Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

O. Terbu
MATTR
D. Fett
Authlete Inc.
B. Campbell
Ping Identity
7 July 2025

SD-JWT-based Verifiable Credentials (SD-JWT VC)
draft-ietf-oauth-sd-jwt-vc-10

Abstract

This specification describes data formats as well as validation and processing rules to express Verifiable Credentials with JSON payloads with and without selective disclosure based on the SD-JWT [I-D.ietf-oauth-selective-disclosure-jwt] format.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/oauth-wg/oauth-sd-jwt-vc.

Status of This Memo

Table of Contents

1. Introduction

1.1. Issuer-Holder-Verifier Model

In the so-called Issuer-Holder-Verifier Model, Issuers issue so-called Verifiable Credentials to a Holder, who can then present the Verifiable Credentials to Verifiers. Verifiable Credentials are cryptographically secured statements about a Subject, typically the Holder.

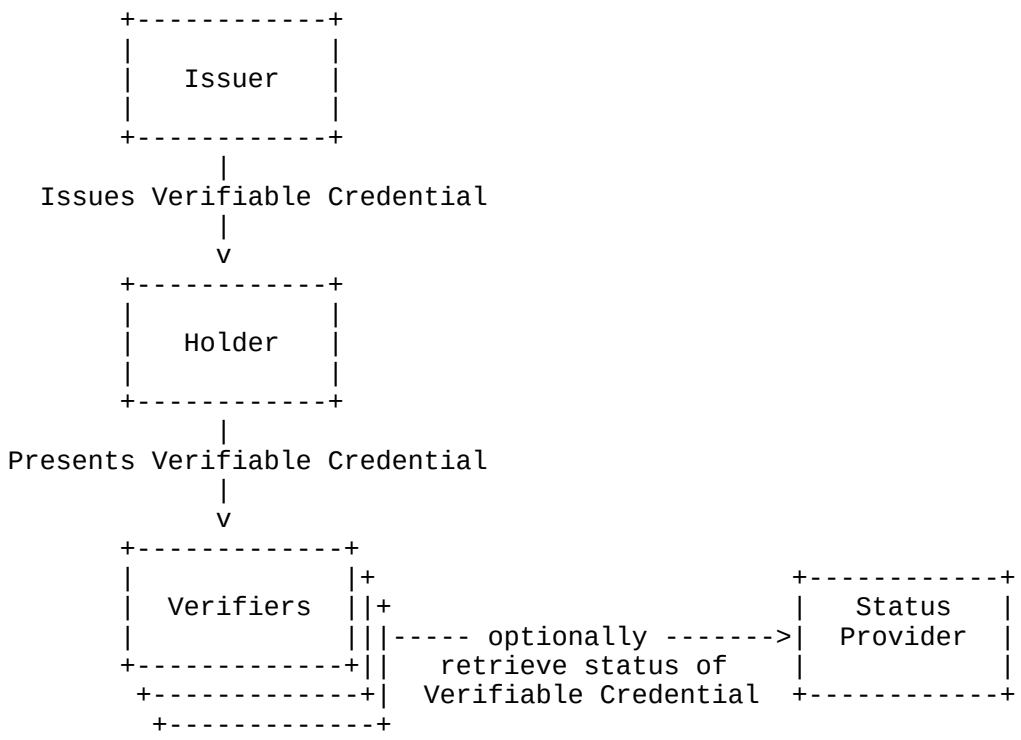

Figure 1: Issuer-Holder-Verifier Model with optional Status Provider

Verifiers can check the authenticity of the data in the Verifiable Credentials and optionally enforce Key Binding, i.e., ask the Holder to prove that they are the intended holder of the Verifiable Credential, for example, by proving possession of a cryptographic key referenced in the credential. This process is further described in [I-D.ietf-oauth-selective-disclosure-jwt].

To support revocation of Verifiable Credentials, revocation information can optionally be retrieved from a Status Provider. The role of a Status Provider can be fulfilled by either a fourth party or by the Issuer.

1.2. SD-JWT as a Credential Format

JSON Web Tokens (JWTs) [RFC7519] can in principle be used to express Verifiable Credentials in a way that is easy to understand and process as it builds upon established web primitives.

Selective Disclosure JWT (SD-JWT) [I-D.ietf-oauth-selective-disclosure-jwt] is a specification that introduces conventions to support selective disclosure for JWTs: For an SD-JWT document, a Holder can decide which claims to release (within bounds defined by the Issuer).

SD-JWT is a superset of JWT as it can also be used when there are no selectively disclosable claims and also supports JWS JSON serialization, which is useful for long term archiving and multi signatures. However, SD-JWT itself does not define the claims that must be used within the payload or their semantics.

This specification uses SD-JWT and the well-established JWT content rules and extensibility model as basis for representing Verifiable Credentials with JSON payloads. These Verifiable Credentials are called SD-JWT VCs. The use of selective disclosure in SD-JWT VCs is OPTIONAL.

SD-JWTs VC can contain claims that are registered in "JSON Web Token Claims" registry as defined in [RFC7519], as well as public and private claims.

Note: This specification does not utilize the W3C’s Verifiable Credentials Data Model v1.0, v1.1, or v2.0.

1.3. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.4. Terms and Definitions

This specification uses the terms "Holder", "Issuer", "Verifier", "Disclosure", "Selectively Disclosable JWT (SD-JWT)", "Key Binding", "Key Binding JWT (KB-JWT)", "Selectively Disclosable JWT with Key Binding (SD-JWT+KB)" defined by [I-D.ietf-oauth-selective-disclosure-jwt].

Consumer: Applications using the Type Metadata specified in Section 6 are called Consumer. This typically includes Issuers, Verifiers, and Wallets.

Verifiable Credential (VC): An assertion with claims about a Subject that is cryptographically secured by an Issuer (usually by a digital signature).

SD-JWT-based Verifiable Credential (SD-JWT VC): A Verifiable

Credential encoded using the format defined in [I-D.ietf-oauth-selective-disclosure-jwt]. It may or may not contain selectively disclosable claims.

Unsecured Payload of an SD-JWT VC: A JSON object containing all selectively disclosable and non-selectively disclosable claims of the SD-JWT VC. The Unsecured Payload acts as the input JSON object to issue an SD-JWT VC complying to this specification.

Status Provider: An entity that provides status information (e.g. revocation) about a Verifiable Credential.

2. Scope

* This specification defines
  - Data model and media types for Verifiable Credentials based on SD-JWTs.
  - Validation and processing rules for Verifiers and Holders.

3. Verifiable Credentials based on SD-JWT

This section defines encoding, validation and processing rules for SD-JWT VCs.

3.1. Media Type

SD-JWT VCs compliant with this specification MUST use the media type application/dc+sd-jwt as defined in Section 3.1.

The base subtype name dc is meant to stand for "digital credential", which is a term that is emerging as a conceptual synonym for "verifiable credential".

3.2. Data Format

SD-JWT VCs MUST be encoded using the SD-JWT format defined in Section 4 of [I-D.ietf-oauth-selective-disclosure-jwt]. A presentation of an SD-JWT VC MAY contain a KB-JWT.

Note that in some cases, an SD-JWT VC MAY have no selectively disclosable claims, and therefore the encoded SD-JWT will not contain any Disclosures.

3.2.1. JOSE Header

This section defines JWT header parameters for the SD-JWT component of the SD-JWT VC.

The typ header parameter of the SD-JWT MUST be present. The typ value MUST use dc+sd-jwt. This indicates that the payload of the SD-JWT contains plain JSON and follows the rules as defined in this specification. It further indicates that the SD-JWT is a SD-JWT component of a SD-JWT VC.

The following is a non-normative example of a decoded SD-JWT header:

```
{
  "alg": "ES256",
  "typ": "dc+sd-jwt"
}
```

Note that this draft used vc+sd-jwt as the value of the typ header from its inception in July 2023 until November 2024 when it was changed to dc+sd-jwt to avoid conflict with the vc media type name registered by the W3C's Verifiable Credentials Data Model draft. In order to facilitate a minimally disruptive transition, it is RECOMMENDED that Verifiers and Holders accept both vc+sd-jwt and dc+sd-jwt as the value of the typ header for a reasonable transitional period.

3.2.2. JWT Claims Set

This section defines the claims that can be included in the payload of SD-JWT VCs.

3.2.2.1. Verifiable Credential Type - vct Claim

This specification defines the new JWT claim vct (for verifiable credential type). The vct value MUST be a case-sensitive StringOrURI (see [RFC7519]) value serving as an identifier for the type of the SD-JWT VC. The vct value MUST be a Collision-Resistant Name as defined in Section 2 of [RFC7515].

A type is associated with rules defining which claims may or must appear in the Unsecured Payload of the SD-JWT VC and whether they may, must, or must not be selectively disclosable. This specification does not define any vct values; instead it is expected that ecosystems using SD-JWT VCs define such values including the semantics of the respective claims and associated rules (e.g., policies for issuing and validating credentials beyond what is defined in this specification).

The following is a non-normative example of how vct is used to express a type:

```
{
  "vct": "https://credentials.example.com/identity_credential"
}
```

For example, a value of https://credentials.example.com/identity_credential can be associated with rules that define that at least the registered JWT claims given_name, family_name, birthdate, and address must appear in the Unsecured Payload. Additionally, the registered JWT claims email and phone_number, and the private claims is_over_18, is_over_21, and is_over_65 may be used. The type might also indicate that any of the aforementioned claims can be selectively disclosable.

3.2.2.2. Registered JWT Claims

SD-JWT VCs MAY use any claim registered in the "JSON Web Token Claims" registry as defined in [RFC7519].

The following registered JWT claims are used within the SD-JWT component of the SD-JWT VC and MUST NOT be included in the Disclosures, i.e., cannot be selectively disclosed:

* iss
  - OPTIONAL. As defined in Section 4.1.1 of [RFC7519] this claim explicitly indicates the Issuer of the Verifiable Credential when it is not conveyed by other means (e.g., the subject of the end-entity certificate of an x5c header).
* nbf
  - OPTIONAL. The time before which the Verifiable Credential MUST NOT be accepted before validating. See [RFC7519] for more information.
* exp
  - OPTIONAL. The expiry time of the Verifiable Credential after which the Verifiable Credential is no longer valid. See [RFC7519] for more information.
* cnf
  - OPTIONAL unless cryptographic Key Binding is to be supported, in which case it is REQUIRED. Contains the confirmation method identifying the proof of possession key as defined in [RFC7800]. It is RECOMMENDED that this contains a JWK as defined in Section 3.2 of [RFC7800]. For proof of cryptographic Key Binding, the KB-JWT in the presentation of the SD-JWT MUST be secured by the key identified in this claim.
* vct
  - REQUIRED. The type of the Verifiable Credential, e.g., https://credentials.example.com/identity_credential, as defined in Section 3.2.2.1.
* status

- OPTIONAL. The information on how to read the status of the Verifiable Credential. See [I-D.ietf-oauth-status-list] for more information.

The following registered JWT claims are used within the SD-JWT component of the SD-JWT VC and MAY be included in Disclosures, i.e., can be selectively disclosed:

* sub
  - OPTIONAL. The identifier of the Subject of the Verifiable Credential. The Issuer MAY use it to provide the Subject identifier known by the Issuer. There is no requirement for a binding to exist between sub and cnf claims.
* iat
  - OPTIONAL. The time of issuance of the Verifiable Credential. See [RFC7519] for more information.

3.2.2.3. Public and Private JWT claims

Additionally, any public and private claims as defined in Sections 4.2 and 4.3 of [RFC7519] MAY be used.

3.2.2.4. SD-JWT VC without Selectively Disclosable Claims

An SD-JWT VC MAY have no selectively disclosable claims. In that case, the SD-JWT VC MUST NOT contain the _sd claim in the JWT body. It also MUST NOT have any Disclosures.

