Workgroup:	eKYC-IDA	
Internet-Draft:	openid-connect-4-identity-assurance-1_0-10	
Published:	5 May 2020	
Intended Status:	Standards Track	
Authors:	T. Lodderstedt	D. Fett
	yes.com	*yes.com*

OpenID Connect for Identity Assurance 1.0

Abstract

This specification defines an extension of OpenID Connect for providing Relying Parties with verified Claims about End-Users. This extension is intended to be used to verify the identity of a natural person in compliance with a certain law.

Table of Contents

1. Introduction
    1.1. Terminology
2. Scope and Requirements
3. Claims
    3.1. Additional Claims about End-Users
    3.2. txn Claim
4. verified_claims Element
    4.1. verification Element
        4.1.1. Evidence
    4.2. claims Element
    4.3. verified_claims Delivery
5. Requesting Verified Claims
    5.1. Requesting End-User Claims
        5.1.1. Error Handling
    5.2. Requesting Verification Data
        5.2.1. Error Handling
    5.3. Defining further constraints on Verification Data
        5.3.1. value/values
        5.3.2. max_age
6. Examples
    6.1. id_document
    6.2. id_document + utility bill
    6.3. Notified eID system (eIDAS)
    6.4. Multiple Verified Claims
    6.5. Verified Claims in UserInfo Response
        6.5.1. Request
        6.5.2. UserInfo Response
    6.6. Verified Claims in ID Tokens
        6.6.1. Request

6.6.2. ID Token

6.7. OP-attested and External Claims

6.8. Self-Issued OpenID Connect Provider and External Claims

7. OP Metadata

8. Transaction-specific Purpose

9. Privacy Consideration

10. Security Considerations

11. Predefined Values

12. Normative References

13. Informative References

Appendix A. IANA Considerations

A.1. JSON Web Token Claims Registration

A.1.1. Registry Contents

Appendix B. Acknowledgements

Appendix C. Notices

Appendix D. Document History

Authors' Addresses

1. Introduction

This specification defines an extension to OpenID Connect [OpenID] to address the use case of strong identity verification of a natural person in accordance with certain laws. Examples include Anti Money Laundering Laws, Telecommunication Acts, Anti Terror Laws, and regulations on trust services, such as eIDAS [eIDAS].

In such use cases, the Relying Parties (RPs) need to know the assurance level of the Claims about the End-User attested by the OpenID Connect Providers (OPs) or any other trusted source, along with evidence related to the identity verification process.

The `acr` Claim, as defined in Section 2 of the OpenID Connect specification [OpenID], is suited to attest information about the authentication performed in an OpenID Connect transaction. But identity assurance requires a different representation for the following reason: authentication is an aspect of an OpenID Connect transaction while identity assurance is a property of a certain Claim or a group of Claims and several of them will typically be conveyed to the RP as the result of an OpenID Connect transaction.

For example, the assurance an OP typically will be able to attest for an e-mail address will be "self-asserted" or "verified by opt-in or similar mechanism". The family name of a user, in contrast, might have been verified in accordance with the respective Anti Money Laundering Law by showing an ID Card to a trained employee of the OP operator.

Identity assurance therefore requires a way to convey assurance data along with and coupled to the respective Claims about the End-User. This specification proposes a suitable representation and mechanisms the RP will utilize to request verified claims about an End-User along with identity assurance data and for the OP to represent these verified Claims and accompanying identity assurance data.

1.1. Terminology

This section defines some terms relevant to the topic covered in this document, inspired by NIST SP 800-63A [NIST-SP-800-63a].

- Identity Proofing - process in which a user provides evidence to an OP or claim provider reliably identifying themselves, thereby allowing the OP or claim provider to assert that identification at a useful identity assurance level.
- Identify Verification - process conducted by the OP or a claim provider to verify the user's identity.
- Identity Assurance - process in which the OP or a claim provider attests identity data of a certain user with a certain assurance towards an RP, typically expressed by way of an assurance level. Depending on legal requirements, the OP may also be required to provide evidence of the identity verification process to the RP.
- Verified Claims - Claims about an End-User, typically a natural person, whose binding to a particular user account was verified in the course of an identity verification process.

2. Scope and Requirements

The scope of the extension is to define a mechanism to assert verified Claims, in general, and to introduce new Claims about the End-User required in the identity assurance space; one example would be the place of birth.

The RP will be able to request the minimal data set it needs (data minimization) and to express requirements regarding this data and the evidence and the identity verification processes employed by the OP.

This extension will be usable by OPs operating under a certain regulation related to identity assurance, such as eIDAS, as well as other OPs operating without such a regulation.

It is assumed that OPs operating under a suitable regulation can attest identity data without the need to provide further evidence since they are approved to operate according to well-defined rules with clearly defined liability. For example in the case of eIDAS, the peer review ensures eIDAS compliance and the respective member state assumes the liability for the identities asserted by its notified eID systems.

Every other OP not operating under such well-defined conditions may be required to provide the RP data about the identity verification process along with identity evidence to allow the RP to conduct their own risk assessment and to map the data obtained from the OP to other laws. For example, if an OP verifies and maintains identity data in accordance with an Anti Money Laundering Law, it shall be possible for a RP to use the respective identity in a different regulatory context, such as eHealth or the beforementioned eIDAS.

The basic idea of this specification is that the OP provides all identity data along with metadata about the identity verification process at the OP. It is the responsibility of the RP to assess this data and map it into its own legal context.

From a technical perspective, this means this specification allows the OP to attest verified Claims along with information about the respective trust framework (and assurance level), but also supports the externalization of information about the identity verification process.

The representation defined in this specification can be used to provide RPs with verified Claims about the End-User via any appropriate channel. In the context of OpenID Connnect, verified Claims can be attested in ID Tokens or as part of the UserInfo response. It is also possible to utilize the format described here in OAuth Access Tokens or Token Introspection responses (see [RFC7662] and [I-D.ietf-oauth-jwt-introspection-response]) to provide resource servers with verified Claims.

This extension is intended to be truly international and support identity assurance for different jurisdictions and across jurisdictions. The extension is therefore extensible to support various trust frameworks, verification methods, and identity evidence.

In order to give implementors as much flexibility as possible, this extension can be used in conjunction with existing OpenID Connect Claims and other extensions within the same OpenID Connect assertion (e.g., ID Token or UserInfo response) utilized to convey Claims about End-Users.

For example, OpenID Connect [OpenID] defines Claims for representing family name and given name of a user without a verification status. Those Claims can be used in the same OpenID Connect assertion beside verified Claims represented according to this extension.

In the same way, existing Claims to inform the RP of the verification status of the `phone_number` and `email` Claims can be used together with this extension.

Even for asserting verified Claims, this extension utilizes existing OpenID Connect Claims if possible and reasonable. The extension will, however, ensure RPs cannot (accidentally) interpret unverified Claims as verified Claims.

