control the session *timeout*, which will be set to 2 minutes. If after two minutes it does not receive the evidence, as will happen when the user has not downloaded the mobile application, the content provider will close the session.

The mobile application will be able to cache the whitelist until the `nextUpdate` property exceeds the date on which a user wants to query the list. Therefore, if there is no local whitelist of suppliers or the date is later than the one indicated in the `nextUpdate` field of the list:

11. The mobile application requests the whitelist of content providers.
12. It gets the whitelist.
13. It stores the list locally.

If there is a list and the date of the query is less than that indicated in the `nextUpdate` field:

14. It gets the whitelist.
15. It uses the `client_id` field it received in the evidence request to obtain the content provider information associated with that whitelist identifier.
16. It checks whether the content provider associated with that identifier has the `request_uri` indicated in the request for evidence in the whitelist of associated content providers.
17. If it is trusted, that is, `if` the `client_id` is in the whitelist and has the `request_uri` provided in the request associated with it, the mobile application requests the object that contains the parameters of the request for evidence by making a GET request to the URI that has been provided in the `request_uri` field of the previously received

```
GET /request.jwt/GkurKxf5T0Y-mnPFCHqWOMiZi4VS138cQO_V7PZHAdM
```

    request for evidence.
18. It obtains the object with the parameters of the evidence request.
19. The mobile application extracts the `response_uri` parameter from the object obtained in the previous step.

If the mobile application does not have a whitelist of suppliers locally or the date is higher than that indicated in the `nextUpdate` field of the list:

20. It requests the SGAD server for the whitelist of trusted content providers.
21. It gets the whitelist.
22. It stores the list locally.

If there is a list and the date of the query is less than that indicated in the `nextUpdate` field:

23. It retrieves the whitelist locally.
24. It queries the whitelist if the `response_uri` value that comes in the object of the request for evidence is in the whitelist, since it is the response URI that is used in the whitelist as an identifier of the trusted provider.

If the supplier is not on the whitelist, it is not a trusted entity and no answer is given, so that,

25. After the two minutes of *timeout*, the supplier will close the session.

If, on the other hand, the supplier is a trusted entity,

26. The mobile app will check if it has the requested credential.

If the credential is not available, the supplier will not receive any response so that it cannot deduce information about the end user, after the two minutes set in the supplier for the timeout,

27. The supplier will close the session.

Otherwise,

28. The end user will be asked for consent to share the age of majority credential with the content provider.

If the user does not want to share the credential,

29. They refuse consent.
30. After two minutes, the supplier will close the session.

If the user wants to share the credential,

31. They consent to share credential.
32. The mobile application makes a POST request to the service provider with the evidence.

Once the supplier obtains the evidence, the verification process detailed in the next section begins. Finally, if all validations are successful, the provider's service will give access to the adult-restricted content to the platform where the end user will be able to view it, otherwise, they will not.

5 VERIFICATION OF THE CREDENTIAL OF AGE OF MAJORITY

Once the identity provider has obtained the verifiable presentation, it must perform the following validations on it to give the end user access to the adult-restricted content:

1. Both the session and the associated authorization request will be retrieved locally from the `nonce` field received in the evidence. It must be checked to ensure that it has not been previously used.
2. It must be verified that both the evidence and the verifiable presentations it contains have not expired, evaluating the value given by the `exp` field included in the different JWT tokens. Likewise, it must be verified that these have been generated for the expected content provider, evaluating in this case that the `aud` field matches the provider's unique identifier.
3. It must be verified that both the evidence and the verifiable presentation it contains have been signed by the owner of the credential that is presented in it; that is, by the private key associated with the DID that is in the *credentialSubject.id* field of the credential.

4. It must be verified that each of the presentations and credentials included in the evidence comply with the requirements indicated in the `presentation_definition` field of the authorization request. Taking as reference the response-request model defined in this document, the following verifications must be carried out:
    - All required verifiable presentations must be included in the evidence. Specifically, for this use case, it must be ensured that a single verifiable presentation in JWT format has been received. To do this, the `vp_token` field of the evidence must be consulted.
    - In the set of verifiable presentations included in the evidence, all the required verifiable credentials and each one's location must be included. In other words, for each element (id) of the *input_descriptor* field of the authorization request, there must be an associated element (id) in the *vp_token.presentation_submission.descriptor_map* field of the evidence. For this use case, a single presentation with the credential of majority is expected.
    - The *presentation_definition.id field* must match the *presentation_submission.definition_id* field of the evidence
    - The presentation must come in JWT format and the signature algorithm, given by the presentation_definition.format field, must be RS512
    - For each 'j' credential, of a given 'i' presentation, required by the *presentation_definition[i].input_descriptors[j].id* field in the authorization request, an analogous credential must exist in the presentation_submission[i].descriptor_map[j].id field
    - For each `path` constraint defined for a credential 'j', of a given presentation 'i', the existence of that field in the specified credential and location must be validated. To do so, the definition in the field of the *presentation_definition[i] .input_descriptors[j].constraints.fields.path* field must be consulted.
5. It must be ensured that none of the required verifiable credentials have expired. To do so, the *validUntil* field of the credential must be consulted, if there is one.
6. It must be verified that the value of the credential type field is K ("type": ["VerifiableCredential", "K"]), this attribute is the one that proves that the credential holder can access the requested content.
7. The verifier must extract the public key from the issuer's DID and validate the signature on each credential using that key. Next, the verifier must check that the DID associated with the issuer is on the issuer whitelist. To do so, it must keep an updated local record of said list. The local list will preferably be used to perform the checks, unless it has not yet been downloaded or the maximum time for which the list is updated has been exceeded, in which case the verifier will have to download it from the repository enabled for this purpose.

6 ANNEX I – REFERENCES

[DID-Core] Sporny, M., Guy, A., Sabadello, M., and D. Reed, "Decentralized Identifiers (DIDs) v1.0", 19 July 2022, <https://www.w3.org/TR/did-core/>.

[DID-Key] Sporny, M., Zagidulin D., Longley D., Steele O., "The did:key Method v0.7", 02 September 2022, <https://w3c-ccg.github.io/did-method-key/>.

[OpenID4VP] Terbu, O., Lodderstedt, T., Yasuda, K., and T. Looker, "OpenID for Verifiable Presentations", 29 November 2022, <https://openid.net/specs/openid-4-verifiable-presentations-1_0.html>.

[DIF.PresentationExchange] Buchner, D., Zundel, B., Riedel, M., and K. H. Duffy, "Presentation Exchange 2.0.0", <https://identity.foundation/presentation-exchange/spec/v2.0.0/>.

[DIF.ClaimFormatRegistry] Buchner, D., Zundel, B., Riedel, M., and K. H. Duffy, "Claim Format Registry", <https://identity.foundation/claim-format-registry/#registry>.

[RFC2397] L. Masinter, "The "data" URL scheme", <https://www.rfc-editor.org/rfc/rfc2397>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://datatracker.ietf.org/doc/html/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://datatracker.ietf.org/doc/html/rfc7518 >.

[Figura-OpenID4VP] Annex II – OpenID4VP.svg