Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 10 July 2026

T. Looker
MATTR
P. Bastian
Bundesdruckerei
C. Bormann
SPRIND

6 January 2026

Token Status List (TSL)
draft-ietf-oauth-status-list-15

Abstract

This specification defines a status mechanism called Token Status List (TSL), data structures and processing rules for representing the status of tokens secured by JSON Object Signing and Encryption (JOSE) or CBOR Object Signing and Encryption (COSE), such as JWT, SD-JWT VC, CBOR Web Token, and ISO mdoc. It also defines an extension point and a registry for future status mechanisms.

About This Document

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

The latest revision of this draft can be found at https://oauth-wg.github.io/draft-ietf-oauth-status-list/draft-ietf-oauth-status-list.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/.

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

Source for this draft and an issue tracker can be found at https://github.com/oauth-wg/draft-ietf-oauth-status-list.

Status of This Memo

Copyright Notice

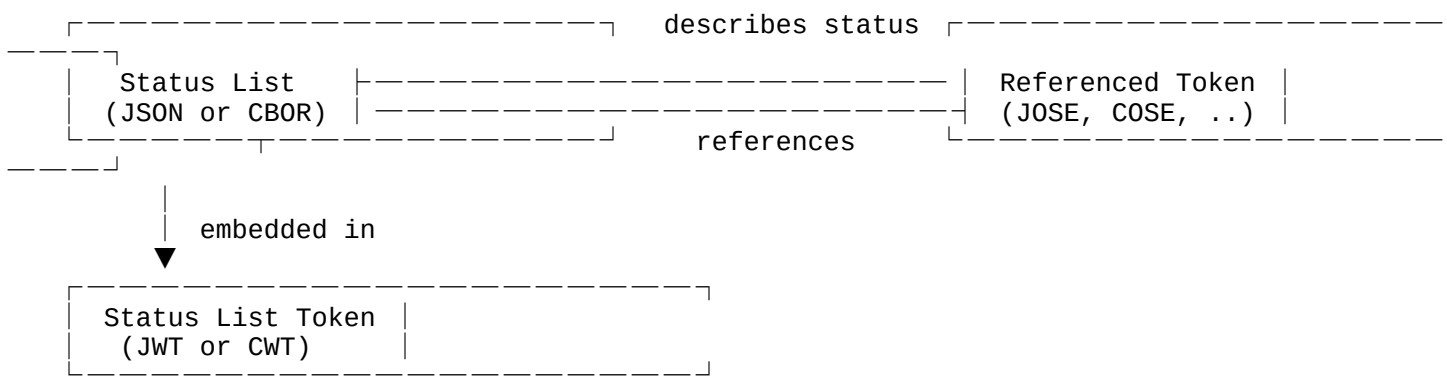
Table of Contents

1. Introduction

Token formats secured by JOSE [RFC7515] or COSE [RFC9052], such as JWTs [RFC7519], SD-JWT VCs [SD-JWT.VC], CWTs [RFC8392] and ISO mdoc [ISO.mdoc], have vast possible applications. Some of these applications can involve issuing a token whereby certain semantics about the token or its validity may change over time. Communicating these changes to relying parties in an interoperable manner, such as whether the token is considered invalidated or suspended by its issuer is important for many of these applications.

This document defines a Status List data structure that describes the individual statuses of multiple Referenced Tokens. A Referenced Token may be of any format, but is most commonly a data structure secured by JOSE or COSE. The Referenced Token is referenced by the Status List, which describes the status of the Referenced Token. The statuses of all Referenced Tokens are conveyed via a bit array in the Status List. Each Referenced Token is allocated an index during issuance that represents its position within this bit array. The value of the bit(s) at this index corresponds to the Referenced Token's status. A Status List is provided within a Status List Token protected by cryptographic signature or MAC and this document defines its representations in JWT and CWT format.