3. Claims

3.1. Additional Claims about End-Users

In order to fulfill the requirements of some jurisdictions on identity assurance, this specification defines the following Claims for conveying user data in addition to the Claims defined in the OpenID Connect specification [OpenID]:

Claim	Type	Description
`place_of_birth`	JSON object	End-User's place of birth. The value of this member is a JSON structure containing some or all of the following members:
		`country`: String representing country in [ISO3166-1] Alpha-2 (e.g., DE) or [ISO3166-3] syntax.
		`region`: String representing state, province, prefecture, or region component. This field might be required in some jurisdictions.
		`locality`: String representing city or locality component.
`nationalities`	array	End-User's nationalities in ICAO 2-letter codes [ICAO-Doc9303], e.g. "US" or "DE". 3-letter codes MAY be used when there is no corresponding ISO 2-letter code, such as "EUE".
`birth_family_name`	string	End-User's family name when he or she is born, or at least from the time he or she is a child. This term can be used by a person who changes the family name later in life for any reason.

Claim	Type	Description
`birth_given_name`	string	End-User's given name when he or she is born, or at least from the time he or she is a child. This term can be used by a person who changes the given name later in life for any reason.
`birth_middle_name`	string	End-User's middle name when he or she is born, or at least from the time he or she is a child. This term can be used by a person who changes the middle name later in life for any reason.
`salutation`	string	End-User's salutation, e.g. "Mr."
`title`	string	End-User's title, e.g. "Dr."

Table 1

3.2. txn Claim

Strong identity verification typically requires the participants to keep an audit trail of the whole process.

The `txn` Claim as defined in [RFC8417] is used in the context of this extension to build audit trails across the parties involved in an OpenID Connect transaction.

If the OP issues a `txn`, it MUST maintain a corresponding audit trail, which at least consists of the following details:

- the transaction ID,
- the authentication method employed, and
- the transaction type (e.g. scope values).

This transaction data MUST be stored as long as it is required to store transaction data for auditing purposes by the respective regulation.

The RP requests this Claim like any other Claim via the `claims` parameter or as part of a default Claim set identified by a scope value.

The `txn` value MUST allow an RP to obtain these transaction details if needed.

Note: The mechanism to obtain the transaction details from the OP and their format is out of scope of this specification.

4. verified_claims Element

This specification defines a generic mechanism to add verified claims to JSON-based assertions. The basic idea is to use a container element, called `verified_claims` to provide the RP with a set of Claims along with the respective metadata and verification evidence related to the verification of these claims. This way RPs cannot mix up verified and unverified Claims and accidentally process unverified Claims as verified Claims.

A following example

```
{
   "verified_claims":{
      "verification":{
         "trust_framework":"ial_example_gold"
      },
      "claims":{
         "given_name":"Max",
         "family_name":"Meier"
      }
   }
}
```

would attest to the RP that the OP has verified the claims provided (`given_name` and `family_name`) according to an example trust framework `ial_example_gold`.

The normative definition is given in the following.

`verified_claims`: Object or array containing one or more verified claims objects.

A single `verified_claims` object consists of the following sub-elements:

- `verification`: REQUIRED. Object that contains data about the verification process.
- `claims`: REQUIRED. Object that is the container for the verified Claims about the End-User.

Note: Implementations MUST ignore any sub-element not defined in this specification or extensions of this specification.

Note: If not stated otherwise, every sub-element in `verified_claims` is defined as optional. Extensions of this specification, including trust framework definitions, can define further constraints on the data structure.

A machine-readable syntax definition of `verified_claims` is given as JSON schema in [verified_claims.json]. It can be used to automatically validate JSON documents containing a `verified_claims` element.

4.1. verification Element

This element contains the information about the process conducted to verify a person's identity and bind the respective person data to a user account.

The `verification` element consists of the following elements:

`trust_framework`: REQUIRED. String determining the trust framework governing the identity verification process and the identity assurance level of the OP.

An example value is `eidas_ial_high`, which denotes a notified eID system under eIDAS [eIDAS] providing identity assurance at level of assurance "High".

For information on predefined trust framework values see Section 11.

RPs SHOULD ignore `verified_claims` claims containing a trust framework ID they don't understand.

The `trust_framework` value determines what further data is provided to the RP in the `verification` element. A notified eID system under eIDAS, for example, would not need to provide any further data whereas an OP not governed by eIDAS would need to provide verification evidence in order to allow the RP to fulfill its legal obligations. An example of the latter is an OP acting under the German Anti-Money Laundering Law (`de_aml`).

`time`: Time stamp in ISO 8601:2004 [ISO8601-2004] `YYYY-MM-DDThh:mm[:ss]TZD` format representing the date and time when the identity verification process took place. This time might deviate from (a potentially also present) `id_document/time` element since the latter represents the time when a certain evidence was checked whereas this element represents the time when the process was completed. Moreover, the overall verification process and evidence verification can be conducted by different parties (see `id_document/verifier`). Presence of this element might be required for certain trust frameworks.

`verification_process`: Unique reference to the identity verification process as performed by the OP. Used for backtracing in case of disputes or audits. Presence of this element might be required for certain trust frameworks.

Note: While `verification_process` refers to the identity verification process at the OP, the `txn` claim refers to a particular OpenID Connect transaction in which the OP attested the user's verified identity data towards an RP.

`evidence`: JSON array containing information about the evidence the OP used to verify the user's identity as separate JSON objects. Every object contains the property `type` which determines the type of the evidence. The RP uses this information to process the `evidence` property appropriately.

Important: Implementations MUST ignore any sub-element not defined in this specification or extensions of this specification.

4.1.1. Evidence

The following types of evidence are defined:

- `id_document`: Verification based on any kind of government issued identity document.
- `utility_bill`: Verification based on a utility bill.
- `qes`: Verification based on an eIDAS Qualified Electronic Signature.

4.1.1.1. id_document

The following elements are contained in an `id_document` evidence sub-element.

`type`: REQUIRED. Value MUST be set to "id_document".

`method`: The method used to verify the ID document. For information on predefined verification method values see Section 11.

`verifier`: JSON object denoting the legal entity that performed the identity verification on behalf of the OP. This object SHOULD only be included if the OP did not perform the identity verification itself. This object consists of the following properties:

- `organization`: String denoting the organization which performed the verification on behalf of the OP.
- `txn`: Identifier referring to the identity verification transaction. This transaction identifier can be resolved into transaction details during an audit.

`time`: Time stamp in ISO 8601:2004 [ISO8601-2004] `YYYY-MM-DDThh:mm[:ss]TZD` format representing the date when this ID document was verified.

`document`: JSON object representing the ID document used to perform the identity verification. It consists of the following properties:

- `type`: REQUIRED. String denoting the type of the ID document. For information on predefined identity document values see Section 11. The OP MAY use other than the predefined values in which case the RPs will either be unable to process the assertion, just store this value for audit purposes, or apply bespoken business logic to it.
- `number`: String representing the number of the identity document.
- `issuer`: JSON object containing information about the issuer of this identity document. This object consists of the following properties:
  - `name`: Designation of the issuer of the identity document.
  - `country`: String denoting the country or organization that issued the document as ICAO 2-letter code [ICAO-Doc9303], e.g. "JP". ICAO 3-letter codes MAY be used when there is no corresponding ISO 2-letter code, such as "UNO".
- `date_of_issuance`: The date the document was issued as ISO 8601:2004 `YYYY-MM-DD` format.
- `date_of_expiry`: The date the document will expire as ISO 8601:2004 `YYYY-MM-DD` format.