3.3. Example

The following is a non-normative example of the user data of an unsecured payload of an SD-JWT VC.

```
{
  "vct": "https://credentials.example.com/identity_credential",
  "given_name": "John",
  "family_name": "Doe",
  "email": "johndoe@example.com",
  "phone_number": "+1-202-555-0101",
  "address": {
    "street_address": "123 Main St",
    "locality": "Anytown",
    "region": "Anystate",
    "country": "US"
  },
  "birthdate": "1940-01-01",
  "is_over_18": true,
  "is_over_21": true,
  "is_over_65": true
}
```

The following is a non-normative example of how the unsecured payload of the SD-JWT VC above can be used in an SD-JWT where the resulting SD-JWT VC contains only claims about the Subject that are selectively disclosable:

```
{
  "_sd": [
    "09vKrJMOlyTWM0sjpu_pdOBVBQ2M1y3KhpH515nXkpY",
    "2rsjGbaC0ky8mT0pJrPioWTq0_daw1sX76poUlgCwbI",
    "EkO8dhW0dHEJbvUHlE_VCeuC9uRELOieLZhh7XbUTtA",
    "IlDzIKeiZdDwpqpK6ZfbyphFvz5FgnWa-sN6wqQXCiw",
    "JzYjH4svliH0R3PyEMfeZu6Jt69u5qehZo7F7EPYlSE",
    "PorFbpKuVu6xymJagvkFsFXAbRoc2JGlAUA2BA4o7cI",
    "TGf4oLbgwd5JQaHyKVQZU9UdGE0w5rtDsrZzfUaomLo",
    "jdrTE8YcbY4EifugihiAe_BPekxJQZICeiUQwY9QqxI",
    "jsu9yVulwQQlhFlM_3JlzMaSFzglhQG0DpfayQwLUK4"
  ],
  "iss": "https://example.com/issuer",
  "iat": 1683000000,
  "exp": 1883000000,
  "vct": "https://credentials.example.com/identity_credential",
  "_sd_alg": "sha-256",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```

Note that a cnf claim has been added to the SD-JWT payload to express the confirmation method of the Key Binding.

The following are the Disclosures belonging to the SD-JWT payload above:

Claim given_name:

* SHA-256 Hash: jsu9yVulwQQlhFlM_3JlzMaSFzglhQG0DpfayQwLUK4
* Disclosure:
  WyIyR0xDNDJzS1F2ZUNmR2ZyeU5STjl3IiwgImdpdmVuX25hbWUiLCAiSm9o
  biJd
* Contents: ["2GLC42sKQveCfGfryNRN9w", "given_name", "John"]

Claim family_name:

* SHA-256 Hash: TGf4oLbgwd5JQaHyKVQZU9UdGE0w5rtDsrZzfUaomLo
* Disclosure:
  WyJlbHVWNU9nM2dTTklJOEVZbnN4QV9BIiwgImZhbWlseV9uYW1lIiwgIkRv
  ZSJd
* Contents: ["eluV5Og3gSNII8EYnsxA_A", "family_name", "Doe"]

Claim email:

* SHA-256 Hash: JzYjH4svliH0R3PyEMfeZu6Jt69u5qehZo7F7EPYlSE
* Disclosure:
  WyI2SWo3dE0tYTVpVlBHYm9TNXRtdlZBIiwgImVtYWlsIiwgImpvaG5kb2VA
  ZXhhbXBsZS5jb20iXQ
* Contents: ["6Ij7tM-a5iVPGboS5tmvVA", "email",
  "johndoe@example.com"]

Claim phone_number:

* SHA-256 Hash: PorFbpKuVu6xymJagvkFsFXAbRoc2JGlAUA2BA4o7cI
* Disclosure:
  WyJlSThaV205UW5LUHBOUGVOZW5IZGhRIiwgInBob25lX251bWJlciIsICIr
  MS0yMDItNTU1LTAxMDEiXQ
* Contents: ["eI8ZWm9QnKPpNPeNenHdhQ", "phone_number",
  "+1-202-555-0101"]

Claim address:

* SHA-256 Hash: IlDzIKeiZdDwpqpK6ZfbyphFvz5FgnWa-sN6wqQXCiw
* Disclosure:
  WyJRZ19PNjR6cUF4ZTQxMmExMDhpcm9BIiwgImFkZHJlc3MiLCB7InN0cmVl
  dF9hZGRyZXNzIjogIjEyMyBNYWluIFN0IiwgImxvY2FsaXR5IjogIkFueXRv
  d24iLCAicmVnaW9uIjogIkFueXN0YXRlIiwgImNvdW50cnkiOiAiVVMifV0
* Contents: ["Qg_O64zqAxe412a108iroA", "address", {"street_address":
  "123 Main St", "locality": "Anytown", "region": "Anystate",
  "country": "US"}]

Claim birthdate:

* SHA-256 Hash: jdrTE8YcbY4EifugihiAe_BPekxJQZICeiUQwY9QqxI
* Disclosure:
  WyJBSngtMDk1VlBycFR0TjRRTU9xUk9BIiwgImJpcnRoZGF0ZSIsICIxOTQw
  LTAxLTAxIl0
* Contents: ["AJx-095VPrpTtN4QMOqROA", "birthdate", "1940-01-01"]

Claim is_over_18:

* SHA-256 Hash: 09vKrJMOlyTWM0sjpu_pdOBVBQ2M1y3KhpH515nXkpY
* Disclosure:
  WyJQYzMzSk0yTGNoY1VfbEhnZ3ZfdWZRIiwgImlzX292ZXJfMTgiLCB0cnVl
  XQ
* Contents: ["Pc33JM2LchcU_lHggv_ufQ", "is_over_18", true]

Claim is_over_21:

* SHA-256 Hash: 2rsjGbaC0ky8mT0pJrPioWTq0_daw1sX76poUlgCwbI

* Disclosure:
  WyJHMDJOU3JRZmpGWFE3SW8wOXN5YWpBIiwgImlzX292ZXJfMjEiLCB0cnVl
  XQ
* Contents: ["G02NSrQfjFXQ7Io09syajA", "is_over_21", true]

Claim is_over_65:

* SHA-256 Hash: EkO8dhW0dHEJbvUHlE_VCeuC9uRELOieLZhh7XbUTtA
* Disclosure:
  WyJsa2x4RjVqTVlsR1RQVW92TU5JdkNBIiwgImlzX292ZXJfNjUiLCB0cnVl
  XQ
* Contents: ["lklxF5jMYlGTPUovMNIvCA", "is_over_65", true]

The SD-JWT and the Disclosures would then be serialized by the Issuer into the following format for issuance to the Holder:

eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZG9jLXNp
Z25lci0wNS0yNS0yMDIyIn0.eyJfc2QiOiBbIjA5dktySk1PbHlUV00wc2pwdV9wZE9C
VkJRMk0xeTNLaHBINTE1blhrcFkiLCAiMnJzakdiYUMwa3k4bVQwcEpyUGlvV1RxMF9k
YXcxc1g3NnBvVWxnQ3diSSIsICJFa084ZGhXMGRIRUpidlVIbEVfVkNldUM5dVJFTE9p
ZUxaaGg3WGJVVHRBIiwgIklsRHpJS2VpWmREd3BxcEs2WmZieXBoRnZ6NUZnbldhLXNO
NndxUVhDaXciLCAiSnpZakg0c3ZsaUgwUjNQeUVNZmVadTZKdDY5dTVxZWhabzdGN0VQ
WWxTRSIsICJQb3JGYnBLdVZ1Nnh5bUphZ3ZrRnNGWEFiUm9jMkpHbEFVQTJCQTRvN2NJ
IiwgIlRHZjRvTGJnd2Q1SlFhSHlLVlFaVTlVZEdFMHc1cnREc3JaemZVYW9tTG8iLCAi
amRyVEU4WWNiWTRFaWZ1Z2loaUFlX0JQZWt4SlFaSUNlaVVRd1k5UXF4SSIsICJqc3U5
eVZ1bHdRUWxoRmxNXzNKbHpNYVNGemdsaFFHMERwZmF5UXdMVUs0Il0sICJpc3MiOiAi
aHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXIiLCAiaWF0IjogMTY4MzAwMDAwMCwgImV4
cCI6IDE4ODMwMDAwMDAsICJ2Y3QiOiAiaHR0cHM6Ly9jcmVkZW50aWFscy5leGFtcGxl
LmNvbS9pZGVudGl0eV9jcmVkZW50aWFsIiwgIl9zZF9hbGciOiAic2hhLTI1NiIsICJj
bmYiOiB7Imp3ayI6IHsia3R5IjogIkVDIiwgImNydiI6ICJQLTI1NiIsICJ4IjogIlRD
QUVSMTladnUzT0hGNGo0VzR2ZlNWb0hJUDFJTGlsRGxzN3ZDZUdlbWMiLCAieSI6ICJa
eGppV1diWk1RR0hWV0tWUTRoYlNJaXJzVmZ1ZWNDRTZ0NGpUOUYySFpRIn19fQ.SLMD7
IrWS996ft4ey63Us4yPMNf6j3JM3kKNFlXHRqfFhXpbU1o27Tozf8pPqMdryKUDFoYGn
Y2_2CFH8FQo7g~WyIyR0xDNDJzS1F2ZUNmR2ZyeU5STjl3IiwgImdpdmVuX25hbWUiLC
AiSm9obiJd~WyJlbHVWNU9nM2dTTklOEVZbnN4QV9BIiwgImZhbWlseV9uYW1lIiwgI
kRvZSJd~WyI2SWo3dE0tYTVpVlBHYm9TNXRtdlZBIiwgImVtYWlsIiwgImpvaG5kb2VA
ZXhhbXBsZS5jb20iXQ~WyJlSThaV205UW5LUHBOUGVOZW5IZGhRIiwgInBob25lX251b
WJlciIsICIrMS0yMDItNTU1LTAxMDEiXQ~WyJRZ19PNjR6cUF4ZTQxMmExMDhpcm9BIi
wgImFkZHJlc3MiLCB7InN0cmVldF9hZGRyZXNzIjogIjEyMyBNYWluIFN0IiwgImxvY2
FsaXR5IjogIkFueXRvd24iLCAicmVnaW9uIjogIkFueXN0YXRlIiwgImNvdW50cnkiOi
AiVVMifV0~WyJBSngtMDk1VlBycFR0TjRRTU9xUk9BIiwgImJpcnRoZGF0ZSIsICIxOT
QwLTAxLTAxIl0~WyJQYzMzSk0yTGNoY1VfbEhnZ3ZfdWZRIiwgImlzX292ZXJfMTgiLC
B0cnVlXQ~WyJHMDJOU3JRZmpGWFE3SW8wOXN5YWpBIiwgImlzX292ZXJfMjEiLCB0cnV
lXQ~WyJsa2x4RjVqTVlsR1RQVW92TU5JdkNBIiwgImlzX292ZXJfNjUiLCB0cnVlXQ~

Examples of what presentations of SD-JWT VCs might look like are provided in Section 4.2.

3.4. Verification and Processing

The recipient (Holder or Verifier) of an SD-JWT VC MUST process and verify an SD-JWT VC as described in Section 7 of [I-D.ietf-oauth-selective-disclosure-jwt]. The check in point 2.3 of Section 7.1 of [I-D.ietf-oauth-selective-disclosure-jwt], which validates the Issuer and ensures that the signing key belongs to the Issuer, MUST be satisfied by determining and validating the public verification key used to verify the Issuer-signed JWT, employing an Issuer Signature Mechanism (defined in Section 3.5) that is permitted for the Issuer according to policy.

If Key Binding is required (refer to the security considerations in Section 9.5 of [I-D.ietf-oauth-selective-disclosure-jwt]), the Verifier MUST verify the KB-JWT according to Section 7.3 of [I-D.ietf-oauth-selective-disclosure-jwt]. To verify the KB-JWT, the cnf claim of the SD-JWT MUST be used.

If a schema is provided in the Type Metadata, a recipient MUST validate the schema as defined in Section 6.5.

If there are no selectively disclosable claims, there is no need to process the _sd claim nor any Disclosures.

If status is present in the verified payload of the SD-JWT, the status SHOULD be checked. It depends on the Verifier policy to reject or accept a presentation of a SD-JWT VC based on the status of the Verifiable Credential.

Any claims used that are not understood MUST be ignored.

Additional validation rules MAY apply, but their use is out of the scope of this specification.

3.5. Issuer Signature Mechanisms

An Issuer Signature Mechanism defines how a Verifier determines the appropriate key and associated procedure for verifying the signature of an Issuer-signed JWT. This allows for flexibility in supporting different trust anchoring systems and key resolution methods without changing the core processing model.

A recipient MUST determine and validate the public verification key for the Issuer-signed JWT using a supported Issuer Signature Mechanism that is permitted for the given Issuer according to policy. This specification defines the following two Issuer Signature Mechanisms:

* JWT VC Issuer Metadata: A mechanism to retrieve the Issuer's public key using web-based resolution. When the value of the iss claim of the Issuer-signed JWT is an HTTPS URI, the recipient obtains the public key using the keys from JWT VC Issuer Metadata as defined in Section 5.

* X.509 Certificates: A mechanism to retrieve the Issuer's public key using the X.509 certificate chain in the SD-JWT header. When the header of the Issuer-signed JWT contains the x5c header parameter, the recipient uses the public key from the end-entity certificate of the certificates from the x5c header parameter and validates the X.509 certificate chain accordingly. In this case, the Issuer of the Verifiable Credential is the subject of the end-entity certificate.

To enable different trust anchoring systems or key resolution methods, separate specifications or ecosystem regulations may define additional Issuer Signature Mechanisms; however, the specifics of such mechanisms are out of scope for this specification. See Section 10.2 for related security considerations.

If a recipient cannot validate that the public verification key corresponds the Issuer of the Issuer-signed JWT using a permitted Issuer Signature Mechanism, the SD-JWT VC MUST be rejected.

4. Presenting Verifiable Credentials

This section defines encoding, validation and processing rules for presentations of SD-JWT VCs.

4.1. Key Binding JWT

If the presentation of the SD-JWT VC is encoded as an SD-JWT+KB, the KB-JWT MUST adhere to the rules defined in Section 4.3 of [I-D.ietf-oauth-selective-disclosure-jwt].

The KB-JWT MAY include additional claims which, when not understood, MUST be ignored by the Verifier.

4.2. Examples

The following is a non-normative example of a presentation of the SD-JWT shown in Section 3.3 including a KB-JWT. In this presentation, the Holder provides only the Disclosures for the address and is_over_65 claims. Other claims are not disclosed to the Verifier.

eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZG9jLXNp
Z25lci0wNS0yNS0yMDIyIn0.eyJfc2QiOiBbIjA5dktySk1PbHlUV00wc2pwdV9wZE9C
VkJRMk0xeTNLaHBINTE1blhrcFkiLCAiMnJzakdiYUMwa3k4bVQwcEpyUGlvV1RxMF9k
YXcxc1g3NnBvVWxnQ3diSSIsICJFa084ZGhXMGRIRUpidlVIbEVfVkNldUM5dVJFTE9p
ZUxaaGg3WGJVVHRBIiwgIklsRHpJS2VpWmREd3BxcEs2WmZieXBoRnZ6NUZnbldhLXNO
NndxUVhDaXciLCAiSnpZakg0c3ZsaUgwUjNQeUVNZmVadTZKdDY5dTVxZWhabzdGN0VQ
WWxTRSIsICJQb3JGYnBLdVZ1Nnh5bUphZ3ZrRnNGWEFiUm9jMkpHbEFVQTJCQTRvN2NJ
IiwgIlRHZjRvTGJnd2Q1SlFhSHlLVlFaVTlVZEdFMHc1cnREc3JaemZVYW9tTG8iLCAi
amRyVEU4WWNiWTRFaWZ1Z2loaUFlX0JQZWt4SlFaSUNlaVVRd1k5UXF4SSIsICJqc3U5
eVZ1bHdRUWxoRmxNXzNKbHpNYVNGemdsaFFHMERwZmF5UXdMVUs0Il0sICJpc3MiOiAi
aHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXIiLCAiaWF0IjogMTY4MzAwMDAwMCwgImV4
cCI6IDE4ODMwMDAwMDAsICJ2Y3QiOiAiaHR0cHM6Ly9jcmVkZW50aWFscy5leGFtcGxl
LmNvbS9pZGVudGl0eV9jcmVkZW50aWFsIiwgIl9zZF9hbGciOiAic2hhLTI1NiIsICJj
bmYiOiB7Imp3ayI6IHsia3R5IjogIkVDIiwgImNydiI6ICJQLTI1NiIsICJ4IjogIlRD
QUVSMTladnUzT0hGNGo0VzR2ZlNWb0hJUDFJTGlsRGxzN3ZDZUdlbWMiLCAieSI6ICJa
eGppV1diWk1RR0hWV0tWUTRoYlNJaXJzVmZ1ZWNDRTZ0NGpUOUYySFpRIn19fQ.SLMD7
IrWS996ft4ey63Us4yPMNf6j3JM3kKNFlXHRqfFhXpbU1o27Tozf8pPqMdryKUDFoYGn
Y2_2CFH8FQo7g~WyJsa2x4RjVqTVlsR1RQVW92TU5JdkNBIiwgImlzX292ZXJfNjUiLC
B0cnVlXQ~WyJRZ19PNjR6cUF4ZTQxMmExMDhpcm9BIiwgImFkZHJlc3MiLCB7InN0cmV
ldF9hZGRyZXNzIjogIjEyMyBNYWluIFN0IiwgImxvY2FsaXR5IjogIkFueXRvd24iLCA
icmVnaW9uIjogIkFueXN0YXRlIiwgImNvdW50cnkiOiAiVVMifV0~eyJhbGciOiAiRVM
yNTYiLCAidHlwIjogImtiK2p3dCJ9.eyJub25jZSI6ICIxMjM0NTY3ODkwIiwgImF1ZC
I6ICJodHRwczovL2V4YW1wbGUuY29tL3ZlcmlmaWVyIiwgImlhdCI6IDE3NTE5MDg2Nz
ksICJzZF9oYXNoIjogIk9vZ2w1cjkxdXpjNUQ1NGJWdVJham1xTVdHZDZFWjJzWEMxdF
hRZGM4UEkifQ.z42nFLM500VQK1_4JLAf1dPVyStV0D4F2HPcs-bpA5MdWmuFCfg0xRy
xmjY2QpnScl4_s9l22E7C35nyA3yLwg

After validation, the Verifier will have the following processed SD-JWT payload available for further handling:

```
{
  "iss": "https://example.com/issuer",
  "iat": 1683000000,
  "exp": 1883000000,
  "vct": "https://credentials.example.com/identity_credential",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  },
  "is_over_65": true,
  "address": {
    "street_address": "123 Main St",
    "locality": "Anytown",
    "region": "Anystate",
    "country": "US"
  }
}
```

The following example shows a presentation of a (similar but different) SD-JWT without a KB-JWT:

```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCJ9.eyJfc2QiOiBbIjA5dkt
ySk1PbHlUV00wc2pwdV9wZE9CVkJRMk0xeTNLaHBINTE1blhrcFkiLCAiMnJzakdiYUM
wa3k4bVQwcEpyUGlvV1RxMF9kYXcxc1g3NnBvVWxnQ3diSSIsICJFa084ZGhXMGRIRUp
idlVIbEVfVkNldUM5dVJFTE9pZUxaaGg3WGJVVHRBIiwgIklsRHpJS2VpWmREd3BxcEs
2WmZieXBoRnZ6NUZnbldhLXNONndxUVhDaXciLCAiSnpZakg0c3ZsaUgwUjNQeUVNZmV
adTZKdDY5dTVxZWhabzdGN0VQWWxTRSIsICJQb3JGYnBLdVZ1Nnh5bUphZ3ZrRnNGWEF
iUm9jMkpHbEFVQTJCQTRvN2NJIiwgIlRHZjRvTGJnd2Q1SlFhSHlLVlFaVTlVZEdFMHc
1cnREc3JaemZVYW9tTG8iLCAiamRyVEU4WWNiWTRFaWZ1Z2loaUFlX0JQZWt4SlFaSUN
laVVRd1k5UXF4SSIsICJqc3U5eVZ1bHdRUWxoRmxNXzNKbHpNYVNGemdsaFFHMERwZmF
5UXdMVUs0Il0sICJpc3MiOiAiaHR0cHM6Ly9leGFtcGxlLmNvbS9pc3N1ZXIiLCAiaWF
0IjogMTY4MzAwMDAwMCwgImV4cCI6IDE4ODMwMDAwMDAsICJ2Y3QiOiAiaHR0cHM6Ly9
jcmVkZW50aWFscy5leGFtcGxlLmNvbS9pZGVudGl0eV9jcmVkZW50aWFsIiwgIl9zZF9
hbGciOiAic2hhLTI1NiJ9.OpvpMh-RdwmS2YdGcMRv8LJ5JeIdPHH6HgERpPaVLoGeAH
MhsuAfWWp-NXgaTtwPl4CBBLja3AMB-AYFWrRcnw~WyJsa2x4RjVqTVlsR1RQVW92TU5
JdkNBIiwgImlzX292ZXJfNjUiLCB0cnVlXQ~WyJRZ19PNjR6cUF4ZTQxMmExMDhpcm9B
IiwgImFkZHJlc3MiLCB7InN0cmVldF9hZGRyZXNzIjogIjEyMyBNYWluIFN0IiwgImxv
Y2FsaXR5IjogIkFueXRvd24iLCAicmVnaW9uIjogIkFueXN0YXRlIiwgImNvdW50cnki
OiAiVVMifV0~
```

The Verifier will have the following processed SD-JWT payload after validation:

```
{
  "iss": "https://example.com/issuer",
  "iat": 1683000000,
  "exp": 1883000000,
  "vct": "https://credentials.example.com/identity_credential",
  "is_over_65": true,
  "address": {
    "street_address": "123 Main St",
    "locality": "Anytown",
    "region": "Anystate",
    "country": "US"
  }
}
```

5. JWT VC Issuer Metadata

This specification defines the JWT VC Issuer Metadata to retrieve the JWT VC Issuer Metadata configuration of the Issuer of the SD-JWT VC. The Issuer is identified by the iss claim in the JWT. Use of the JWT VC Issuer Metadata is OPTIONAL.

Issuers publishing JWT VC Issuer Metadata MUST make a JWT VC Issuer Metadata configuration available at the location formed by inserting the well-known string /.well-known/jwt-vc-issuer between the host component and the path component (if any) of the iss claim value in the JWT. The iss MUST be a case-sensitive URL using the HTTPS scheme that contains scheme, host and, optionally, port number and path components, but no query or fragment components.

5.1. JWT VC Issuer Metadata Request

A JWT VC Issuer Metadata configuration MUST be queried using an HTTP GET request at the path defined in Section 5.

The following is a non-normative example of an HTTP request for the JWT VC Issuer Metadata configuration when iss is set to https://example.com:

```
GET /.well-known/jwt-vc-issuer HTTP/1.1
Host: example.com
```

If the iss value contains a path component, any terminating / MUST be removed before inserting /.well-known/ and the well-known URI suffix between the host component and the path component.

The following is a non-normative example of a HTTP request for the JWT VC Issuer Metadata configuration when iss is set to https://example.com/tenant/1234:

```
GET /.well-known/jwt-vc-issuer/tenant/1234 HTTP/1.1
Host: example.com
```

5.2. JWT VC Issuer Metadata Response

A successful response MUST use the 200 OK HTTP and return the JWT VC Issuer Metadata configuration using the application/json content type.

An error response uses the applicable HTTP status code value.

This specification defines the following JWT VC Issuer Metadata configuration parameters:

* issuer
  - REQUIRED. The Issuer identifier, which MUST be identical to the iss value in the JWT.
* jwks_uri
  - OPTIONAL. URL string referencing the Issuer's JSON Web Key (JWK) Set [RFC7517] document which contains the Issuer's public keys. The value of this field MUST point to a valid JWK Set document.
* jwks
  - OPTIONAL. Issuer's JSON Web Key Set [RFC7517] document value, which contains the Issuer's public keys. The value of this field MUST be a JSON object containing a valid JWK Set.

JWT VC Issuer Metadata MUST include either jwks_uri or jwks in their JWT VC Issuer Metadata, but not both.

It is RECOMMENDED that the JWT contains a kid JWT header parameter that can be used to look up the public key in the JWK Set included by value or referenced in the JWT VC Issuer Metadata.

The following is a non-normative example of a JWT VC Issuer Metadata configuration including jwks:

```
{
   "issuer":"https://example.com",
   "jwks":{
      "keys":[
         {
            "kid":"doc-signer-05-25-2022",
            "kty": "EC", "crv": "P-256",
            "x": "b28d4MwZMjw8-00CG4xfnn9SLMVMM19SlqZpVb_uNtQ",
            "y": "Xv5zWwuoaTgdS6hV43yI6gBwTnjukmFQQnJ_kCxzqk8"
         },
         {
            "kid":"doc-signer-06-05-2025",
            "kty": "EC", "crv": "P-256",
            "x":"DfjUx4WHBds61vGbqUQhsy3FGX13fAS13QWh2EHIkX8",
            "y":"NfqJt9Kp0EA93xq9ysO80DRZ_hCGlISz-pYLgv4RFvg"
         }
      ]
   }
}
```

The following is a non-normative example of a JWT VC Issuer Metadata configuration including jwks_uri:

```
{
   "issuer":"https://example.com",
   "jwks_uri":"https://jwt-vc-issuer.example.com/my_public_keys.jwks"
}
```

Additional JWT VC Issuer Metadata configuration parameters MAY also be used.

5.3. JWT VC Issuer Metadata Validation

The issuer value returned MUST be identical to the iss value of the JWT. If these values are not identical, the data contained in the response MUST NOT be used.

6. SD-JWT VC Type Metadata

An SD-JWT VC type, i.e., the vct value, is associated with Type Metadata defining, for example, information about the type or a schema defining (see Section 6.5.1) which claims MAY or MUST appear in the SD-JWT VC, and how credentials are displayed.

This section defines Type Metadata that can be associated with a type of an SD-JWT VC, as well as a method for retrieving the Type Metadata and processing rules. This Type Metadata is intended to be used, among other things, for the following purposes:

* Developers of Issuers and Verifiers can use the Type Metadata to understand the semantics of the type and the associated rules. While in some cases, Issuers are the parties that define types, this is not always the case. For example, a type can be defined by a standardization body or a community.
* Verifiers can use the Type Metadata to determine whether a credential is valid according to the rules of the type. For example, a Verifier can check whether a credential contains all required claims and whether the claims are selectively disclosable.
* Wallets can use the metadata to display the credential in a way that is consistent with the intent of the provider of the Type Metadata.

Type Metadata can be retrieved as described in Section 6.3.

6.1. Type Metadata Example

All examples in this section are non-normative.

The following is an example of an SD-JWT VC payload, containing a vct claim with the value https://betelgeuse.example.com/education_credential:

```
{
  "vct": "https://betelgeuse.example.com/education_credential",
  "vct#integrity": "sha256-WRL5ca/xGgX3c1VLmXfh+9cLlJNXN+TsMk+PmKjZ5t0=",
  ...
}
```

Type Metadata for the type https://betelgeuse.example.com/education_credential can be retrieved using various mechanisms as described in Section 6.3. For this example, the vct value is a URL as defined in Section 6.3.1 and the following Type Metadata Document is retrieved from it:

```
{
  "vct":"https://betelgeuse.example.com/education_credential",
  "name":"Betelgeuse Education Credential - Preliminary Version",
  "description":"This is our development version of the education credential. Don’t pa
nic.",
  "extends":"https://galaxy.example.com/galactic-education-credential-0.9",
  "extends#integrity":"sha256-ilOUJsTultOwLfz7QUcFALaRa3BP/jelX1ds04kB9yU=",
  "schema_uri":"https://exampleuniversity.com/public/credential-schema-0.9",
  "schema_uri#integrity":"sha256-He4fNeA4xvjLbh/e+rd9Hw3l60OS4tEliHE7NDYXRwA="
}
```

This example is shortened for presentation, a full Type Metadata example can be found in Appendix B.2.

Note: The hash of the Type Metadata document shown in the second example must be equal to the one in the vct#integrity claim in the SD-JWT VC payload, WRL5ca/xGgX3c1VLmXfh+9cLlJNXN+TsMk+PmKjZ5t0=.

6.2. Type Metadata Format

The Type Metadata document MUST be a JSON object. The following properties are defined:

* name
  - OPTIONAL. A human-readable name for the type, intended for developers reading the JSON document.
* description
  - OPTIONAL. A human-readable description for the type, intended for developers reading the JSON document.
* extends
  - OPTIONAL. A URI of another type that this type extends, as described in Section 6.4.
* display: An array of objects containing display information for the type, as described in Section 8. This property is OPTIONAL.
* claims: An array of objects containing claim information for the type, as described in Section 9. This property is OPTIONAL.
* schema
  - OPTIONAL. An embedded JSON Schema document describing the structure of the Verifiable Credential as described in Section 6.5.1. schema MUST NOT be used if schema_uri is present.
* schema_uri
  - OPTIONAL. A URL pointing to a JSON Schema document describing the structure of the Verifiable Credential as described in Section 6.5.1. schema_uri MUST NOT be used if schema is present.

An example of a Type Metadata document is shown in Appendix B.2.

6.3. Retrieving Type Metadata

6.3.1. From a URL in the vct Claim

A URI in the vct claim can be used to express a type. If the type is a URL using the HTTPS scheme, Type Metadata MAY be retrieved from it.

The Type Metadata is retrieved using the HTTP GET method. The response MUST be a JSON object as defined in Section 6.2.

If the claim vct#integrity is present in the SD-JWT VC, its value vct#integrity MUST be an "integrity metadata" string as defined in Section Section 7.

6.3.2. From a Registry

A Consumer MAY use a registry to retrieve Type Metadata for a SD-JWT VC type, e.g., if the type is not an HTTPS URL or if the Consumer does not have access to the URL. The registry MUST be a trusted registry, i.e., the Consumer MUST trust the registry to provide correct Type Metadata for the type.

The registry MUST provide the Type Metadata in the same format as described in Section 6.2.

6.3.3. Using a Defined Retrieval Method

Ecosystems MAY define additional methods for retrieving Type Metadata. For example, a standardization body or a community MAY define a service which has to be used to retrieve Type Metadata based on a URN in the vct claim.

6.3.4. From a Local Cache

A Consumer MAY cache Type Metadata for a SD-JWT VC type. If a hash for integrity protection is present in the Type Metadata as defined in Section 7, the Consumer MAY assume that the Type Metadata is static and can be cached indefinitely. Otherwise, the Consumer MUST use the Cache-Control header of the HTTP response to determine how long the metadata can be cached.

6.3.5. From Type Metadata Glue Documents

Credentials MAY encode Type Metadata directly, providing it as "glue information" to the Consumer.

For JSON-serialized JWS-based credentials, such Type Metadata documents MAY be included in the unprotected header of the JWS. In this case, the key vctm MUST be used in the unprotected header and its value MUST be an array of base64url-encoded Type Metadata documents as defined in this specification. Multiple documents MAY be included for providing a whole chain of types to the Consumer (see Section 6.4).

A Consumer of a credential MAY use the documents in the vctm array instead of retrieving the respective Type Metadata elsewhere as follows:

* When resolving a vct in a credential, the Consumer MUST ensure that the vct claim in the credential matches the one in the Type Metadata document, and it MUST verify the integrity of the Type Metadata document as defined in Section 7. The Consumer MUST NOT use the Type Metadata if no hash for integrity protection was provided in vct#integrity.
* When resolving an extends property in a Type Metadata document, the Consumer MUST ensure that the value of the extends property in the Type Metadata document matches that of the vct in the Type Metadata document, and it MUST verify the integrity of the Type Metadata document as defined in Section 7. The Consumer MUST NOT use the Type Metadata if no hash for integrity protection was provided.

6.4. Extending Type Metadata

An SD-JWT VC type can extend another type. The extended type is identified by the URI in the extends property. Consumers MUST retrieve and process Type Metadata for the extended type before processing the Type Metadata for the extending type.

The extended type MAY itself extend another type. This can be used to create a chain or hierarchy of types. The security considerations described in Section 10.3 apply in order to avoid problems with circular dependencies.

6.5. Schema Type Metadata

6.5.1. Schema Definition

Schemas for Verifiable Credentials are contained in the schema or retrieved via the schema_uri Type Metadata parameters (as defined in Section 6.2). A schema MUST be represented by a JSON Schema document according to draft version 2020-12 [JSON.SCHEMA.2020-12] or above.

The schema of a Verifiable Credential MUST include all properties that are required by this specification and MUST NOT override their cardinality, JSON data type, or semantic intent.

The following is a non-normative example of a JSON Schema document for the example in Section 3.3 requiring the presence of the cnf claim in an SD-JWT VC presentation:

```
{
  "$schema":"https://json-schema.org/draft/2020-12/schema",
  "type":"object",
  "properties":{
    "vct":{
      "type":"string"
    },
    "iss":{
      "type":"string"
    },
    "nbf":{
      "type":"number"
    },
    "exp":{
      "type":"number"
    },
    "cnf":{
      "type":"object"
    },
    "status":{
      "type":"object"
    },
    "given_name":{
      "type":"string"
    },
    "family_name":{
      "type":"string"
    },
    "email":{
      "type":"string"
    },
    "phone_number":{
      "type":"string"
    },
    "address":{
      "type":"object",
      "properties":{
        "street_address":{
          "type":"string"
        },
        "locality":{
          "type":"string"
        },
        "region":{
          "type":"string"
        },
        "country":{
          "type":"string"
```

```
            }
          }
        },
        "birthdate":{
          "type":"string"
        },
        "is_over_18":{
          "type":"boolean"
        },
        "is_over_21":{
          "type":"boolean"
        },
        "is_over_65":{
          "type":"boolean"
        }
      },
      "required":[
        "iss",
        "vct",
        "cnf"
      ]
    }
```

Note that iss and vct are always required by this specification.

6.5.2. Schema Validation

If a schema or schema_uri property is present, a Consumer MUST validate the Processed SD-JWT Payload JSON document resulting from the SD-JWT verification algorithm (as defined in Section 7.3 of [I-D.ietf-oauth-selective-disclosure-jwt]) against the JSON Schema document provided by the schema or schema_uri property.

If an extends property is present, the schema of the extended type MUST also be validated in the same manner. This process includes validating all subsequent extended types recursively until a type is encountered that does not contain an extends property in its Type Metadata. Each schema in this chain MUST be evaluated for a specific Verifiable Credential.

If the schema validation fails for any of the types in the chain, the Consumer MUST reject the Verifiable Credential.

The following is a non-normative example of a result JSON document after executing the SD-JWT verification algorithm that is validated against the JSON Schema document in the example provided in Section 6.5.1:

```
{
  "vct":"https://credentials.example.com/identity_credential",
  "iss":"https://example.com/issuer",
  "iat":1683000000,
  "exp":1883000000,
  "sub":"6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "address":{
    "country":"DE"
  },
  "cnf":{
    "jwk":{
      "kty":"EC",
      "crv":"P-256",
      "x":"TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y":"ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```

Note, the example above does not contain any _sd_alg, _sd, or ... claims.

7. Document Integrity

Both the vct claim in the SD-JWT VC and the various URIs in the Type Metadata MAY be accompanied by a respective claim suffixed with #integrity, in particular:

* vct as defined in Section 3.2.2.2,
* extends as defined in Section 6.4
* uri as used in two places in Section 8.1
* schema_uri as defined in Section 6.5

The value MUST be an "integrity metadata" string as defined in Section 3 of [W3C.SRI]. A Consumer of the respective documents MUST verify the integrity of the retrieved document as defined in Section 3.3.5 of [W3C.SRI].

8. Display Metadata

The display property is an array containing display information for the type. The array MUST contain an object for each language that is supported by the type. The consuming application MUST use the language tag it considers most appropriate for the user.

The objects in the array have the following properties:

* lang: A language tag as defined in Section 2 of [RFC5646]. This property is REQUIRED.
* name: A human-readable name for the type, intended for end users. This property is REQUIRED.
* description: A human-readable description for the type, intended for end users. This property is OPTIONAL.
* rendering: An object containing rendering information for the type, as described in Section 8.1. This property is OPTIONAL.

8.1. Rendering Metadata

The rendering property is an object containing rendering information for the type. The object MUST contain a property for each rendering method that is supported by the type. The property name MUST be a rendering method identifier and the property value MUST be an object containing the properties defined for the rendering method.

8.1.1. Rendering Method "simple"

The simple rendering method is intended for use in applications that do not support SVG rendering. The object contains the following properties:

* logo: An object containing information about the logo to be displayed for the type, as described in Section 8.1.1.1. This property is OPTIONAL.
* background_color: An RGB color value as defined in [W3C.CSS-COLOR] for the background of the credential. This property is OPTIONAL.
* text_color: An RGB color value as defined in [W3C.CSS-COLOR] value for the text of the credential. This property is OPTIONAL.

8.1.1.1. Logo Metadata

The logo property is an object containing information about the logo to be displayed for the type. The object contains the following properties:

* uri: A URI pointing to the logo image. This property is REQUIRED.
* uri#integrity: An "integrity metadata" string as described in Section 7. This property is OPTIONAL.
* alt_text: A string containing alternative text for the logo image. This property is OPTIONAL.

8.1.2. Rendering Method "svg_template"

The svg_template rendering method is intended for use in applications that support SVG rendering. The object MUST contain an array of objects containing information about the SVG templates available for the type. Each object contains the following properties:

* uri: A URI pointing to the SVG template. This property is REQUIRED.
* uri#integrity: An "integrity metadata" string as described in Section 7. This property is OPTIONAL.
* properties: An object containing properties for the SVG template, as described in Section 8.1.2.1. This property is REQUIRED if more than one SVG template is present, otherwise it is OPTIONAL.

8.1.2.1. SVG Template Properties

The properties property is an object containing properties for the SVG template. Consuming applications MUST use these properties to find the best SVG template available for display to the user based on the display properties (landscape/portrait) and user preferences (color scheme, contrast). The object MUST contain at least one of the following properties:

* orientation: The orientation for which the SVG template is optimized, with valid values being portrait and landscape. This property is OPTIONAL.
* color_scheme: The color scheme for which the SVG template is optimized, with valid values being light and dark. This property is OPTIONAL.
* contrast: The contrast for which the SVG template is optimized, with valid values being normal and high. This property is OPTIONAL.

8.1.2.2. SVG Rendering

Consuming application MUST preprocess the SVG template by replacing placeholders in the SVG template with properly escaped values of the claims in the credential. The placeholders MUST be defined in the SVG template using the syntax {{svg_id}}, where svg_id is an identifier defined in the claim metadata as described in Section 9.

Placeholders MUST only be used in the text content of the SVG template and MUST NOT be used in any other part of the SVG template, e.g., in attributes or comments.

A consuming application MUST ensure that all special characters in the claim values are properly escaped before inserting them into the SVG template. At least the following characters MUST be escaped:

* & as &
* < as <
* > as >
* " as "
* ' as '

If the svg_id is not present in the claim metadata, the consuming application SHOULD reject not render the SVG template. If the svg_id is present in the claim metadata, but the claim is not present in the credential, the placeholder MUST be replaced with an empty string or a string appropriate to indicate that the value is absent.

The following non-normative example shows a minimal SVG with one placeholder using the svg_id value address_street_address which is defined in the example in Appendix B.2:

```
<svg xmlns="http://www.w3.org/2000/svg" width="100" height="100">
  <text x="10" y="20">Street address: {{address_street_address}}</text>
</svg>
```

When rendering the SVG template, the consuming application MUST ensure that malicious schema providers or issuers cannot inject executable code into the SVG template and thereby compromise the security of the consuming application. The consuming application MUST NOT execute any code in the SVG template. If code execution cannot be prevented reliably, the SVG display MUST be sandboxed.

9. Claim Metadata

The claims property is an array of objects containing information about particular claims for displaying and validating the claims.

The array MAY contain an object for each claim that is supported by the type. Each object contains the following properties:

* path: An array indicating the claim or claims that are being addressed, as described below. This property is REQUIRED.
* display: An object containing display information for the claim, as described in Section 9.2. This property is OPTIONAL.
* sd: A string indicating whether the claim is selectively disclosable, as described in Section 9.3. This property is OPTIONAL.

* svg_id: A string defining the ID of the claim for reference in the SVG template, as described in Section 8.1.2.2. The ID MUST be unique within the type metadata. It MUST consist of only alphanumeric characters and underscores and MUST NOT start with a digit. This property is OPTIONAL.

9.1. Claim Path

The path property MUST be a non-empty array of strings, null values, or non-negative integers. It is used to select a particular claim in the credential or a set of claims. A string indicates that the respective key is to be selected, a null value indicates that all elements of the currently selected array(s) are to be selected, and a non-negative integer indicates that the respective index in an array is to be selected.

The following shows a non-normative, reduced example of a credential:

```
{
  "vct": "https://betelgeuse.example.com/education_credential",
  "name": "Arthur Dent",
  "address": {
    "street_address": "42 Market Street",
    "city": "Milliways",
    "postal_code": "12345"
  },
  "degrees": [
    {
      "type": "Bachelor of Science",
      "university": "University of Betelgeuse"
    },
    {
      "type": "Master of Science",
      "university": "University of Betelgeuse"
    }
  ],
  "nationalities": ["British", "Betelgeusian"]
}
```

The following shows examples of path values and the respective selected claims in the credential above:

* ["name"]: The claim name with the value Arthur Dent is selected.
* ["address"]: The claim address with its sub-claims as the value is selected.
* ["address", "street_address"]: The claim street_address with the value 42 Market Street is selected.

* ["degrees", null, "type"]: All type claims in the degrees array are selected.