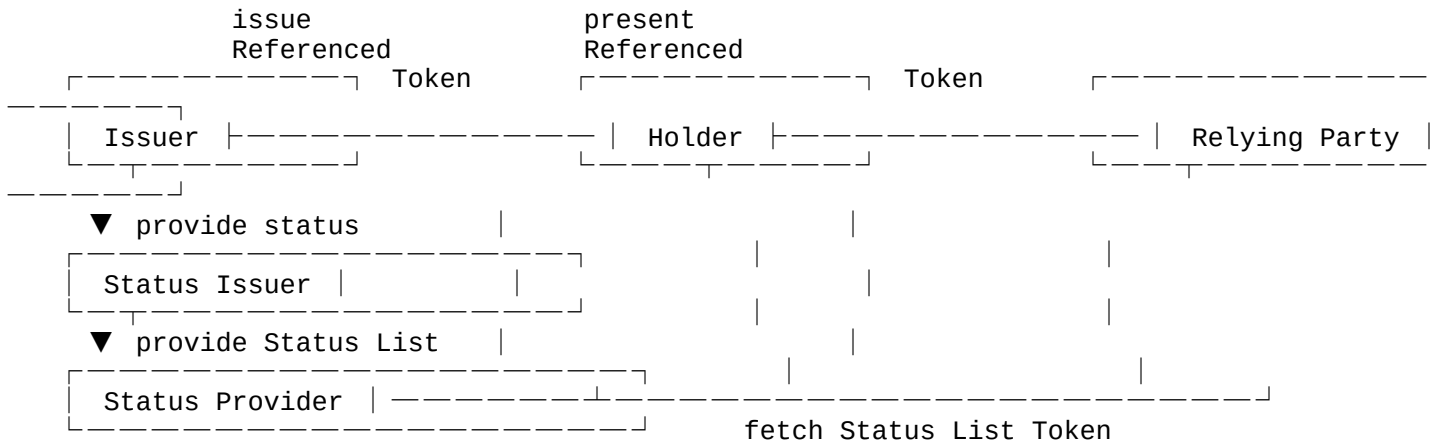
The following diagram depicts the relationship between the artifacts:



An Issuer issues Referenced Tokens to a Holder, the Holder uses and presents those Referenced Tokens to a Relying Party. The Issuer gives updated status information to the Status Issuer, who issues a Status List Token. The Status Issuer can be either the Issuer or an entity that has been authorized by the Issuer to issue Status List Tokens. The Status Issuer provides the Status List Token to the Status Provider, who serves the Status List Token on a public, resolvable endpoint. The Relying Party or the Holder may fetch the Status List Token to retrieve the status of the Referenced Token.

The roles of the Issuer (of the Referenced Token), the Status Issuer and the Status Provider may be fulfilled by the same entity. If not further specified, the term Issuer may refer to an entity acting for all three roles. This document describes how an Issuer references a Status List Token and how a Relying Party fetches and validates Status Lists.

The following diagram depicts the relationship between the involved roles (Relying Party is equivalent to Verifier of [SD-JWT.VC]):



Status Lists can be used to express a variety of Status Types. This document defines basic Status Types for the most common use cases as well as an extensibility mechanism for custom Status Types.

Furthermore, the document creates an extension point and an IANA registry that enables other specifications to describe additional status mechanisms.

1.1. Example Use Cases

An example of the usage of a Status List is to manage the statuses of issued access tokens as defined in Section 1.4 of [RFC6749]. Token Introspection [RFC7662] provides a method to determine the status of an issued access token, but it necessitates the party attempting to validate the state of access tokens to directly contact the Issuer of each token for validation. In contrast, the mechanism defined in this specification allows a party to retrieve the statuses for many tokens, reducing interactions with the Issuer substantially. This not only improves scalability but also enhances privacy by preventing the Issuer from gaining knowledge of access tokens being verified (herd anonymity).

Another possible use case for the Status List is to express the status of verifiable credentials (Referenced Tokens) issued by an Issuer in the Issuer-Holder-Verifier model [SD-JWT.VC].

1.2. Rationale

Revocation mechanisms are an essential part of most identity ecosystems. In the past, revocation of X.509 TLS certificates has been proven difficult. Traditional certificate revocation lists (CRLs) have limited scalability; Online Certificate Status Protocol (OCSP) has additional privacy risks, since the client is leaking the requested website to a third party. OCSP stapling is addressing some

of these problems at the cost of less up-to-date data. Modern approaches use accumulator-based revocation registries and Zero-Knowledge-Proofs to accommodate for this privacy gap, but face scalability issues again. Another alternative is short-lived Referenced Tokens with regular re-issuance, but this puts additional burden on the Issuer's infrastructure.

This specification seeks to find a balance between scalability, security and privacy by representing statuses as individual bits, packing them into an array, and compressing the resulting binary data. Thereby, a Status List may contain statuses of many thousands or millions Referenced Tokens while remaining as small as possible. Placing a large number of Referenced Tokens into the same list also offers Holders and Relying Parties herd privacy from the Status Provider.