4.1.1.2. utility_bill

The following elements are contained in a `utility_bill` evidence sub-element.

`type`: REQUIRED. Value MUST be set to "utility_bill".

`provider`: JSON object identifying the respective provider that issued the bill. The object consists of the following properties:

- `name`: String designating the provider.
- All elements of the OpenID Connect `address` Claim ([OpenID])

`date`: String in ISO 8601:2004 `YYYY-MM-DD` format containing the date when this bill was issued.

4.1.1.3. qes

The following elements are contained in a `qes` evidence sub-element.

`type`: REQUIRED. Value MUST be set to "qes".

`issuer`: String denoting the certification authority that issued the signer's certificate.

`serial_number`: String containing the serial number of the certificate used to sign.

`created_at`: The time the signature was created as ISO 8601:2004 `YYYY-MM-DDThh:mm[:ss]TZD` format.

4.2. claims Element

The `claims` element contains the claims about the End-User which were verified by the process and according to the policies determined by the corresponding `verification` element.

The `claims` element MAY contain one or more of the following Claims as defined in Section 5.1 of the OpenID Connect specification [OpenID]

- `name`

- `given_name`
- `middle_name`
- `family_name`
- `birthdate`
- `address`

and the claims defined in Section 3.1.

The `claims` element MAY also contain other claims provided the value of the respective claim was verified in the verification process represented by the sibling `verification` element.

Claim names MAY be annotated with language tags as specified in Section 5.2 of the OpenID Connect specification [OpenID].

4.3. verified_claims Delivery

OPs can deliver `verified_claims` in various ways.

A `verified_claims` element can be added to a OpenID Connect UserInfo response or an ID Token.

OAuth Authorization Servers can add `verified_claims` to access tokens in JWT format or Token Introspection responses, either in plain JSON or JWT-protected format.

An OP or AS MAY also include `verified_claims` in the beforementioned assertions as aggregated or distributed claims (see Section 5.6.2 of the OpenID Connect specification [OpenID]).

In this case, every assertion provided by the external claims source MUST contain

- an `iss` claim identifying the claims source,
- a `sub` claim identifying the user in the context of the claim source,
- a `verified_claims` element containing one or more verified_claims objects.

Claims sources SHOULD sign the assertions containing `verified_claims` in order to protect integrity and authenticity. The way a RP determines the key material used for validation of the signed assertions is out of scope. The recommended way is to determine the claims source's public keys by obtaining its JSON Web Key Set via the `jwks_uri` metadata value read from its `openid-configuration` metadata document. This document can be discovered using the `iss` claim of the particular JWT.

The following are examples of assertions including verified claims as aggregated claims

```
{
  "iss": "https://server.example.com",
  "sub": "248289761001",
  "email": "janedoe@example.com",
  "email_verified": true,
  "_claim_names": {
    "verified_claims": "src1"
  },
  "_claim_sources": {
    "src1": {
      "JWT": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwcz
      ovL3NlcnZlci5vdGhlcm9wLmNvbSIsInN1YiI6ImU4MTQ4NjAzLTg5MzQtNDI0N
      S04MjViLWMxMDhiOGI2Yjk0NSIsInZlcmlmaWVkX2NsYWltcyI6eyJ2ZXJpZmlj
      YXRpb24iOnsidHJ1c3RfZnJhbWV3b3JrIjoiaWFsX2V4YW1wbGVfZ29sZCJ9LCJ
      jbGFpbXMiOnsiZ2l2ZW5fbmFtZSI6Ik1heCIsImZhbWlseV9uYW1lIjoiTWVpZX
      IiLCJiaXJ0aGRhdGUiOiIxOTU2LTAxLTI4In19fQ.FArlPUtUVn95HCExePlWJQ
      6ctVfVpQyeSbe3xkH9MH1QJjnk5GVbBW0qe1b7R3lE-8iVv__0mhRTUI5lcFhLj
      oGjDS8zgWSarVsEEjwBK7WD3r9cEw6ZAhfEkhHL9eqAaED2rhhDbHD5dZWXkJCu
      XIcn65g6rryiBanxlXK0ZmcK4fD9HV9MFduk0LRG_p4yocMaFvVkqawat5NV9QQ
      3ij7UBr3G7A4FojcKEkoJKScdGoozir8m5XD83Sn45_79nCcgWSnCX2QTukL8Ny
      wIItu_K48cjHiAGXXSzydDm_ccGCe0sY-Ai2-iFFuQo2PtfuK2SqPPmAZJxEFrF
      oLY4g"
    }
  }
}
```


and distributed claims.


```
{
  "iss": "https://server.example.com",
  "sub": "248289761001",
  "email": "janedoe@example.com",
  "email_verified": true,
  "_claim_names": {
    "verified_claims": "src1"
  },
  "_claim_sources": {
    "src1": {
      "endpoint": "https://server.yetanotherop.com/claim_source",
      "access_token": "ksj3n283dkeafb76cdef"
    }
  }
}
```


An external assertion MAY include (or refer to) multiple `verified_claims` provided by different external claims sources. To support this use case, this specification extends the syntax as defined in Section 5.6.2 of the OpenID Connect specification [OpenID]) to also allow references to multiple claims sources as string array.

The following example shows an ID token containing `verified_claims` from two different external claims sources, one as aggregated and the other as distributed claims.

```
{
  "iss": "https://server.example.com",
  "sub": "248289761001",
  "email": "janedoe@example.com",
  "email_verified": true,
  "_claim_names": {
    "verified_claims": [
      "src1",
      "src2"
    ]
  },
  "_claim_sources": {
    "src1": {
      "JWT": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwcz
      ovL3NlcnZlci5vdGhlcm9wLmNvbSIsInN1YiI6ImU4MTQ4NjAzLTg5MzQtNDI0N
      S04MjViLWMxMDhiOGI2Yjk0NSIsInZlcmlmaWVkX2NsYWltcyI6eyJ2ZXJpZmlj
      YXRpb24iOnsidHJ1c3RfZnJhbWV3b3JrIjoiaWFsX2V4YW1wbGVfZ29sZCJ9LCJ
      jbGFpbXMiOnsiZ2l2ZW5fbmFtZSI6Ik1heCIsImZhbWlseV9uYW1lIjoiTWVpZX
      IiLCJiaXJ0aGRhdGUiOiIxOTU2LTAxLTI4In19fQ.FArlPUtUVn95HCExePlWJQ
      6ctVfVpQyeSbe3xkH9MH1QJjnk5GVbBW0qe1b7R3lE-8iVv__0mhRTUI5lcFhLj
      oGjDS8zgWSarVsEEjwBK7WD3r9cEw6ZAhfEkhHL9eqAaED2rhhDbHD5dZWXkJCu
      XIcn65g6rryiBanxlXK0ZmcK4fD9HV9MFduk0LRG_p4yocMaFvVkqawat5NV9QQ
      3ij7UBr3G7A4FojcKEkoJKScdGoozir8m5XD83Sn45_79nCcgWSnCX2QTukL8Ny
      wIItu_K48cjHiAGXXSzydDm_ccGCe0sY-Ai2-iFFuQo2PtfuK2SqPPmAZJxEFrF
      oLY4g"
    },
    "src2": {
      "endpoint": "https://server.yetanotherop.com/claim_source",
      "access_token": "ksj3n283dkeafb76cdef"
    }
  }
}
```


The OP MAY combine aggregated and distributed claims with `verified_claims` attested by itself (see Section 6.7).