In detail, the array is processed from left to right as follows:

1. Select the root element of the credential, i.e., the top-level JSON object.
2. Process the path components from left to right:
    1. If the path component is a string, select the element in the respective key in the currently selected element(s). If any of the currently selected element(s) is not an object, abort processing and return an error. If the key does not exist in any element currently selected, remove that element from the selection.
    2. If the path component is null, select all elements of the currently selected array(s). If any of the currently selected element(s) is not an array, abort processing and return an error.
    3. If the path component is a non-negative integer, select the element at the respective index in the currently selected array(s). If any of the currently selected element(s) is not an array, abort processing and return an error. If the index does not exist in a selected array, remove that array from the selection.
    4. If the set of elements currently selected is empty, abort processing and return an error.

The result of the processing is the set of elements to which the respective claim metadata applies.

The path property MUST point to the respective claim as if all selectively disclosable claims were disclosed to a Verifier. That means that a consuming application which does not have access to all disclosures may not be able to identify the claim which is being addressed.

Note: This specification intentionally does not use JSON Pointer [RFC6901] for selecting claims, as JSON Pointer requires string parsing and does not support wildcard selection of array elements. It does not use JSON Path [RFC9535] as that introduces a considerable complexity and brings in many features which are not needed for the use case of selecting claims in a credential. There are also security concerns with some implementations.

9.2. Claim Display Metadata

The display property is an array containing display information for the claim. The array MUST contain an object for each language that is supported by the type. The consuming application MUST use the language tag it considers most appropriate for the user.

The objects in the array have the following properties:

* lang: A language tag as defined in Section 2 of [RFC5646]. This property is REQUIRED.
* label: A human-readable label for the claim, intended for end users. This property is REQUIRED.
* description: A human-readable description for the claim, intended for end users. This property is OPTIONAL.

9.3. Claim Selective Disclosure Metadata

The sd property is a string indicating whether the claim is selectively disclosable. The following values are defined:

* always: The Issuer MUST make the claim selectively disclosable.
* allowed: The Issuer MAY make the claim selectively disclosable.
* never: The Issuer MUST NOT make the claim selectively disclosable.

If omitted, the default value is allowed.

10. Security Considerations

The Security Considerations in the SD-JWT specification [I-D.ietf-oauth-selective-disclosure-jwt] apply to this specification. Additionally, the following security considerations need to be taken into account when using SD-JWT VCs:

10.1. Server-Side Request Forgery

The JWT VC Issuer Metadata configuration is retrieved from the JWT VC Issuer by the Holder or Verifier. Similar to other metadata endpoints, the URL for the retrieval MUST be considered an untrusted value and could be a vector for Server-Side Request Forgery (SSRF) attacks.

Before making a request to the JWT VC Issuer Metadata endpoint, the Holder or Verifier MUST validate the URL to ensure that it is a valid HTTPS URL and that it does not point to internal resources. This requires, in particular, ensuring that the host part of the URL does not address an internal service (by IP address or an internal host name) and that, if an external DNS name is used, the resolved DNS name does not point to an internal IPv4 or IPv6 address.

When retrieving the metadata, the Holder or Verifier MUST ensure that the request is made in a time-bound and size-bound manner to prevent denial of service attacks. The Holder or Verifier MUST also ensure that the response is a valid JWT VC Issuer Metadata configuration document before processing it.

Additional considerations can be found in [OWASP_SSRF].

10.2. Ecosystem-specific Public Key Verification Methods

When defining ecosystem-specific rules for resolution and verification of the public key, as outlined in Section 3.5, it is critical that those rules maintain the integrity of the relationship between the iss value of the SD-JWT, if present, and the public keys of the Issuer.

It MUST be ensured that for any given iss value, an attacker cannot influence the type of verification process used. Otherwise, an attacker could attempt to make the Verifier use a verification process not intended by the Issuer, allowing the attacker to potentially manipulate the verification result to their advantage.

10.3. Circular "extends" Dependencies of Types

A type MUST NOT extend another type that extends (either directly or with steps in-between) the first type. This would result in a circular dependency that could lead to infinite recursion when retrieving and processing the metadata.

Consumers MUST detect such circular dependencies and reject the credential.

10.4. Robust Retrieval of Type Metadata

In Section 6.3, various methods for distributing and retrieving metadata are described. Methods relying on a network connection may fail due to network issues or unavailability of a network connection due to offline usage of credentials, temporary server outages, or denial-of-service attacks on the metadata server.

Consumers SHOULD therefore implement a local cache as described in Section 6.3.4 if possible. Such a cache MAY be populated with metadata before the credential is used.

Issuers MAY provide glue documents as described in Section 6.3.5 to provide metadata directly with the credential and avoid the need for network requests.

These measures allow the Consumers to continue to function even if the metadata server is temporarily unavailable and avoid privacy issues as described in Section 12.1.

10.5. Risks Associated with Textual Information

Some claims in the SD-JWT VC and properties in the Type Metadata, e.g., display, allows issuers and providers of metadata to specify human-readable information. These can contain arbitrary textual information that may be displayed to developers. As such, any consuming application MUST ensure that maliciously crafted information cannot be used to compromise the security of the application or the privacy of the user. To this end, the following considerations apply:

* The consuming application MUST ensure that the text is properly escaped before displaying it to the user or transferring it into other contexts. For example, if the data is displayed in an HTML document, the text MUST be properly escaped to prevent Cross-Site Scripting (XSS) attacks.
* The consuming application MUST ensure that the display of the user interface elements cannot be distorted by overly long text or special characters.

11. Privacy Considerations

The Privacy Considerations in the SD-JWT specification [I-D.ietf-oauth-selective-disclosure-jwt] apply to this specification. Additionally, the following privacy considerations need to be taken into account when using SD-JWT VCs.

11.1. Unlinkability

The Privacy Considerations in Section 10.1 of [I-D.ietf-oauth-selective-disclosure-jwt] apply especially to the cnf claim.

11.2. Verifiable Credential Type Identifier

Issuers and Holders have to be aware that while this specification supports selective disclosure of claims of a given SD-JWT VC, the vct claim is not selectively disclosable. In certain situations this could lead to unwanted leakage of additional context information.

In general, Issuers are advised to choose vct values following data minimization principles. For example, government Issuers issuing an SD-JWT VC to their citizens to enable them to prove their age, might consider using a vct value that does not allow third-parties to infer additional personal information about the Holder, e.g., country of residency or citizenship.

Additionally, Holders have to be informed that, besides the actual requested claims, the vct information is shared with the Verifier.

11.3. Issuer Phone-Home

A malicious Issuer can choose the Issuer identifier of the SD-JWT VC to enable tracking the usage behavior of the Holder if the Issuer identifier is Holder-specific and if the resolution of the key material to verify the Issuer-signed JWT requires the Verifier to phone home to the Issuer.

For example, a malicious Issuer could generate a unique value for the Issuer identifier per Holder, e.g., https://example.com/issuer/holder-1234 and host the JWT VC Issuer Metadata. The Verifier would create a HTTPS GET request to the Holder-specific well-known URI when the SD-JWT VC is verified. This would allow the malicious Issuer to keep track where and how often the SD-JWT VC was used.

Verifiers are advised to establish trust in an SD-JWT VC by pinning specific Issuer identifiers and should monitor suspicious behaviour such as frequently rotating Issuer identifiers. If such behaviour was detected, Verifiers are advised to reject SD-JWT VCs issued by such Issuers.

Holders are advised to reject SD-JWT VCs if they contain easily correlatable information in the Issuer identifier.

12. Relationships to Other Documents

This specification defines validation and processing rules for verifiable credentials using JSON payloads and secured by SD-JWT [I-D.ietf-oauth-selective-disclosure-jwt]. Other specifications exist that define their own verifiable credential formats; for example, W3C Verifiable Credential Data Model (VCDM) 2.0 [W3C.VCDM] defines a data model for verifiable credentials encoded as JSON-LD, and ISO/IEC 18013-5:2021 [ISO.18013-5] defines a representation of verifiable credentials in the mobile document (mdoc) format encoded as CBOR and secured using COSE.

12.1. Privacy-Preserving Retrieval of Type Metadata

In Section 6.3, various methods for distributing and retrieving Type Metadata are described. For methods which rely on a network connection to a URL (e.g., provided by an Issuer), third parties (like the Issuer) may be able to track the usage of a credential by observing requests to the Type Metadata URL.

Consumers SHOULD prefer methods for retrieving Type Metadata that do not leak information about the usage of a credential to third parties. The recommendations in Section 10.4 apply.

13. References

13.1. Normative References

[I-D.ietf-oauth-selective-disclosure-jwt]
Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JWTs (SD-JWT)", Work in Progress, Internet-Draft, draft-ietf-oauth-selective-disclosure-jwt-22, 29 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-selective-disclosure-jwt-22>.

[I-D.ietf-oauth-status-list]
Looker, T., Bastian, P., and C. Bormann, "Token Status List (TSL)", Work in Progress, Internet-Draft, draft-ietf-oauth-status-list-12, 23 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-status-list-12>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[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>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/info/rfc7800>.

[W3C.CSS-COLOR] テ☒lik, T., Lilley, C., and L. D. Baron, "CSS Color Module Level 3", 18 January 2022, <https://www.w3.org/TR/css-color-3>.

[W3C.SRI] Akhawe, D., Braun, F., Marier, F., and J. Weinberger, "Subresource Integrity", 23 June 2016, <https://www.w3.org/TR/SRI/>.

13.2. Informative References

[EUDIW.ARF] Commission, E., "The European Digital Identity Wallet Architecture and Reference Framework", <https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/releases>.

[IANA.well-known] IANA, "Well-Known URIs", <http://www.iana.org/assignments/well-known-uris>.

[ISO.18013-5] ISO/IEC, "ISO/IEC 18013-5:2021", 1 September 2024, <https://www.iso.org/standard/69084.html>.

[JSON.SCHEMA.2020-12]
Foundation, O., "JSON Schema (2020-12)", <https://json-schema.org/draft/2020-12/release-notes>.

[OWASP_SSRF]
OWASP, "Server Side Request Forgery Prevention Cheat Sheet", <https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html/>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <https://www.rfc-editor.org/info/rfc6901>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC9535] Gテカssner, S., Ed., Normington, G., Ed., and C. Bormann, Ed., "JSONPath: Query Expressions for JSON", RFC 9535, DOI 10.17487/RFC9535, February 2024, <https://www.rfc-editor.org/info/rfc9535>.

[W3C.VCDM] Sporny, M., Longley, D., Chadwick, D., and O. Steele, "Verifiable Credentials Data Model v2.0", 10 February 2024, <https://www.w3.org/TR/vc-data-model-2.0/>.

Appendix A. IANA Considerations

A.1. JSON Web Token Claims Registration

* Claim Name: "vct"
* Claim Description: Verifiable credential type identifier
* Change Controller: IETF
* Specification Document(s): [[Section 3.2.2.1 of this specification]]
* Claim Name: "vct#integrity"
* Claim Description: SD-JWT VC vct claim "integrity metadata" value
* Change Controller: IETF
* Specification Document(s): [[Section 7 of this specification]]

A.2. Media Types Registry

A.2.1. application/dc+sd-jwt

The Internet media type for an SD-JWT VC is application/dc+sd-jwt.

* Type name: application
* Subtype name: dc+sd-jwt
* Required parameters: n/a
* Optional parameters: n/a
* Encoding considerations: 8-bit code points; SD-JWT VC values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') and tilde ('~') characters.
* Security considerations: See Security Considerations in Section 10.
* Interoperability considerations: n/a
* Published specification: [[this specification]]
* Applications that use this media type: Applications that issue, present, and verify SD-JWT-based verifiable credentials.
* Additional information:
  - Magic number(s): n/a
  - File extension(s): n/a
  - Macintosh file type code(s): n/a
* Person & email address to contact for further information: Oliver Terbu oliver.terbu@mattr.global (mailto:oliver.terbu@mattr.global)
* Intended usage: COMMON
* Restrictions on usage: none
* Author: Oliver Terbu oliver.terbu@mattr.global (mailto:oliver.terbu@mattr.global)
* Change controller: IETF

A.3. Well-Known URI Registry

This specification requests the well-known URI defined in Section 5 in the IANA "Well-Known URIs" registry [IANA.well-known] established by [RFC5785].

A.3.1. Registry Contents

* URI suffix: jwt-vc-issuer
* Change controller: IETF
* Specification document: [[Section 5 of this specification]]
* Related information: (none)
* Status: permanent

Appendix B. Examples

Important: The following examples are not normative and provided for illustrative purposes only. In particular, neither the structure of the claims nor the selection of selectively disclosable claims are normative.

Line breaks have been added for readability.

B.1. Example 1: Person Identification Data (PID) Credential

This example shows how the artifacts defined in this specification could be used to represent the concept of a Person Identification Data (PID) as defined in the "PID Rulebook" in [EUDIW.ARF]. This example uses fictional data of a German citizen.

Key Binding is applied using the Holder’s public key passed in a cnf claim in the SD-JWT.