1.3. Design Considerations

The decisions taken in this specification aim to achieve the following design goals:

* the specification shall favor a simple and easy-to-understand concept
* the specification shall be easy, fast and secure to implement in all major programming languages
* the specification shall be optimized to support the most common use cases and avoid unnecessary complexity of corner cases
* the Status List shall scale up to millions of tokens to support large-scale government or enterprise use cases
* the Status List shall enable caching policies and offline support
* the specification shall support JSON and CBOR based tokens
* the specification shall not specify key resolution or trust frameworks
* the specification shall define an extension point that enables other mechanisms to convey information about the status of a Referenced Token

1.4. Prior Work

Representing statuses with bits in an array is a rather old and well-known concept in computer science and there has been prior work to use this for revocation and status management such as a paper by Smith et al. [smith2020let] that proposed a mechanism called Certificate Revocation Vectors based on xz compressed bit vectors for each expiration day and the W3C bit Status List [W3C.SL] that similarly uses a compressed bit representation.

1.5. Status Mechanisms Registry

This specification establishes IANA "Status Mechanisms" registries for status mechanisms for JOSE-based tokens and for status mechanisms for COSE-based tokens and registers the members defined by this specification. Other specifications can register other members used for status retrieval.

Other status mechanisms may have different tradeoffs regarding security, privacy, scalability and complexity. The privacy and security considerations in this document only represent the properties of the Status List mechanism.

2. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

Issuer: An entity that issues the Referenced Token. Also known as a Provider.

Status Issuer: An entity that issues the Status List Token about the status information of the Referenced Token. This role may be fulfilled by the Issuer.

Status Provider: An entity that provides the Status List Token on a public endpoint. This role may be fulfilled by the Status Issuer.

Holder: An entity that receives Referenced Tokens from the Issuer and presents them to Relying Parties.

Relying Party: An entity that relies on the Referenced Token and

fetches the corresponding Status List Token to validate the status of that Referenced Token. Also known as a Verifier.

Status List: An object in JSON or CBOR representation containing a compressed byte array that represents the statuses of many Referenced Tokens.

Status List Token: A token in JWT (as defined in [RFC7519]) or CWT (as defined in [RFC8392]) representation that contains a cryptographically secured Status List.

Referenced Token: A cryptographically secured data structure that contains a "status" claim that references a mechanism to retrieve status information about this Referenced Token. This document defines the Status List mechanism in which case the Referenced Token contains a reference to an entry in a Status List Token. It is RECOMMENDED to use JSON [RFC8259] with JOSE as defined in [RFC7515] or CBOR [RFC8949] with COSE as defined in [RFC9052]. Examples for Referenced Tokens are SD-JWT VC and ISO mdoc.

Client: An application that fetches information, such as a Status List Token, from the Status List Provider on behalf of the Holder or Relying Party.

base64url: Denotes the URL-safe base64 encoding with all trailing '=' characters omitted as defined in Section 2 of [RFC7515] as "Base64url Encoding".

4. Status List

A Status List is a data structure that contains the statuses of many Referenced Tokens represented by one or multiple bits. Section 4.1 describes how to construct a compressed byte array that is the base component for the Status List data structure. The second and third sections describe how to encode such a Status List in JSON and CBOR representations.

4.1. Compressed Byte Array

A compressed byte array containing the status information of the Referenced Token is composed by the following algorithm:

1. The Status Issuer MUST define a number of bits (bits) of either 1,2,4 or 8, that represents the amount of bits used to describe the status of each Referenced Token within this Status List. Therefore, up to 2,4,16 or 256 statuses for a Referenced Token are possible, depending on the bit size. This limitation is intended to limit bit manipulation necessary to a single byte for every operation and thus keeping implementations simpler and less error-prone.

2. The Status Issuer creates a byte array of size = amount of Referenced Tokens * bits / 8 or greater. Depending on the bits, each byte in the array corresponds to 8/(bits) statuses (8,4,2 or 1).

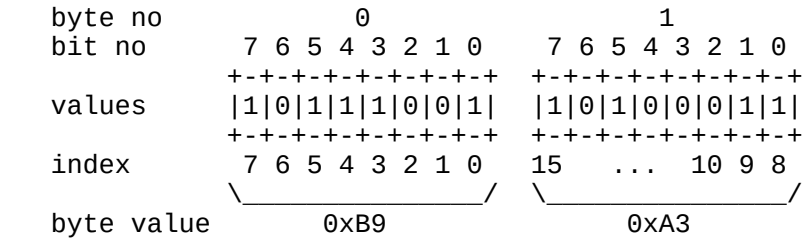
3. The Status Issuer sets the status values for all Referenced Tokens within the byte array. Each Referenced Token is assigned a distinct index from 0 to one less than the number of Referenced Tokens assigned to the Status List. Each index identifies a contiguous block of bits in the byte array, with the blocks being packed into bytes from the least significant bit ("0") to the most significant bit ("7"). These bits contain the encoded status value of the Referenced Token (see Section 7 for more details on the values).