If `verified_claims` elements are contained in multiple places of a response, e.g. in the ID token and a embedded aggregated claim, the RP MUST preserve the claims source as context of the particular `verified_claims` element.

Note: any assertion provided by an OP or AS including aggregated or distributed claims MAY contain multiple instances of the same End-User claim. It is up to the RP to decide how to process those different instances.

5. Requesting Verified Claims

5.1. Requesting End-User Claims

Verified Claims can be requested on the level of individual Claims about the End-User by utilizing the `claims` parameter as defined in Section 5.5 of the OpenID Connect specification [OpenID].

Note: A machine-readable definition of the syntax to be used to request `verified_claims` is given as JSON schema in [verified_claims_request.json]. It can be used to automatically validate `claims` request parameters.

To request verified claims, the `verified_claims` element is added to the `userinfo` or the `id_token` element of the `claims` parameter.

Since `verified_claims` contains the effective Claims about the End-User in a nested `claims` element, the syntax is extended to include expressions on nested elements as follows. The `verified_claims` element includes a `claims` element, which in turn includes the desired Claims as keys with a `null` value. An example is shown in the following:

```
{
   "userinfo":{
      "verified_claims":{
         "verification": {
            "trust_framework": null
         },
         "claims":{
            "given_name":null,
            "family_name":null,
            "birthdate":null
         }
      }
   }
}
```

Use of the `claims` parameter allows the RP to exactly select the Claims about the End-User needed for its use case. This extension therefore allows RPs to fulfill the requirement for data minimization.

RPs MAY indicate that a certain Claim is essential to the successful completion of the user journey by utilizing the `essential` field as defined in Section 5.5.1 of the OpenID Connect specification [OpenID]. The following example designates both given name as well as family name as being essential.

```
{
   "userinfo":{
      "verified_claims":{
         "verification": {
            "trust_framework": null
         },
         "claims":{
            "given_name":{"essential": true},
            "family_name":{"essential": true},
            "birthdate":null
         }
      }
   }
}
```

This specification introduces the additional field `purpose` to allow an RP to state the purpose for the transfer of a certain End-User Claim it is asking for. The field `purpose` can be a member value of each individually requested Claim, but a Claim cannot have more than one associated purpose.

`purpose`: OPTIONAL. String describing the purpose for obtaining a certain End-User Claim from the OP. The purpose MUST NOT be shorter than 3 characters or longer than 300 characters. If this rule is violated, the authentication request MUST fail and the OP return an error `invalid_request` to the RP. The OP MUST display this purpose in the respective user consent screen(s) in order to inform the user about the designated use of the data to be transferred or the authorization to be approved. If the parameter `purpose` is not present in the request, the OP MAY display a value that was pre-configured for the respective RP. For details on UI localization, see Section 8.

Example:

```
{
   "userinfo":{
      "verified_claims":{
         "verification": {
            "trust_framework": null
         },
         "claims":{
            "given_name":{
               "essential":true,
               "purpose":"To make communication look more personal"
            },
            "family_name":{
               "essential":true
            },
            "birthdate":{
               "purpose":"To send you best wishes on your birthday"
            }
         }
      }
   }
}
```

5.1.1. Error Handling

If the `claims` sub-element is empty, the OP MUST abort the transaction with an `invalid_request` error.

Claims unknown to the OP or not available as verified claims MUST be ignored and omitted from the response. If the resulting `claims` sub-element is empty, the OP MUST omit the `verified_claims` element.

5.2. Requesting Verification Data

RPs request verification data in the same way they request claims about the end-user. The syntax is based on the rules given in Section 5.1 and extends them for navigation into the structure of the `verification` element.

Elements within `verification` are requested by adding the respective element as shown in the following example:

```
{
  "userinfo": {
    "verified_claims": {
      "verification": {
        "trust_framework": null,
        "time": null
      },
      "claims": {
        "given_name": null,
        "family_name": null,
        "birthdate": null
      }
    }
  }
}
```

It requests the trust framework the OP complies with and the date of the verification of the user claims.

The RP MUST explicitly request any data it wants the OP to add to the `verification` element.

Therefore, the RP MUST set fields one step deeper into the structure if it wants to obtain evidence. One or more entries in the `evidence` array are used as filter criteria and templates for all entries in the result array. The following examples shows a request asking for evidence of type `id_document`.

```
{
  "userinfo": {
    "verified_claims": {
      "verification": {
        "trust_framework": null,
        "time": null,
        "evidence": [
          {
            "type": {
              "value": "id_document"
            },
            "method": null,
            "document": {
              "type": null
            }
          }
        ]
      },
      "claims": {
        "given_name": null,
        "family_name": null,
        "birthdate": null
      }
    }
  }
}
```

The example also requests the OP to add the respective `method` and the `document` elements (including data about the document type) for every evidence to the resulting `verified_claims` claim.

A single entry in the `evidence` array represents a filter over elements of a certain evidence type. The RP therefore MUST specify this type by including the `type` field including a suitable `value` sub-element value. The `values` sub-element MUST NOT be used for the `evidence/type` field.

If multiple entries are present in `evidence`, these filters are linked by a logical OR.

The RP MAY also request certain data within the `document` element to be present. This again follows the syntax rules used above:

```
{
  "userinfo": {
    "verified_claims": {
      "verification": {
        "trust_framework": null,
        "time": null,
        "evidence": [
          {
            "type": {
              "value": "id_document"
            },
            "method": null,
            "document": {
              "type": null,
              "issuer":  {
                "country": null,
                "name": null
              },
              "number": null,
              "date_of_issuance": null
            }
          }
        ]
      },
      "claims": {
        "given_name": null,
        "family_name": null,
        "birthdate": null
      }
    }
  }
}
```

5.2.1. Error Handling

The OP has the discretion to decide whether the requested verification data is to be provided to the RP. An OP MUST NOT return an error in case it cannot return a requested verification data, even if it was marked as essential, regardless of the data being unavailable or the End-User not authorizing its release.

5.3. Defining further constraints on Verification Data

5.3.1. value/values

The RP MAY limit the possible values of the elements `trust_framework`, `evidence/method`, and `evidence/document/type` by utilizing the `value` or `values` fields and the element `evidence/type` by utilizing the `value` field.

Note: Examples on the usage of a restriction on `evidence/type` were given in the previous section.