The following data about the citizen comprises the input JWT Claims Set used by the Issuer:

```
{
  "iss": "https://pid-issuer.bund.de.example",
  "vct": "urn:eudi:pid:de:1",
  "given_name": "Erika",
  "family_name": "Mustermann",
  "birthdate": "1963-08-12",
  "address": {
    "street_address": "Heidestraテ夕 17",
    "locality": "Kテカln",
    "postal_code": "51147",
    "country": "DE"
  },
  "nationalities": [
    "DE"
  ],
  "sex": 2,
  "birth_family_name": "Gabler",
  "place_of_birth": {
    "locality": "Berlin",
    "country": "DE"
  },
  "age_equal_or_over": {
    "12": true,
    "14": true,
    "16": true,
    "18": true,
    "21": true,
    "65": false
  },
  "age_in_years": 62,
  "age_birth_year": 1963,
  "issuance_date": "2020-03-11",
  "expiry_date": "2030-03-12",
  "issuing_authority": "DE",
  "issuing_country": "DE"
}
```


The following is the issued SD-JWT:


```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCJ9.eyJfc2QiOiBbIjBIWm1
uU0lQejMzN2tTV2U3QzM0bC0tODhnekppLWVCSjJWel9ISndBVGciLCAiMUNybjAzV21
VZVJXcDR6d1B2dkNLWGw5WmFRcC1jZFFWX2dIZGFHU1dvdyIsICIycjAwOWR6dkh1VnJ
XclJYVDVrSk1tSG5xRUhIbldlME1MVlp3OFBBVEI4IiwgIjZaTklTRHN0NjJ5bWxyT0F
rYWRqZEQ1WnVsVDVBMjk5Sjc4U0xoTV9fT3MiLCAiNzhqZzc3LUdZQmVYOElRZm9FTFB
5TDBEWVBkbWZabzBKZ1ZpVjBfbEtDTSIsICI5MENUOEFhQlBibjVYOG5SWGtlc2p1MWk
wQnFoV3FaM3dxRDRqRi1xREdrIiwgIkkwMGZjRlVvRFhDdWNwNXl5MnVqcVBzc0RWR2F
XTmlVbGlOel9hd0QwZ2MiLCAiS2pBWGdBQTlONVdIRUR0UkloNHU1TW4xWnNXaXhoaFd
BaVgtQTRRaXdnQSIsICJMYWk2SVU2ZDdHUWFnWFI3QXZHVHJuWGdTbGQzejhFSWdfZnY
```

zZk9aMVdnIiwgIkxlemphYlJxaVpPWHpFWW1WWmY4Uk1pOXhBa2QzX00xTFo4VTdFNHM
zdTQiLCAiUlR6M3FUbUZOSGJwV3JyT01aUzQxRjQ3NGtGcVJ2M3ZJUHF0aDZQVWhsTSI
sICJXMTRYSGJVZmZ6dVc0SUZNanBTVGIxbWVsV3hVV2Y0Tl9vMmxka2tJcWM4IiwgIld
UcEk3UmNNM2d4WnJ1UnBYemV6U2JrYk9yOTNQVkZ2V3g4d29KM2oxY0UiLCAiX29oSlZ
JUUlCc1U0dXBkTlM0X3c0S2IxTUhxSjBMOXFMR3NoV3E2SlhRcyIsICJ5NTBjemMwSVN
DaHlfYnNiYTFkTW9VdUFPUTVBTW1PU2ZHb0VlODF2MUZVIl0sICJpc3MiOiAiaHR0cHM
6Ly9waWQtaXNzdWVyLmJ1bmQuZGUuZXhhbXBsZSIsICJpYXQiOiAxNjgzMDAwMDAwLCA
iZXhwIjogMTg4MzAwMDAwMCwgInZjdCI6ICJ1cm46ZXVkaTpwaWQ6ZGU6MSIsICJfc2R
fYWxnIjogInNoYS0yNTYiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJFQyIsICJjcnY
iOiAiUC0yNTYiLCAieCI6ICJUQ0FFUjE5WnZ1M09IRjRqNFc0dmZTVm9ISVAxSUxpbER
sczd2Q2VHZW1jIiwgInkiOiAiWnhqaVdXYlpNUUdIVldLVlE0aGJTSWlyc1ZmdWVjQ0U
2dDRqVDlGMkhaUSJ9fX0.M-XsTq2IhTck79HoxaHgbajJUSSGq7Hxlph1emFy0AhBjuQ
9XGAD3KUaYuqeOFMKKlVmMSVxTo2g_BcJsBDtAA~WyIyR0xDNDJzS1F2ZUNmR2ZyeU5S
Tjl3IiwgImdpdmVuX25hbWUiLCAiRXJpa2EiXQ~WyJlbHVWNU9nM2dTTklJOEVZbnN4Q
V9BIiwgImZhbWlseV9uYW1lIiwgIk11c3Rlcm1hbm4iXQ~WyI2SWo3dE0tYTVpVlBHYm
9TNXRtdlZBIiwgImJpcnRoZGF0ZSIsICIxOTYzLTA4LTEyIl0~WyJlSThaV205UW5LUH
BOUGVOZW5IZGhRIiwgInN0cmVldF9hZGRyZXNzIiwgIkhlaWRlc3RyYVx1MDBkZmUgMT
ciXQ~WyJRZ19PNjR6cUF4ZTQxMmExMDhpcm9BIiwgImxvY2FsaXR5IiwgIktcdTAwZjZ
sbiJd~WyJBSngtMDk1VlBycFR0TjRRTU9xUk9BIiwgInBvc3RhbF9jb2RlIiwgIjUxMT
Q3Il0~WyJQYzMzSk0yTGNoY1VfbEhnZ3ZfdWZRIiwgImNvdW50cnkiLCAiREUiXQ~WyJ
HMDJOU3JRZmpGWFE3SW8wOXN5YWpBIiwgImFkZHJlc3MiLCB7Il9zZCI6IFsiQUxaRVJ
zU241V05pRVhkQ2tzVzhJNXFRdzNfTnBBblJxcFNBWkR1ZGd3OCIsICJEX19XX3VZY3Z
SejN0dlVuSUp2QkRIaVRjN0NfX3FIZDB4Tkt3SXNfdzlrIiwgImVCcENYVTFKNWRoSDJ
nNHQ4UVlOVzVFeFM5QXhVVmJsVW9kb0xZb1BobzAiLCAieE9QeTktZ0pBTEs2VWJXS0Z
MUjg1Y09CeVVEM0FiTndGZzNJM1lmUUVfSSJdfV0~WyJsa2x4RjVqTVlsR1RQVW92TU5
JdkNBIiwgIm5hdGlvbmFsaXRpZXMiLCBbIkRFIl1d~WyJuUHVvUW5rUkZxM0JJZUFtN0
FuWEZBIiwgInNleCIsIDJd~WyI1YlBzMUlxdVpOYTBoa2FGenp6Wk53IiwgImJpcnRoX
2ZhbWlseV9uYW1lIiwgIkdhYmxlciJd~WyI1YTJXMF9OcmxFWnpmcW1rXzdQcS13Iiwg
ImxvY2FsaXR5IiwgIkJlcmxpbiJd~WyJ5MXNWVTV3ZGZKYWhWZGd3UGdTN1JRIiwgImN
vdW50cnkiLCAiREUiXQ~WyJIYlE0WDhzclZXM1FEeG5JSmRxeU9BIiwgInBsYWNlX29m
X2JpcnRoIiwgeyJfc2QiOiBbIktVVmlhYUxuWTVqU01MOTBHMjlPT0xFTlBiYlhmaFNq
U1BNalphR2t4QUUiLCAiWWJzVDBTNzZWcVhDVnNkMWpVU2x3S1BEZ21BTGVCMXVaY2xG
SFhmLVVTUSJdfV0~WyJDOUdTb3VqdmlKcXVFZ1lmb2pDYjFBIiwgIjEyIiwgdHJ1ZV0~
WyJreDVrRjE3Vi14MEptd1V4OXZndnR3IiwgIjE0IiwgdHJ1ZV0~WyJIM28xdXN3UDc2
MEZpMnllR2RWQ0VRIiwgIjE2IiwgdHJ1ZV0~WyJPQktsVFZsdkxnLUFkd3FZR2JQOFpB
IiwgIjE4IiwgdHJ1ZV0~WyJNMEpiNTd0NDF1YnJrU3V5ckRUM3hBIiwgIjIxIiwgdHJ1
ZV0~WyJEc210S05ncFY0ZEFIcGpyY2Fvc0F3IiwgIjY1IiwgZmFsc2Vd~WyJlSzVvNXB
IZmd1cFBwbHRqMXFoQUp3IiwgImFnZV9lcXVhbF9vcl9vdmVyIiwgeyJfc2QiOiBbIjF
0RWl5elBSWU9Lc2Y3U3NZR01nUFpLc09UMWxRWlJ4SFhBMHI1X0J3a2siLCAiQ1ZLbmx
5NVA5MHlKczNFd3R4UWlPdFVjemFYQ1lOQTRJY3pSYW9ock1EZyIsICJhNDQtZzJHcjh
fM0FtSncyWFo4a0kxeTBRel96ZTlpT2NXMlczUkxwWEdnIiwgImdrdnkwRnV2QkJ2ajB
oczJaTnd4Y3FPbGY4bXUyLWtDRTctTmIyUXh1QlUiLCAiaHJZNEhubUY1YjVKd0M5ZVR
6YUZDVWNlSVFBYUlkaHJxVVhRTkNXYmZaSSIsICJ5NlNGclZGUnlxNTBJYlJKdmlUWnF
xalFXejB0TGl1Q21NZU8wS3FhekdJIl19XQ~WyJqN0FEZGIwVVZiMExpMGNpUGNQMGV3
IiwgImFnZV9pbl95ZWFycyIsIDYyXQ~WyJXcHhCkZ1WDh1U2kycDRodDA5anZ3IiwgI
mFnZV9iaXJ0aF95ZWFyIiwgMTk2M10~WyJhdFNtRkFDWU1iSlZLRDA1bzNKZ3RRIiwgI
mlzc3VhbmNlX2RhdGUiLCAiMjAyMC0wMy0xMSJd~WyI0S3lSMzJvSVp0LXprV3ZGcWJV
TEtnIiwgImV4cGlyeV9kYXRlIiwgIjIwMzAtMDMtMTIiXQ~WyJjaEJDc3loeWgtSjg2S

```
S1hd1FEaUNRIiwgImlzc3VpbmdfYXV0aG9yaXR5IiwgIkRFIl0~WyJmbE5QMW5jTXo5T
GctYzlxTUl6XzlnIiwgImlzc3VpbmdfY291bnRyeSIsICJERSJd~
```