4. The Status Issuer compresses the byte array using DEFLATE [RFC1951] with the ZLIB [RFC1950] data format. Implementations are RECOMMENDED to use the highest compression level available.

The following example illustrates the byte array of a Status List that represents the statuses of 16 Referenced Tokens with a bits of 1, requiring 2 bytes (16 bits) for the uncompressed byte array:

```
status[0] = 0b1
status[1] = 0b0
status[2] = 0b0
status[3] = 0b1
status[4] = 0b1
status[5] = 0b1
status[6] = 0b0
status[7] = 0b1
status[8] = 0b1
status[9] = 0b1
status[10] = 0b0
status[11] = 0b0
status[12] = 0b0
status[13] = 0b1
status[14] = 0b0
status[15] = 0b1
```

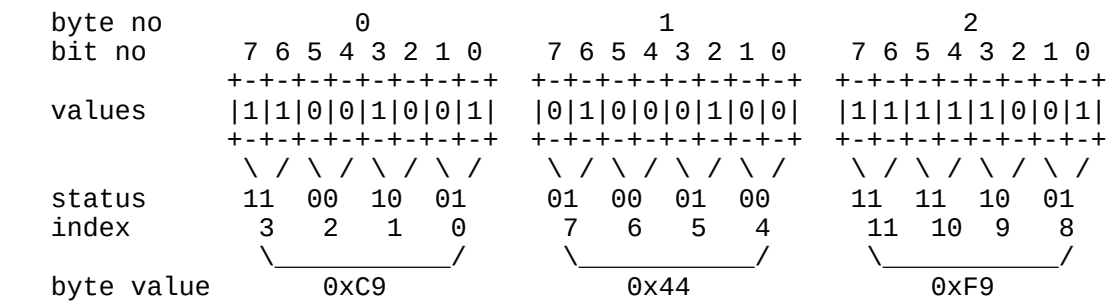
These bits are concatenated:



In the following example, the Status List additionally includes the Status Type "SUSPENDED". As the Status Type value for "SUSPENDED" is 0x02 and does not fit into 1 bit, the bits is required to be 2. This example illustrates the byte array of a Status List that represents the statuses of 12 Referenced Tokens with a bits of 2, requiring 3 bytes (24 bits) for the uncompressed byte array:

```
status[0] = 0b01
status[1] = 0b10
status[2] = 0b00
status[3] = 0b11
status[4] = 0b0
status[5] = 0b01
status[6] = 0b00
status[7] = 0b01
status[8] = 0b01
status[9] = 0b10
status[10] = 0b11
status[11] = 0b11
```

These bits are concatenated:



4.2. Status List in JSON Format

This section defines the data structure for a JSON-encoded Status List:

* The StatusList structure is a JSON Object that contains the following members:

  - bits: REQUIRED. JSON Integer specifying the number of bits per Referenced Token in the compressed byte array (lst). The allowed values for bits are 1, 2, 4, and 8.

  - lst: REQUIRED. JSON String that contains the status values for all the Referenced Tokens it conveys statuses for. The value MUST be the base64url-encoded compressed byte array as specified in Section 4.1.

  - aggregation_uri: OPTIONAL. JSON String that contains a URI to retrieve the Status List Aggregation for this type of Referenced Token or Issuer. See Section 9 for further details.

The following example illustrates the JSON representation of the Status List with bits=1 from the examples above:

```
byte_array = [0xb9, 0xa3]
encoded:
{
  "bits": 1,
  "lst": "eNrbuRgAAhcBXQ"
}
```

The following example illustrates the JSON representation of the Status List with bits=2 from the examples above:

```
byte_array = [0xc9, 0x44, 0xf9]
encoded:
{
  "bits": 2,
  "lst": "eNo76fITAAPfAgc"
}
```

See Appendix C for more test vectors.

4.3. Status List in CBOR Format

This section defines the data structure for a CBOR-encoded Status List:

* The StatusList structure is a CBOR map (major type 5) and defines the following entries:

- bits: REQUIRED. CBOR Unsigned integer (major type 0) that contains the number of bits per Referenced Token in the compressed byte array (lst). The allowed values for bits are 1, 2, 4, and 8.