The following example shows that the RP wants to obtain an attestation based on the German Anti Money Laundering Law (trust framework `de_aml`) and limited to users who were identified in a bank branch in person (physical in person proofing - method `pipp`) using either an `idcard` or a `passport`.

```
{
  "userinfo": {
    "verified_claims": {
      "verification": {
        "trust_framework": {
          "value": "de_aml"
        },
        "evidence": [
          {
            "type": {
              "value": "id_document"
            },
            "method": {
              "value": "pipp"
            },
            "document": {
              "type": {
                "values": [
                  "idcard",
                  "passport"
                ]
              }
            }
          }
        ]
      },
      "claims": {
        "given_name": null,
        "family_name": null,
        "birthdate": null
      }
    }
  }
}
```

In case the RP limits the possible values of any of the aforementioned four elements and the OP does not understand/support some or all of them (i.e. their values are not listed under its OP metadata) or they are not applicable/fulfillable for a certain user, the OP MUST NOT return an error, but instead not deliver at all the `verified_claims` claim.

The OP MUST NOT ignore some or all of the query restrictions on possible values and deliver available verified/verification data that does not match these constraints.

5.3.2. max_age

The RP MAY also express a requirement regarding the age of certain data, like the time elapsed since the issuance/expiry of certain evidence types or since the verification process asserted in the `verification` element took place. Section 5.5.1 of the OpenID Connect specification [OpenID] defines a query syntax that allows for new special query members to be defined. This specification introduces a new such member `max_age`, which is applicable to the possible values of any elements containing dates or timestamps (e.g. `time`, `date_of_issuance` and `date_of_expiry` elements of evidence of type `id_document` or element `date` of evidence of type `utility_bill`).

`max_age`: OPTIONAL. JSON number value only applicable to Claims that contain dates or timestamps. It defines the maximum time (in seconds) to be allowed to elapse since the value of the date/timestamp up to the point in time of the request. The OP should make the calculation of elapsed time starting from the last valid second of the date value.

The following is an example of a request for Claims where the verification process of the data is not allowed to be older than 63113852 seconds:

```
{
  "userinfo": {
    "verified_claims": {
      "verification": {
        "trust_framework": {
          "value": "jp_aml"
        },
        "time": {
          "max_age": 63113852
        }
      },
      "claims": {
        "given_name": null,
        "family_name": null,
        "birthdate": null
      }
    }
  }
}
```

The OP SHOULD try to fulfill this requirement. If the verification data of the user is older than the requested `max_age`, the OP MAY attempt to refresh the user's verification by sending her through an online identity verification process, e.g. by utilizing an electronic ID card or a video identification approach. If the OP is unable to fulfill the `max_age` constraint it MUST NOT deliver the `verified_claims` claim at all.

6. Examples

The following sections show examples of responses containing `verified_claims`.

The first and second sections show JSON snippets of the general identity assurance case, where the RP is provided with verification evidence for different verification methods along with the actual Claims about the End-User.

The third section illustrates how the contents of this object could look like in case of a notified eID system under eIDAS, where the OP does not need to provide evidence of the identity verification process to the RP.

Subsequent sections contain examples for using the `verified_claims` Claim on different channels and in combination with other (unverified) Claims.

6.1. id_document

```
{
   "verified_claims":{
      "verification":{
         "trust_framework":"de_aml",
         "time":"2012-04-23T18:25Z",
         "verification_process":"f24c6f-6d3f-4ec5-973e-b0d8506f3bc7",
         "evidence":[
            {
               "type":"id_document",
               "method":"pipp",
               "time": "2012-04-22T11:30Z",
               "document":{
                  "type":"idcard",
                  "issuer":{
                     "name":"Stadt Augsburg",
                     "country":"DE"
                  },
                  "number":"53554554",
                  "date_of_issuance":"2010-03-23",
                  "date_of_expiry":"2020-03-22"
               }
            }
         ]
      },
      "claims":{
         "given_name":"Max",
         "family_name":"Meier",
         "birthdate":"1956-01-28",
         "place_of_birth":{
            "country":"DE",
            "locality":"Musterstadt"
         },
         "nationalities":[
            "DE"
         ],
         "address":{
            "locality":"Maxstadt",
            "postal_code":"12344",
            "country":"DE",
            "street_address":"An der Sandd&#252;ne 22"
         }
      }
   }
}
```


6.2. id_document + utility bill

```
{
   "verified_claims":{
      "verification":{
         "trust_framework":"de_aml",
         "time":"2012-04-23T18:25Z",
         "verification_process":"513645-e44b-4951-942c-7091cf7d891d",
         "evidence":[
            {
               "type":"id_document",
               "method":"pipp",
               "time": "2012-04-22T11:30Z",
               "document":{
                  "type":"de_erp_replacement_idcard",
                  "issuer":{
                     "name":"Stadt Augsburg",
                     "country":"DE"
                  },
                  "number":"53554554",
                  "date_of_issuance":"2010-04-23",
                  "date_of_expiry":"2020-04-22"
               }
            },
            {
               "type":"utility_bill",
               "provider":{
                  "name":"Stadtwerke Musterstadt",
                  "country":"DE",
                  "region":"Th&#252;ringen",
                  "street_address":"Energiestrasse 33"
               },
               "date":"2013-01-31"
            }
         ]
      },
      "claims":{
         "given_name":"Max",
         "family_name":"Meier",
         "birthdate":"1956-01-28",
         "place_of_birth":{
            "country":"DE",
            "locality":"Musterstadt"
         },
         "nationalities":[
            "DE"
         ],
         "address":{
            "locality":"Maxstadt",
            "postal_code":"12344",
            "country":"DE",
            "street_address":"An der Sandd&#252;ne 22"
         }
      }
   }
}
```


6.3. Notified eID system (eIDAS)

```
{
   "verified_claims":{
      "verification":{
         "trust_framework":"eidas_ial_substantial"
      },
      "claims":{
         "given_name":"Max",
         "family_name":"Meier",
         "birthdate":"1956-01-28",
         "place_of_birth":{
            "country":"DE",
            "locality":"Musterstadt"
         },
         "nationalities":[
            "DE"
         ]
      }
   }
}
```

6.4. Multiple Verified Claims

```
{
  "verified_claims":[
    {
      "verification": {
        "trust_framework": "eidas_ial_substantial"
      },
      "claims": {
        "given_name": "Max",
        "family_name": "Meier",
        "birthdate": "1956-01-28",
        "place_of_birth": {
          "country": "DE",
          "locality": "Musterstadt"
        },
        "nationalities": [
          "DE"
        ]
      }
    },
    {
      "verification":{
        "trust_framework":"de_aml",
        "time":"2012-04-23T18:25Z",
        "verification_process":"f24c6f-6d3f-4ec5-973e-b0d8506f3bc7",
        "evidence":[
          {
            "type":"id_document",
            "method":"pipp",
            "time": "2012-04-22T11:30Z",
            "document":{
              "type":"idcard"
            }
          }
        ]
      },
      "claims":{
        "address":{
          "locality":"Maxstadt",
          "postal_code":"12344",
          "country":"DE",
          "street_address":"An der Sandd&#252;ne 22"
        }
      }
    }
  ]
}
```


6.5. Verified Claims in UserInfo Response

6.5.1. Request

In this example we assume the RP uses the `scope` parameter to request the email address and, additionally, the `claims` parameter, to request verified Claims.