This is the payload of that SD-JWT:

```
{
  "_sd": [
    "0HZmnSIPz337kSWe7C34l--88gzJi-eBJ2Vz_HJwATg",
    "1Crn03WmUeRWp4zwPvvCKXl9ZaQp-cdQV_gHdaGSWow",
    "2r009dzvHuVrWrRXT5kJMmHnqEHHnWe0MLVZw8PATB8",
    "6ZNISDst62ymlrOAkadjdD5ZulT5A299J78SLhM__Os",
    "78jg77-GYBeX8IQfoELPyL0DYPdmfZo0JgViV0_lKCM",
    "90CT8AaBPbn5X8nRXkesju1i0BqhWqZ3wqD4jF-qDGk",
    "I00fcFUoDXCucp5yy2ujqPssDVGaWNiUliNz_awD0gc",
    "KjAXgAA9N5WHEDtRIh4u5Mn1ZsWixhhWAiX-A4QiwgA",
    "Lai6IU6d7GQagXR7AvGTrnXgSld3z8EIg_fv3fOZ1Wg",
    "LezjabRqiZOXzEYmVZf8RMi9xAkd3_M1LZ8U7E4s3u4",
    "RTz3qTmFNHbpWrrOMZS41F474kFqRv3vIPqth6PUhlM",
    "W14XHbUffzuW4IFMjpSTb1melWxUWf4N_o2ldkkIqc8",
    "WTpI7RcM3gxZruRpXzezSbkbOr93PVFvWx8woJ3j1cE",
    "_ohJVIQIBsU4updNS4_w4Kb1MHqJ0L9qLGshWq6JXQs",
    "y50czc0ISChy_bsba1dMoUuAOQ5AMmOSfGoEe81v1FU"
  ],
  "iss": "https://pid-issuer.bund.de.example",
  "iat": 1683000000,
  "exp": 1883000000,
  "vct": "urn:eudi:pid:de:1",
  "_sd_alg": "sha-256",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```

The digests in the SD-JWT payload reference the following Disclosures:

Claim given_name:

* SHA-256 Hash: 0HZmnSIPz337kSWe7C34l--88gzJi-eBJ2Vz_HJwATg
* Disclosure:
  WyIyR0xDNDJzS1F2ZUNmR2ZyeU5STjl3IiwgImdpdmVuX25hbWUiLCAiRXJp
  a2EiXQ
* Contents: ["2GLC42sKQveCfGfryNRN9w", "given_name", "Erika"]

Claim family_name:

* SHA-256 Hash: I00fcFUoDXCucp5yy2ujqPssDVGaWNiUliNz_awD0gc
* Disclosure:
  WyJlbHVWNU9nM2dTTklJOEVZbnN4QV9BIiwgImZhbWlseV9uYW1lIiwgIk11
  c3Rlcm1hbm4iXQ
* Contents: ["eluV5Og3gSNII8EYnsxA_A", "family_name", "Mustermann"]

Claim birthdate:

* SHA-256 Hash: Lai6IU6d7GQagXR7AvGTrnXgSld3z8EIg_fv3fOZ1Wg
* Disclosure:
  WyI2SWo3dE0tYTVpVlBHYm9TNXRtdlZBIiwgImJpcnRoZGF0ZSIsICIxOTYz
  LTA4LTEyIl0
* Contents: ["6Ij7tM-a5iVPGboS5tmvVA", "birthdate", "1963-08-12"]

Claim street_address:

* SHA-256 Hash: ALZERsSn5WNiEXdCksW8I5qQw3_NpAnRqpSAZDudgw8
* Disclosure:
  WyJlSThaV205UW5LUHBOUGVOZW5IZGhRIiwgInN0cmVldF9hZGRyZXNzIiwg
  IkhlaWRlc3RyYVx1MDBkZmUgMTciXQ
* Contents: ["eI8ZWm9QnKPpNPeNenHdhQ", "street_address",
  "Heidestra\u00dfe 17"]

Claim locality:

* SHA-256 Hash: D__W_uYcvRz3tvUnIJvBDHiTc7C__qHd0xNKwIs_w9k
* Disclosure:
  WyJRZ19PNjR6cUF4ZTQxMmExMDhpcm9BIiwgImxvY2FsaXR5IiwgIktcdTAw
  ZjZsbiJd
* Contents: ["Qg_O64zqAxe412a108iroA", "locality", "K\u00f6ln"]

Claim postal_code:

* SHA-256 Hash: xOPy9-gJALK6UbWKFLR85cOByUD3AbNwFg3I3YfQE_I
* Disclosure:
  WyJBSngtMDk1VlBycFR0TjRRTU9xUk9BIiwgInBvc3RhbF9jb2RlIiwgIjUx
  MTQ3Il0
* Contents: ["AJx-095VPrpTtN4QMOqROA", "postal_code", "51147"]

Claim country:

* SHA-256 Hash: eBpCXU1J5dhH2g4t8QYNW5ExS9AxUVblUodoLYoPho0
* Disclosure:
  WyJQYzMzSk0yTGNoY1VfbEhnZ3ZfdWZRIiwgImNvdW50cnkiLCAiREUiXQ
* Contents: ["Pc33JM2LchcU_lHggv_ufQ", "country", "DE"]

Claim address:

* SHA-256 Hash: RTz3qTmFNHbpWrrOMZS41F474kFqRv3vIPqth6PUhlM
* Disclosure:
  WyJHMDJOU3JRZmpGWFE3SW8wOXN5YWpBIiwgImFkZHJlc3MiLCB7Il9zZCI6
  IFsiQUxaRVJzU241V05pRVhkQ2tzVzhJNXFRdzNfTnBBblJxcFNBWkR1ZGd3
  OCIsICJEX19XX3VZY3ZSejN0dlVuSUp2QkRIaVRjN0NfX3FIZDB4Tkt3SXNf
  dzlrIiwgImVCcENYVTFKNWRoSDJnNHQ4UVlOVzVFeFM5QXhVVmJsVW9kb0xZ
  b1BobzAiLCAieE9QeTktZ0pBTEs2VWJXS0ZMUjg1Y09CeVVEM0FiTndGZzNJ
  M1lmUUVfSSJdfV0
* Contents: ["G02NSrQfjFXQ7Io09syajA", "address", {"_sd":
  ["ALZERsSn5WNiEXdCksW8I5qQw3_NpAnRqpSAZDudgw8",
  "D__W_uYcvRz3tvUnIJvBDHiTc7C__qHd0xNKwIs_w9k",
  "eBpCXU1J5dhH2g4t8QYNW5ExS9AxUVblUodoLYoPho0",
  "xOPy9-gJALK6UbWKFLR85cOByUD3AbNwFg3I3YfQE_I"]}]

Claim nationalities:

* SHA-256 Hash: y50czc0ISChy_bsba1dMoUuAOQ5AMmOSfGoEe81v1FU
* Disclosure:
  WyJsa2x4RjVqTVlsR1RQVW92TU5JdkNBIiwgIm5hdGlvbmFsaXRpZXMiLCBb
  IkRFIl1d
* Contents: ["lklxF5jMYlGTPUovMNIvCA", "nationalities", ["DE"]]

Claim sex:

* SHA-256 Hash: 90CT8AaBPbn5X8nRXkesju1i0BqhWqZ3wqD4jF-qDGk
* Disclosure:
  WyJuUHVvUW5rUkZxM0JJZUFtN0FuWEZBIiwgInNleCIsIDJd
* Contents: ["nPuoQnkRFq3BIeAm7AnXFA", "sex", 2]

Claim birth_family_name:

* SHA-256 Hash: KjAXgAA9N5WHEDtRIh4u5Mn1ZsWixhhWAiX-A4QiwgA
* Disclosure:
  WyI1YlBzMUlxdVpOYTBoa2FGenp6Wk53IiwgImJpcnRoX2ZhbWlseV9uYW1l
  IiwgIkdhYmxlciJd
* Contents: ["5bPs1IquZNa0hkaFzzzZNw", "birth_family_name",
  "Gabler"]

Claim locality:

* SHA-256 Hash: KUViaaLnY5jSML90G29OOLENPbbXfhSjSPMjZaGkxAE
* Disclosure:
  WyI1YTJXMF9OcmxFWnpmcW1rXzdQcS13IiwgImxvY2FsaXR5IiwgIkJlcmxp
  biJd
* Contents: ["5a2W0_NrlEZzfqmk_7Pq-w", "locality", "Berlin"]

Claim country:

* SHA-256 Hash: YbsT0S76VqXCVsd1jUSlwKPDgmALeB1uZclFHXf-USQ
* Disclosure:
  WyJ5MXNWVTV3ZGZKYWhWZGd3UGdTN1JRIiwgImNvdW50cnkiLCAiREUiXQ
* Contents: ["y1sVU5wdfJahVdgwPgS7RQ", "country", "DE"]

Claim place_of_birth:

* SHA-256 Hash: 1Crn03WmUeRWp4zwPvvCKXl9ZaQp-cdQV_gHdaGSWow
* Disclosure:
  WyJIYlE0WDhzclZXM1FEeG5JSmRxeU9BIiwgInBsYWNlX29mX2JpcnRoIiwg
  eyJfc2QiOiBbIktVVmlhYUxuWTVqU01MOTBHMjlPT0xFTlBiYlhmaFNqU1BN
  alphR2t4QUUiLCAiWWJzVDBTNzZWcVhDVnNkMWpVU2x3S1BEZ21BTGVCMXVa
  Y2xGSFhmLVVTUSJdfV0
* Contents: ["HbQ4X8srVW3QDxnIJdqyOA", "place_of_birth", {"_sd":
  ["KUViaaLnY5jSML90G29OOLENPbbXfhSjSPMjZaGkxAE",
  "YbsT0S76VqXCVsd1jUSlwKPDgmALeB1uZclFHXf-USQ"]}]

Claim 12:

* SHA-256 Hash: gkvy0FuvBBvj0hs2ZNwxcqOlf8mu2-kCE7-Nb2QxuBU
* Disclosure:
  WyJDOUdTb3VqdmlKcXVFZ1lmb2pDYjFBIiwgIjEyIiwgdHJ1ZV0
* Contents: ["C9GSoujviJquEgYfojCb1A", "12", true]

Claim 14:

* SHA-256 Hash: y6SFrVFRyq50IbRJviTZqqjQWz0tLiuCmMeO0KqazGI
* Disclosure:
  WyJreDVrRjE3Vi14MEptd1V4OXZndnR3IiwgIjE0IiwgdHJ1ZV0
* Contents: ["kx5kF17V-x0JmwUx9vgvtw", "14", true]

Claim 16:

* SHA-256 Hash: hrY4HnmF5b5JwC9eTzaFCUceIQAaIdhrqUXQNCWbfZI
* Disclosure:
  WyJIM28xdXN3UDc2MEZpMnllR2RWQ0VRIiwgIjE2IiwgdHJ1ZV0
* Contents: ["H3o1uswP760Fi2yeGdVCEQ", "16", true]

Claim 18:

* SHA-256 Hash: CVKnly5P90yJs3EwtxQiOtUczaXCYNA4IczRaohrMDg
* Disclosure:
  WyJPQktsVFZsdkxnLUFkd3FZR2JQOFpBIiwgIjE4IiwgdHJ1ZV0
* Contents: ["OBKlTVlvLg-AdwqYGbP8ZA", "18", true]

Claim 21:

* SHA-256 Hash: 1tEiyzPRYOKsf7SsYGMgPZKsOT1lQZRxHXA0r5_Bwkk
* Disclosure:
  WyJNMEpiNTd0NDF1YnJrU3V5ckRUM3hBIiwgIjIxIiwgdHJ1ZV0
* Contents: ["M0Jb57t41ubrkSuyrDT3xA", "21", true]

Claim 65:

* SHA-256 Hash: a44-g2Gr8_3AmJw2XZ8kI1y0Qz_ze9iOcW2W3RLpXGg
* Disclosure:
  WyJEc210S05ncFY0ZEFIcGpyY2Fvc0F3IiwgIjY1IiwgZmFsc2Vd
* Contents: ["DsmtKNgpV4dAHpjrcaosAw", "65", false]

Claim age_equal_or_over:

* SHA-256 Hash: 2r009dzvHuVrWrRXT5kJMmHnqEHHnWe0MLVZw8PATB8
* Disclosure:
  WyJlSzVvNXBIZmd1cFBwbHRqMXFoQUp3IiwgImFnZV9lcXVhbF9vcl9vdmVy
  IiwgeyJfc2QiOiBbIjF0RWl5elBSWU9Lc2Y3U3NZR01nUFpLc09UMWxRWlJ4
  SFhBMHI1X0J3a2siLCAiQ1ZLbmx5NVA5MHlKczNFd3R4UWlPdFVjemFYQ1lO
  QTRJY3pSYW9ock1EZyIsICJhNDQtZzJHcjhfM0FtSncyWFo4a0kxeTBRel96
  ZTlpT2NXMlczUkxwWEdnIiwgImdrdnkwRnV2QkJ2ajBoczJaTnd4Y3FPbGY4
  bXUyLWtDRTctTmIyUXh1QlUiLCAiaHJZNEhubUY1YjVKd0M5ZVR6YUZDVWNl
  SVFBYUlkaHJxVVhRTkNXYmZaSSIsICJ5NlNGclZGUnlxNTBJYlJKdmlUWnFx
  alFXejB0TGl1Q21NZU8wS3FhekdJIl19XQ
* Contents: ["eK5o5pHfgupPpltj1qhAJw", "age_equal_or_over", {"_sd":
  ["1tEiyzPRYOKsf7SsYGMgPZKsOT1lQZRxHXA0r5_Bwkk",
  "CVKnly5P90yJs3EwtxQiOtUczaXCYNA4IczRaohrMDg",
  "a44-g2Gr8_3AmJw2XZ8kI1y0Qz_ze9iOcW2W3RLpXGg",
  "gkvy0FuvBBvj0hs2ZNwxcqOlf8mu2-kCE7-Nb2QxuBU",
  "hrY4HnmF5b5JwC9eTzaFCUceIQAaIdhrqUXQNCWbfZI",
  "y6SFrVFRyq50IbRJviTZqqjQWz0tLiuCmMeO0KqazGI"]}]

Claim age_in_years:

* SHA-256 Hash: WTpI7RcM3gxZruRpXzezSbkbOr93PVFvWx8woJ3j1cE
* Disclosure:
  WyJqN0FEZGIwVVZiMExpMGNpUGNQMGV3IiwgImFnZV9pbl95ZWFycyIsIDYy
  XQ
* Contents: ["j7ADdb0UVb0Li0ciPcP0ew", "age_in_years", 62]

Claim age_birth_year:

* SHA-256 Hash: LezjabRqiZOXzEYmVZf8RMi9xAkd3_M1LZ8U7E4s3u4
* Disclosure:
  WyJXcHhKckZ1WDh1U2kycDRodDA5anZ3IiwgImFnZV9iaXJ0aF95ZWFyIiwg
  MTk2M10
* Contents: ["WpxJrFuX8uSi2p4ht09jvw", "age_birth_year", 1963]

Claim issuance_date:

* SHA-256 Hash: W14XHbUffzuW4IFMjpSTb1melWxUWf4N_o2ldkkIqc8
* Disclosure:
  WyJhdFNtRkFDWU1iSlZLRDA1bzNKZ3RRIiwgImlzc3VhbmNlX2RhdGUiLCAi
  MjAyMC0wMy0xMSJd
* Contents: ["atSmFACYMbJVKD05o3JgtQ", "issuance_date",
  "2020-03-11"]

Claim expiry_date:

* SHA-256 Hash: 78jg77-GYBeX8IQfoELPyL0DYPdmfZo0JgViV0_lKCM
* Disclosure:
  WyI0S3lSMzJvSVp0LXprV3ZGcWJVTEtnIiwgImV4cGlyeV9kYXRlIiwgIjIw
  MzAtMDMtMTIiXQ
* Contents: ["4KyR32oIZt-zkWvFqbULKg", "expiry_date", "2030-03-12"]

Claim issuing_authority:

* SHA-256 Hash: 6ZNISDst62ymlrOAkadjdD5ZulT5A299J78SLhM__Os
* Disclosure:
  WyJjaEJDc3loeWgtSjg2SS1hd1FEaUNRIiwgImlzc3VpbmdfYXV0aG9yaXR5
  IiwgIkRFIl0
* Contents: ["chBCsyhyh-J86I-awQDiCQ", "issuing_authority", "DE"]

Claim issuing_country:

* SHA-256 Hash: _ohJVIQIBsU4updNS4_w4Kb1MHqJ0L9qLGshWq6JXQs
* Disclosure:
  WyJmbE5QMW5jTXo5TGctYzlxTUl6XzlnIiwgImlzc3VpbmdfY291bnRyeSIs
  ICJERSJd
* Contents: ["flNP1ncMz9Lg-c9qMIz_9g", "issuing_country", "DE"]

This shows a presentation of the SD-JWT with a Key Binding JWT that discloses only nationality and the fact that the person is over 18 years old:

eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCJ9.eyJfc2QiOiBbIjBIWm1
uU0lQejMzN2tTV2U3QzM0bC0tODhnekppLWVCSjJWel9ISndBVGciLCAiMUNybjAzV21
VZVJXcDR6d1B2dkNLWGw5WmFRcC1jZFFWX2dIZGFHU1dvdyIsICIycjAwOWR6dkh1VnJ
XclJYVDVrSk1tSG5xRUhIbldlME1MVlp3OFBBVEI4IiwgIjZaTklTRHN0NjJ5bWxyT0F
rYWRqZEQ1WnVsVDVBMjk5Sjc4U0xoTV9fT3MiLCAiNzhqZzc3LUdZQmVYOElRZm9FTFB
5TDBEWVBkbWZabzBKZ1ZpVjBfbEtDTSIsICI5MENUOEFhQlBibjVYOG5SWGtlc2p1MWk
wQnFoV3FaM3dxRDRqRi1xREdrIiwgIkkwMGZjRlVvRFhDdWNwNXl5MnVqcVBzc0RWR2F
XTmlVbGlOel9hd0QwZ2MiLCAiS2pBWGdBQTlONVdIRUR0UkloNHU1TW4xWnNXaXhoaFd
BaVgtQTRRaXdnQSIsICJMYWk2SVU2ZDdHUWFnWFI3QXZHVHJuWGdTbGQzejhFSWdfZnY
zZk9aMVdnIiwgIkxlemphYlJxaVpPWHpFWW1WWmY4Uk1pOXhBa2QzX00xTFo4VTdFNHM
zdTQiLCAiUlR6M3FUbUZOSGJwV3JyT01aUzQxRjQ3NGtGcVJ2M3ZJUHF0aDZQVWhsTSI
sICJXMTRYSGJVZmZ6dVc0SUZNanBTVGIxbWVsV3hVV2Y0Tl9vMmxka2tJcWM4IiwgIld
UcEk3UmNNM2d4WnJ1UnBYemV6U2JrYk9yOTNQVkZ2V3g4d29KM2oxY0UiLCAiX29oSlZ
JUUlCc1U0dXBkTlM0X3c0S2IxTUhxSjBMOXFMR3NoV3E2SlhRcyIsICJ5NTBjemMwSVN
DaHlfYnNiYTFkTW9VdUFPUTVBTW1PU2ZHb0VlODF2MUZVIl0sICJpc3MiOiAiaHR0cHM
6Ly9waWQtaXNzdWVyLmJ1bmQuZGUuZXhhbXBsZSIsICJpYXQiOiAxNjgzMDAwMDAwLCA
iZXhwIjogMTg4MzAwMDAwMCwgInZjdCI6ICJ1cm46ZXVkaTpwaWQ6ZGU6MSIsICJfc2R
fYWxnIjogInNoYS0yNTYiLCAiY25mIjogeyJqd2siOiB7Imt0eSI6ICJFQyIsICJjcnY
iOiAiUC0yNTYiLCAieCI6ICJUQ0FFUjE5WnZ1M09IRjRqNFc0dmZTVm9ISVAxSUxpbER
sczd2Q2VHZW1jIiwgInkiOiAiWnhqaVdXYlpNUUdIVldLVlE0aGJTSWlyc1ZmdWVjQ0U
2dDRqVDlGMkhaUSJ9fX0.M-XsTq2IhTck79HoxaHgbajJUSSGq7Hxlph1emFy0AhBjuQ
9XGAD3KUaYuqeOFMKKlVmMSVxTo2g_BcJsBDtAA~WyJlSzVvNXBIZmd1cFBwbHRqMXFo
QUp3IiwgImFnZV9lcXVhbF9vcl9vdmVyIiwgeyJfc2QiOiBbIjF0RWl5elBSWU9Lc2Y3
U3NZR01nUFpLc09UMWxRWlJ4SFhBMHI1X0J3a2siLCAiQ1ZLbmx5NVA5MHlKczNFd3R4
UWlPdFVjemFYQ1lOQTRJY3pSYW9ock1EZyIsICJhNDQtZzJHcjhfM0FtSncyWFo4a0kx
eTBRel96ZTlpT2NXMlczUkxwWEdnIiwgImdrdnkwRnV2QkJ2ajBoczJaTnd4Y3FPbGY4
bXUyLWtDRTctTmIyUXh1QlUiLCAiaHJZNEhubUY1YjVKd0M5ZVR6YUZDVWNlSVFBYUlk
aHJxVVhRTkNXYmZaSSIsICJ5NlNGclZGUnlxNTBJYlJKdmlUWnFxalFXejB0TGl1Q21N
ZU8wS3FhekdJIl19XQ~WyJPQktsVFZsdkxnLUFkd3FZR2JQOFpBIiwgIjE4IiwgdHJ1Z
V0~WyJsa2x4RjVqTVlsR1RQVW92TU5JdkNBIiwgIm5hdGlvbmFsaXRpZXMiLCBbIkRFI
l1d~eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImtiK2p3dCJ9.eyJub25jZSI6ICIxMjM
0NTY3ODkwIiwgImF1ZCI6ICJodHRwczovL2V4YW1wbGUuY29tL3ZlcmlmaWVyIiwgIml
hdCI6IDE3NTE5MDg2NzksICJzZF9oYXNoIjogInFvbURKekh4ODg4M3NyQUFBM3JTVXh
EU2FXelBDSThOREhNaE1HNXhwRncifQ.B1DpnQl4wcyhM9NPOx7NPCTWGoPAXpA9VtoN
c6Mc2SS_oXWuidMGISK1K-ck32-124jIDqVhISUqNLSP2iwUzQ

The following is the payload of a corresponding Key Binding JWT:

{
  "nonce": "1234567890",
  "aud": "https://example.com/verifier",
  "iat": 1751908679,
  "sd_hash": "qomDJzHx8883srAAA3rSUxDSaWzPCI8NDHMhMG5xpFw"
}

After validation, the Verifier will have the following processed SD-JWT payload available for further handling:

```
   {
     "iss": "https://pid-issuer.bund.de.example",
     "iat": 1683000000,
     "exp": 1883000000,
     "vct": "urn:eudi:pid:de:1",
     "cnf": {
       "jwk": {
         "kty": "EC",
         "crv": "P-256",
         "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
         "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
       }
     },
     "age_equal_or_over": {
       "18": true
     },
     "nationalities": [
       "DE"
     ]
   }
```

B.2. Example 2: Type Metadata

```
   {
     "vct": "https://betelgeuse.example.com/education_credential",
     "name": "Betelgeuse Education Credential - Preliminary Version",
     "description": "This is our development version of the education credential. Don’t p
anic.",
     "extends": "https://galaxy.example.com/galactic-education-credential-0.9",
     "extends#integrity": "sha256-ilOUJsTultOwLfz7QUcFALaRa3BP/jelX1ds04kB9yU=",
     "display": [
       {
         "lang": "en-US",
         "name": "Betelgeuse Education Credential",
         "description": "An education credential for all carbon-based life forms on Betel
geusians",
         "rendering": {
           "simple": {
             "logo": {
               "uri": "https://betelgeuse.example.com/public/education-logo.png",
               "uri#integrity": "sha256-LmXfh+9cLlJNXN+TsMk+PmKjZ5t0WRL5ca/xGgX3c1U=",
               "alt_text": "Betelgeuse Ministry of Education logo"
             },
             "background_color": "#12107c",
             "text_color": "#FFFFFF"
           },
           "svg_templates": [
             {
               "uri": "https://betelgeuse.example.com/public/credential-english.svg",
               "uri#integrity": "sha256-I4JcBGO7UfrkOBrsV7ytNJAfGuKLQh+e+Z31mc7iAb4=",
```

```
              "properties": {
                "orientation": "landscape",
                "color_scheme": "light",
                "contrast": "high"
              }
            }
          ]
        }
      },
      {
        "lang": "de-DE",
        "name": "Betelgeuse-Bildungsnachweis",
        "rendering": {
          "simple": {
            "logo": {
              "uri": "https://betelgeuse.example.com/public/education-logo-de.png",
              "uri#integrity": "sha256-LmXfh+9cLlJNXN+TsMk+PmKjZ5t0WRL5ca/xGgX3c1U=",
              "alt_text": "Logo des Betelgeusischen Bildungsministeriums"
            },
            "background_color": "#12107c",
            "text_color": "#FFFFFF"
          },
          "svg_templates": [
            {
              "uri": "https://betelgeuse.example.com/public/credential-german.svg",
              "uri#integrity": "sha256-I4JcBGO7UfrkOBrsV7ytNJAfGuKLQh+e+Z31mc7iAb4=",
              "properties": {
                "orientation": "landscape",
                "color_scheme": "light",
                "contrast": "high"
              }
            }
          ]
        }
      }
    ],
    "claims": [
      {
        "path": ["name"],
        "display": [
          {
            "lang": "de-DE",
            "label": "Vor- und Nachname",
            "description": "Der Name des Studenten"
          },
          {
            "lang": "en-US",
            "label": "Name",
```

```
            "description": "The name of the student"
          }
        ],
        "sd": "allowed"
      },
      {
        "path": ["address"],
        "display": [
          {
            "lang": "de-DE",
            "label": "Adresse",
            "description": "Adresse zum Zeitpunkt des Abschlusses"
          },
          {
            "lang": "en-US",
            "label": "Address",
            "description": "Address at the time of graduation"
          }
        ],
        "sd": "always"
      },
      {
        "path": ["address", "street_address"],
        "display": [
          {
            "lang": "de-DE",
            "label": "Straテ歹"
          },
          {
            "lang": "en-US",
            "label": "Street Address"
          }
        ],
        "sd": "always",
        "svg_id": "address_street_address"
      },
      {
        "path": ["degrees", null],
        "display": [
          {
            "lang": "de-DE",
            "label": "Abschluss",
            "description": "Der Abschluss des Studenten"
          },
          {
            "lang": "en-US",
            "label": "Degree",
            "description": "Degree earned by the student"
```

```
          }
        ],
        "sd": "allowed"
      }
    ],
    "schema_uri": "https://exampleuniversity.com/public/credential-schema-0.9",
    "schema_uri#integrity": "sha256-He4fNeA4xvjLbh/e+rd9Hw3l60OS4tEliHE7NDYXRwA="
  }
```

Appendix C. Acknowledgements

We would like to thank Aaron Parecki, Alen Horvat, Andres Uribe, Christian Bormann, George J Padayatti, Giuseppe De Marco, Lukas J Han, Leif Johansson, Michael B. Jones, Mike Prorock, Orie Steele, Paul Bastian, Pavel Zarecky, Torsten Lodderstedt, Tobias Looker, and Kristina Yasuda for their contributions (some of which substantial) to this draft and to the initial set of implementations.

Appendix D. Document History

-10

* Rename ’Issuer-signed JWT Verification Key Validation’ to ’Issuer Signature Mechanisms’ and rework some text accordingly. Provide a web-based metadata resolution mechanism and an inline x509 mechanism. A DID-based mechanism is not explicitly provided herein but still possible via profile/extension. Be explicit that the employed Issuer Signature Mechanism has to be one that is permitted for the Issuer according to policy. Be more clear that one permitted Issuer Signature Mechanism is sufficient.
* Fix [...]#integrity claim values in examples (Subresource Integrity uses regular base64 encoding and some were wrong length)

-09

* Use SD-JWT KB in place of SD-JWT with Key Binding JWT
* Editorial changes
* Document reasons for not using JSON Pointer or JSON Path (Issue #267)
* Clarify that private claim names MAY be used
* Update PID Example
* Fix section numbering in a few SD-JWT references

-08

* Fix formatting issue introduced by the reintroduction of the DID paragraph in -07

-07

* Revert change from previous release that removed explicit mention of DIDs in the Issuer-signed JWT Verification Key Validation section
* Remove the requirement to insert a .well-known part for vct URLs
* fix section numbering in SD-JWT references to align with the latest -14 version

-06

* Update the anticipated media type registration request from application/vc+sd-jwt to application/dc+sd-jwt
* Tightened the exposition of the Issuer-signed JWT Verification Key Validation section
* Add the 窶彝tatus窶field for the well-known URI registration per IANA early review

-05

* Include display and claim type metadata
* Added example for type metadata
* Clarify, add context, or otherwise improved the examples

-04

* update reference to IETF Status List
* Include Type Metadata
* Include schema Type Metadata
* Editorial changes
* Updated terminology to clarify digital signatures are one way to secure VCs and presentations
* Rework key resolution/validation for x5c

-03

* Include disclosure of age_equal_or_over/18 in the PID example

-02

* Made specific rules for public verification key validation conditional
* Finetuned rules for obtaining public verification key
* Editorial changes
* added Brian Campbell as co-author
* Renamed JWT Issuer Metadata to JWT VC Issuer Metadata
* ’iat’ is now optional and allowed to be selectively disclosable
* Fix inconstancy in the .well-known path construction

* Added registration request to IANA for the well-known URI
* Fix some formatting and text in the media type and JWT claim registration requests
* Clarify the optionality of the cnf claim
* Added relationships to other documents
* Added PID example

-01

* Introduce rules for type identifiers (Collision-Resistant Name)
* Rename type to vct
* Removed duplicated and inconsistent requirements on KB-JWT
* Editorial changes
* Added issuer public verification key discovery section.

-00

* Upload as draft-ietf-oauth-sd-jwt-vc-00
* Aligned terminology and descriptions with latest version of SD-JWT

[[pre Working Group Adoption:]]

-00

* Initial Version
* Removed W3C VCDM transformation algorithm
* Various editorial changes based on feedback
* Adjusted terminology based on feedback
* Added non-selectively disclosable JWT VC
* Added a note that this is not W3C VCDM

Authors' Addresses

Oliver Terbu
MATTR
Email: oliver.terbu@mattr.global

Daniel Fett
Authlete Inc.
Email: mail@danielfett.de

Brian Campbell
Ping Identity
Email: bcampbell@pingidentity.com