- lst: REQUIRED. CBOR Byte string (major type 2) that contains the status values for all the Referenced Tokens it conveys statuses for. The value MUST be the compressed byte array as specified in Section 4.1.

- aggregation_uri: OPTIONAL. CBOR Text string (major type 3) that contains a URI to retrieve the Status List Aggregation for this type of Referenced Token. See Section 9 for further detail.

The following is the CDDL [RFC8610] definition of the StatusList structure:

```
StatusList = {
    bits: 1 / 2 / 4 / 8, ; The number of bits used per Referenced Token
    lst: bstr, ; Byte string that contains the Status List
    ? aggregation_uri: tstr ; link to the Status List Aggregation
}
```

The following example illustrates the CBOR representation of the Status List in Hex:

```
byte_array = [0xb9, 0xa3]
encoded:
a2646269747301636c73744a78dadbb918000217015d
```

The following is the CBOR Annotated Hex output of the example above:

```
a2                              # map(2)
  64                            #   string(4)
    62697473                    #     "bits"
  01                            #   uint(1)
  63                            #   string(3)
    6c7374                      #     "lst"
  4a                            #   bytes(10)
    78dadbb918000217015d        #     "x\x18\x00\x02\x17\x01]"
```

See Appendix C for more test vectors.

5. Status List Token

A Status List Token embeds a Status List into a token that is cryptographically signed and protects the integrity of the Status List. This allows for the Status List Token to be hosted by third parties or be transferred for offline use cases.

This section specifies Status List Tokens in JSON Web Token (JWT) and CBOR Web Token (CWT) format.

5.1. Status List Token in JWT Format

The Status List Token MUST be encoded as a "JSON Web Token (JWT)" according to [RFC7519].

The following content applies to the JWT Header:

* typ: REQUIRED. The JWT type MUST be statuslist+jwt.

The following content applies to the JWT Claims Set:

* sub: REQUIRED. As generally defined in [RFC7519]. The sub (subject) claim MUST specify the URI of the Status List Token. The value MUST be equal to that of the uri claim contained in the status_list claim of the Referenced Token.

* iat: REQUIRED. As generally defined in [RFC7519]. The iat (issued at) claim MUST specify the time at which the Status List Token was issued.

* exp: RECOMMENDED. As generally defined in [RFC7519]. The exp (expiration time) claim, if present, MUST specify the time at which the Status List Token is considered expired by the Status Issuer. Consider the guidance provided in Section 13.7.

* ttl: RECOMMENDED. The ttl (time to live) claim, if present, MUST specify the maximum amount of time, in seconds, that the Status List Token can be cached by a consumer before a fresh copy SHOULD be retrieved. The value of the claim MUST be a positive number encoded in JSON as a number. Consider the guidance provided in Section 13.7.

* status_list: REQUIRED. The status_list (status list) claim MUST specify the Status List conforming to the structure defined in Section 4.2.

The following additional rules apply:

1. The JWT MAY contain other claims.

2. The JWT MUST be secured using a cryptographic signature or MAC algorithm. Relying Parties MUST reject JWTs with an invalid signature.

3. Relying Parties MUST reject JWTs that are not valid in all other respects per "JSON Web Token (JWT)" [RFC7519].

4. Application of additional restrictions and policies are at the discretion of the Relying Party.

The following is a non-normative example of a Status List Token in JWT format:

```
{
  "alg": "ES256",
  "kid": "12",
  "typ": "statuslist+jwt"
}
.
{
  "exp": 2291720170,
  "iat": 1686920170,
  "status_list": {
    "bits": 1,
    "lst": "eNrbuRgAAhcBXQ"
  },
  "sub": "https://example.com/statuslists/1",
  "ttl": 43200
}
```

5.2. Status List Token in CWT Format

The Status List Token MUST be encoded as a "CBOR Web Token (CWT)" according to [RFC8392]. The Status List Token MUST NOT be tagged with the CWT tag defined in Section 6 of [RFC8392]. The COSE message MUST either be the tagged COSE_Sign1_Tagged (18) or COSE_Mac0_Tagged (17) as defined in Section 2 of [RFC9052].

The following content applies to the protected header of the CWT:

* 16 (type): REQUIRED. The type of the CWT MUST be application/statuslist+cwt or the registered CoAP Content-Format ID (see Section 14.8) as defined in [RFC9596].