The scope value is: `scope=openid email`

The value of the `claims` parameter is:

```
{
   "userinfo":{
       "verified_claims":{
         "verification": {
           "trust_framework": null
         },
         "claims":{
            "given_name":null,
            "family_name":null,
            "birthdate":null
         }
      }
   }
}
```

6.5.2. UserInfo Response

The respective UserInfo response would be

```
{
  "sub": "248289761001",
  "email": "janedoe@example.com",
  "email_verified": true,
  "verified_claims": {
    "verification": {
      "trust_framework": "de_aml",
      "time": "2012-04-23T18:25:43Z",
      "verification_process": "f24c6f-6d3f-4ec5-973e-b0d8506f3bc7",
      "evidence": [
        {
          "type": "id_document",
          "method": "pipp",
          "time": "2012-04-22T11:30Z",
          "document": {
            "type": "idcard",
            "issuer": {
              "name": "Stadt Augsburg",
              "country": "DE"
            },
            "number": "53554554",
            "date_of_issuance": "2010-03-23",
            "date_of_expiry": "2020-03-22"
          }
        }
      ]
    },
    "claims": {
      "given_name": "Max",
      "family_name": "Meier",
      "birthdate": "1956-01-28"
    }
  }
}
```

6.6. Verified Claims in ID Tokens

6.6.1. Request

In this case, the RP requests verified Claims along with other Claims about the End-User in the `claims` parameter and allocates the response to the ID Token (delivered from the token endpoint in case of grant type `authorization_code`).

The `claims` parameter value is

```
{
   "id_token":{
      "email":null,
      "preferred_username":null,
      "picture":null,
      "verified_claims":{
         "verification": {
            "trust_framework": null
         },
         "claims":{
            "given_name":null,
            "family_name":null,
            "birthdate":null
         }
      }
   }
}
```

6.6.2. ID Token

The respective ID Token could be

```
{
   "iss":"https://server.example.com",
   "sub":"24400320",
   "aud":"s6BhdRkqt3",
   "nonce":"n-0S6_WzA2Mj",
   "exp":1311281970,
   "iat":1311280970,
   "auth_time":1311280969,
   "acr":"urn:mace:incommon:iap:silver",
   "email":"janedoe@example.com",
   "preferred_username":"j.doe",
   "picture":"http://example.com/janedoe/me.jpg",
   "verified_claims":{
      "verification":{
         "trust_framework":"de_aml",
         "time":"2012-04-23T18:25Z",
         "verification_process":"f24c6f-6d3f-4ec5-973e-b0d8506f3bc7",
         "evidence":[
            {
               "type":"id_document",
               "method":"pipp",
               "time": "2012-04-22T11:30Z",
               "document":{
                  "type":"idcard",
                  "issuer":{
                     "name":"Stadt Augsburg",
                     "country":"DE"
                  },
                  "number":"53554554",
                  "date_of_issuance":"2010-03-23",
                  "date_of_expiry":"2020-03-22"
               }
            }
         ]
      },
      "claims":{
         "given_name":"Max",
         "family_name":"Meier",
         "birthdate":"1956-01-28"
      }
   }
}
```


6.7. OP-attested and External Claims

This example shows how an OP can mix own claims and claims attested by external sources in a single ID token.

```
{
  "iss": "https://server.example.com",
  "sub": "248289761001",
  "email": "janedoe@example.com",
  "email_verified": true,
  "verified_claims": {
    "verification": {
      "trust_framework": "ial_example_gold"
    },
    "claims": {
      "given_name": "Max",
      "family_name": "Meier"
    }
  },
  "_claim_names": {
    "verified_claims": [
      "src1",
      "src2"
    ]
  },
  "_claim_sources": {
    "src1": {
      "JWT": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwcz
      ovL3NlcnZlci5vdGhlcm9wLmNvbSIsInN1YiI6ImU4MTQ4NjAzLTg5MzQtNDI0N
      S04MjViLWMxMDhiOGI2Yjk0NSIsInZlcmlmaWVkX2NsYWltcyI6eyJ2ZXJpZmlj
      YXRpb24iOnsidHJ1c3RfZnJhbWV3b3JrIjoiaWFsX2V4YW1wbGVfZ29sZCJ9LCJ
      jbGFpbXMiOnsiZ2l2ZW5fbmFtZSI6Ik1heCIsImZhbWlseV9uYW1lIjoiTWVpZX
      IiLCJiaXJ0aGRhdGUiOiIxOTU2LTAxLTI4In19fQ.FArlPUtUVn95HCExePlWJQ
      6ctVfVpQyeSbe3xkH9MH1QJjnk5GVbBW0qe1b7R3lE-8iVv__0mhRTUI5lcFhLj
      oGjDS8zgWSarVsEEjwBK7WD3r9cEw6ZAhfEkhHL9eqAaED2rhhDbHD5dZWXkJCu
      XIcn65g6rryiBanxlXK0ZmcK4fD9HV9MFduk0LRG_p4yocMaFvVkqawat5NV9QQ
      3ij7UBr3G7A4FojcKEkoJKScdGoozir8m5XD83Sn45_79nCcgWSnCX2QTukL8Ny
      wIItu_K48cjHiAGXXSzydDm_ccGCe0sY-Ai2-iFFuQo2PtfuK2SqPPmAZJxEFrF
      oLY4g"
    },
    "src2": {
      "endpoint": "https://server.yetanotherop.com/claim_source",
      "access_token": "ksj3n283dkeafb76cdef"
    }
  }
}
```

6.8. Self-Issued OpenID Connect Provider and External Claims

This example shows how a Self-Issued OpenID Connect Provider (SIOP) may include verified claims obtained from different external claim sources into a ID Token.

```
{
  "iss": "https://self-issued.me",
  "sub": "248289761001",
  "preferred_username": "superman445",
  "_claim_names": {
    "verified_claims": [
      "src1",
      "src2"
    ]
  },
  "_claim_sources": {
    "src1": {
      "JWT": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwcz
      ovL3NlcnZlci5vdGhlcm9wLmNvbSIsInN1YiI6ImU4MTQ4NjAzLTg5MzQtNDI0N
      S04MjViLWMxMDhiOGI2Yjk0NSIsInZlcmlmaWVkX2NsYWltcyI6eyJ2ZXJpZmlj
      YXRpb24iOnsidHJ1c3RfZnJhbWV3b3JrIjoiaWFsX2V4YW1wbGVfZ29sZCJ9LCJ
      jbGFpbXMiOnsiZ2l2ZW5fbmFtZSI6Ik1heCIsImZhbWlseV9uYW1lIjoiTWVpZX
      IiLCJiaXJ0aGRhdGUiOiIxOTU2LTAxLTI4In19fQ.FArlPUtUVn95HCExePlWJQ
      6ctVfVpQyeSbe3xkH9MH1QJjnk5GVbBW0qe1b7R3lE-8iVv__0mhRTUI5lcFhLj
      oGjDS8zgWSarVsEEjwBK7WD3r9cEw6ZAhfEkhHL9eqAaED2rhhDbHD5dZWXkJCu
      XIcn65g6rryiBanxlXK0ZmcK4fD9HV9MFduk0LRG_p4yocMaFvVkqawat5NV9QQ
      3ij7UBr3G7A4FojcKEkoJKScdGoozir8m5XD83Sn45_79nCcgWSnCX2QTukL8Ny
      wIItu_K48cjHiAGXXSzydDm_ccGCe0sY-Ai2-iFFuQo2PtfuK2SqPPmAZJxEFrF
      oLY4g"
    },
    "src2": {
      "endpoint": "https://op.mymno.com/claim_source",
      "access_token": "ksj3n283dkeafb76cdef"
    }
  }
}
```


7. OP Metadata

The OP advertises its capabilities with respect to verified Claims in its openid-configuration (see [OpenID-Discovery]) using the following new elements:

`verified_claims_supported`: Boolean value indicating support for `verified_claims`, i.e. the OpenID Connect for Identity Assurance extension.

`trust_frameworks_supported`: JSON array containing all supported trust frameworks.

`evidence_supported`: JSON array containing all types of identity evidence the OP uses.

`id_documents_supported`: JSON array containing all identity documents utilized by the OP for identity verification.

`id_documents_verification_methods_supported`: JSON array containing the ID document verification methods the OP supports as defined in Section 4.1.

`claims_in_verified_claims_supported`: JSON array containing all claims supported within `verified_claims`.

This is an example openid-configuration snippet:

```
{
...
   "verified_claims_supported":true,
   "trust_frameworks_supported":[
     "nist_800_63A_ial_2",
     "nist_800_63A_ial_3"
   ],
   "evidence_supported":[
      "id_document",
      "utility_bill",
      "qes"
   ],
   "id_documents_supported":[
       "idcard",
       "passport",
       "driving_permit"
   ],
   "id_documents_verification_methods_supported":[
       "pipp",
       "sripp",
       "eid"
   ],
   "claims_in_verified_claims_supported":[
      "given_name",
      "family_name",
      "birthdate",
      "place_of_birth",
      "nationalities",
      "address"
   ],
...
}
```

The OP MUST support the `claims` parameter and needs to publish this in its openid-configuration using the `claims_parameter_supported` element.

8. Transaction-specific Purpose

This specification introduces the request parameter `purpose` to allow an RP to state the purpose for the transfer of user data it is asking for.

`purpose`: OPTIONAL. String describing the purpose for obtaining certain user data from the OP. The purpose MUST NOT be shorter than 3 characters and MUST NOT be longer than 300 characters. If these rules are violated, the authentication request MUST fail and the OP returns an error `invalid_request` to the RP.

The OP MUST display this purpose in the respective user consent screen(s) in order to inform the user about the designated use of the data to be transferred or the authorization to be approved.

In order to ensure a consistent UX, the RP MAY send the `purpose` in a certain language and request the OP to use the same language using the `ui_locales` parameter.

If the parameter `purpose` is not present in the request, the OP MAY utilize a description that was pre-configured for the respective RP.

Note: In order to prevent injection attacks, the OP MUST escape the text appropriately before it will be shown in a user interface. The OP MUST expect special characters in the URL decoded purpose text provided by the RP. The OP MUST ensure that any special characters in the purpose text cannot be used to inject code into the

web interface of the OP (e.g., cross-site scripting, defacing). Proper escaping MUST be applied by the OP. The OP SHALL NOT remove characters from the purpose text to this end.

9. Privacy Consideration

OP and RP MUST establish a legal basis before exchanging any personally identifiable information. It can be established upfront or in the course of the OpenID process.

Timestamps with a time zone component can potentially reveal the person's location. To preserve the person's privacy timestamps within the verification element and verified claims that represent times SHOULD be represented in Coordinated Universal Time (UTC), unless there is a specific reason to include the time zone, such as the time zone being an essential part of a consented time related claim in verified data.

10. Security Considerations

The integrity and authenticity of the issued assertions MUST be ensured in order to prevent identity spoofing. The Claims source MUST therefore cryptographically sign all assertions.

The confidentiality of all user data exchanged between the protocol parties MUST be ensured using suitable methods at transport or application layer.

11. Predefined Values

This specification focuses on the technical mechanisms to convey verified claims and thus does not define any identifiers for trust frameworks, id documents, or verification methods. This is left to adopters of the technical specification, e.g. implementers, identity schemes, or jurisdictions.

Each party defining such identifier MUST ensure the collision resistance of this identifiers. This is achieved by including a domain name under the control of this party into the identifier name, e.g. `https://mycompany.com/identifiers/cool_verification_method`.

The eKYC and Identity Assurance Working Group maintains a wiki page [predefined_values_page] that can be utilized to share predefined values with other parties.

12. Normative References

[OpenID-Discovery] Sakimura, N., Bradley, J., de Medeiros, B., and E. Jay, "OpenID Connect Discovery 1.0 incorporating errata set 1", 8 November 2014, <https://openid.net/specs/openid-connect-discovery-1_0.html>.

[ISO3166-3] ISO, "ISO 3166-1:2013. Codes for the representation of names of countries and their subdivisions -- Part 3: Code for formerly used names of countries", 2013, <https://www.iso.org/standard/63547.html>.

[ICAO-Doc9303] INTERNATIONAL CIVIL AVIATION ORGANIZATION, "Machine Readable Travel Documents, Seventh Edition, 2015, Part 3: Specifications Common to all MRTDs", 2015, <https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf>.

[RFC8417] Hunt, P., Ed., Jones, M., Denniss, W., and M. Ansari, "Security Event Token (SET)", RFC 8417, DOI 10.17487/RFC8417, July 2018, <https://www.rfc-editor.org/info/rfc8417>.

[verified_claims.json]

OpenID Foundation, "JSON Schema for assertions using verified_claims", 2020, <https://openid.net/schemas/verified_claims-10.json>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[OpenID] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1", 8 November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[ISO3166-1] ISO, "ISO 3166-1:1997. Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes", 2013, <https://www.iso.org/standard/63545.html>.

[verified_claims_request.json] OpenID Foundation, "JSON Schema for requesting verified_claims", 2020, <https://openid.net/schemas/verified_claims_request-10.json>.

[predefined_values_page] OpenID Foundation, "Overview page for predefined values", 2020, <https://openid.net/wg/ekyc-ida/identifiers/>.

13. Informative References

[eIDAS] European Parliament, "REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC", 23 July 2014, <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014R0910>.

[NIST-SP-800-63a] Grassi, Paul. A., Fentony, James L., Lefkovitz, Naomi B., Danker, Jamie M., Choong, Yee-Yin., Greene, Kristen K., and Mary F. Theofanos, "NIST Special Publication 800-63A, Digital Identity Guidelines, Enrollment and Identity Proofing Requirements", June 2017, <https://doi.org/10.6028/NIST.SP.800-63a>.

[I-D.ietf-oauth-jwt-introspection-response] Lodderstedt, T. and V. Dzhuvinov, "JWT Response for OAuth Token Introspection", Work in Progress, Internet-Draft, draft-ietf-oauth-jwt-introspection-response-09, 25 April 2020, <https://tools.ietf.org/html/draft-ietf-oauth-jwt-introspection-response-09>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

Appendix A. IANA Considerations

A.1. JSON Web Token Claims Registration

This specification requests registration of the following value in the IANA "JSON Web Token Claims Registry" established by [RFC7519].

A.1.1. Registry Contents

Claim Name: `verified_claims`

Claim Description: This container claim is composed of the verification evidence related to a certain verification process and the corresponding Claims about the End-User which were verified in this process.

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Verified Claims of this document

Claim Name: `place_of_birth`

Claim Description: A structured Claim representing the End-User's place of birth.

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Claims of this document

Claim Name: `nationalities`

Claim Description: String array representing the user's nationalities.

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Claims of this document

Claim Name: `birth_family_name`

Claim Description: Family name someone has when he or she is born, or at least from the time he or she is a child. This term can be used by a person who changes the family name later in life for any reason.

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Claims of this document

Claim Name: `birth_given_name`

Claim Description: Given name someone has when he or she is born, or at least from the time he or she is a child. This term can be used by a person who changes the given name later in life for any reason.

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Claims of this document

Claim Name: `birth_middle_name`

Claim Description: Middle name someone has when he or she is born, or at least from the time he or she is a child. This term can be used by a person who changes the middle name later in life for any reason.

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Claims of this document

Claim Name: `salutation`

Claim Description: End-User's salutation, e.g. "Mr."

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Claims of this document

Claim Name: `title`

Claim Description: End-User's title, e.g. "Dr."

Change Controller: eKYC and Identity Assurance Working Group - openid-specs-ekyc-ida@lists.openid.net

Specification Document(s): Section Claims of this document

Appendix B. Acknowledgements

The following people at yes.com and partner companies contributed to the concept described in the initial contribution to this specification: Karsten Buch, Lukas Stiebig, Sven Manz, Waldemar Zimpfer, Willi Wiedergold, Fabian Hoffmann, Daniel Keijsers, Ralf Wagner, Sebastian Ebling, Peter Eisenhofer.

We would like to thank Joseph Heenan, Vladimir Dzhuvinov, Kosuke Koiwai, Azusa Kikuchi, Naohiro Fujie, Takahiko Kawasaki, Sebastian Ebling, Marcos Sanz, Tom Jones, Mike Pegman, Michael B. Jones, Jeff Lombardo, Taylor Ongaro, and Mark Haine for their valuable feedback and contributions that helped to evolve this specification.

Appendix C. Notices

Appendix D. Document History

[[To be removed from the final specification]]

-10

- Editorial improvements
- Improved JSON schema (alignment with spec and bug fix)

-09

- `verified_claims` element may contain one or more verified claims objects
- an individual assertion may contain `verified_claims` elements in the assertion itself and any aggregated or distributed claims sets it includes or refers to, respectively
- moved all definitions of pre-defined values for trust frameworks, id documents and verification methods to a wiki page as non-normative overview
- clarified and simplified request syntax
- reduced mandatory requirement `verified_claims` to bare minimum
- removed JSON schema from draft and added reference to JSON schema file instead
- added request JSON schema
- added IANA section with JSON Web Token Claims Registration
- integrated source into single md file
- added privacy considerations regarding time zone data, enhanced syntax definition of time and date-time fields in spec and response schema
- fixed typos

-08

- added `uripp` verification method
- small fixes to examples

-07

- fixed typos
- changed `nationality` String claim to `nationalities` String array claim
- replaced `agent` in id_document verifier element by `txn` element
- qes method: fixed error in description of `issuer`
- qes method: changed `issued_at` to `created_at` since this field applies to the signature (that is created and not issued)
- Changed format of `nationalities` and issuing `country` to ICAO codes
- Changed `date` in verification element to `time`
- Added Japanese trust frameworks to pre-defined values
- Added Japanese id documents to pre-defined values
- adapted JSON schema and examples

-06

- Incorporated review feedback by Marcos Sanz and Adam Cooper
- Added text on integrity, authenticity, and confidentiality for data passed between OP and RP to Security Considerations section
- added `purpose` field to `claims` parameter
- added feature to let the RP explicitly requested certain `verification` data

-05

- incorporated review feedback by Mike Jones
- Added OIDF Copyright Notices
- Moved Acknowledgements to Appendix A
- Removed RFC 2119 keywords from scope & requirements section and rephrased section
- rephrased introduction
- replaced `birth_name` with `birth_family_name`, `birth_given_name`, and `birth_middle_name`
- replaced `transaction_id` with `txn` from RFC 8417
- added references to eIDAS, ISO 3166-1, ISO 3166-3, and ISO 8601-2004
- added note on `purpose` and locales
- changed file name and document title to include 1.0 version id
- corrected evidence plural
- lots of editorial fixes
- Alignment with OpenID Connect Core wording
- Renamed `id` to `verification_process`
- Renamed `verified_person_data` to `verified_claims`

-04

- incorporated review feedback by Marcos Sanz

-03

- enhanced draft to support multiple evidence
- added a JSON Schema for assertions containing the `verified_person_data` Claim
- added more identity document definitions
- added `region` field to `place_of_birth` Claim
- changed `eidas_loa_substantial/high` to `eidas_ial_substantial/high`
- fixed typos in examples

- uppercased all editorial occurences of the term `claims` to align with OpenID Connect Core

-02

- added new request parameter `purpose`
- simplified/reduced number of verification methods
- simplfied identifiers
- added `identity_documents_supported` to metadata section
- improved examples

-01

- fixed some typos
- remove organization element (redundant) (issue 1080)
- allow other Claims about the End-User in the `claims` sub element (issue 1079)
- changed `legal_context` to `trust_framework`
- added explanation how the content of the verification element is determined by the trust framework
- added URI-based identifiers for `trust_framework`, `identity_document` and (verification) `method`
- added example attestation for notified/regulated eID system
- adopted OP metadata section accordingly
- changed error behavior for `max_age` member to alig with OpenID Core
- Added feature to let the RP express requirements for verification data (trust framework, identity documents, verification method)
- Added privacy consideration section and added text on legal basis for data exchange
- Added explanation about regulated and un-regulated eID systems

-00 (WG document)

- turned the proposal into a WG document
- changed name
- added terminology section and reworked introduction
- added several examples (ID Token vs UserInfo, unverified & verified claims, aggregated & distributed claims)
- incorporated text proposal of Marcos Sanz regarding max_age
- added IANA registration for new error code `unable_to_meet_requirement`

Authors' Addresses

Torsten Lodderstedt
yes.com
Email: torsten@lodderstedt.net

Daniel Fett
yes.com
Email: mail@danielfett.de