The following content applies to the CWT Claims Set:

* 2 (subject): REQUIRED. As generally defined in [RFC8392]. The subject claim MUST specify the URI of the Status List Token. The value MUST be equal to that of the uri claim contained in the status_list claim of the Referenced Token.

* 6 (issued at): REQUIRED. As generally defined in [RFC8392]. The issued at claim MUST specify the time at which the Status List Token was issued.

* 4 (expiration time): RECOMMENDED. As generally defined in [RFC8392]. The expiration time claim, if present, MUST specify the time at which the Status List Token is considered expired by its issuer. Consider the guidance provided in Section 13.7.

* 65534 (time to live): RECOMMENDED. Unsigned integer (major type 0). The time to live claim, if present, MUST specify the maximum amount of time, in seconds, that the Status List Token can be cached by a consumer before a fresh copy SHOULD be retrieved. The value of the claim MUST be a positive number. Consider the guidance provided in Section 13.7.

* 65533 (status list): REQUIRED. The status list claim MUST specify the Status List conforming to the structure defined in Section 4.3.

The following additional rules apply:

1. The CWT MAY contain other claims.

2. The CWT MUST be secured using a cryptographic signature or MAC algorithm. Relying Parties MUST reject CWTs with an invalid signature.

3. Relying Parties MUST reject CWTs that are not valid in all other respects per "CBOR Web Token (CWT)" [RFC8392].

4. Application of additional restrictions and policies are at the discretion of the Relying Party.

The following is a non-normative example of a Status List Token in CWT format in Hex:

```
d2845820a2012610781a6170706c69636174696f6e2f7374617475736c6973742b63
7774a1044231325850a502782168747470733a2f2f6578616d706c652e636f6d2f73
74617475736c697374732f31061a648c5bea041a8898dfea19fffe19a8c019fffda2
646269747301636c73744a78dadbb918000217015d584093fa4d01032b18c35e2fe1
101b77fd6cc9440022caa4694450c4e4e9feab4e99d1fa6d9772ce2bf3a12e0323de
d7c982c5e101a5e67f0cbc1e2b6f57ce99c279
```

The following is the CBOR Annotated Hex output of the example above:

```
d2                                # tag(18)
  84                              #   array(4)
    58 20                         #     bytes(32)
      a2012610781a6170706c6963    #       "¢ \x01&\x10x\x1aapplic"
      6174696f6e2f737461747573    #       "ation/status"
      6c6973742b637774            #       "list+cwt"
    a1                            #     map(1)
      04                          #       uint(4)
      42                          #       bytes(2)
        3132                      #         "12"
    58 50                         #     bytes(80)
      a502782168747470733a2f2f    #       "¥ \x02x!https://"
      6578616d706c652e636f6d2f    #       "example.com/"
      7374617475736c697374732f    #       "statuslists/"
      31061a648c5bea041a8898df    #       "1\x06\x1ad\x8c[\x04\x1a\x88\x98"
      ea19fffe19a8c019fffda264    #       "\x19\x19¨\x19¢d"
      6269747301636c73744a78da    #       "bits\x01clstJx"
      dbb918000217015d            #       "\x18\x00\x02\x17\x01]"
    58 40                         #     bytes(64)
      93fa4d01032b18c35e2fe110    #       "\x93M\x01\x03+\x18^/\x10"
      1b77fd6cc9440022caa46944    #       "\x1bwlD\x00"iD"
      50c4e4e9feab4e99d1fa6d97    #       "PN\x99m\x97"
      72ce2bf3a12e0323ded7c982    #       "r+.\x03#× \x82"
      c5e101a5e67f0cbc1e2b6f57    #       "\x01¥\x7f\x0c\x1e+oW"
      ce99c279                    #       "\x99y"
```

6. Referenced Token

6.1. Status Claim

By including a "status" claim in a Referenced Token, the Issuer is referencing a mechanism to retrieve status information about this Referenced Token. This specification defines one possible member of the "status" object, called "status_list". Other members of the "status" object may be defined by other specifications. This is analogous to "cnf" claim in Section 3.1 of [RFC7800] in which different authenticity confirmation methods can be included.

6.2. Referenced Token in JOSE

The Referenced Token MAY be encoded as a "JSON Web Token (JWT)" according to [RFC7519] or other formats based on JOSE.

The following content applies to the JWT Claims Set:

* status: REQUIRED. The status (status) claim MUST specify a JSON Object that contains at least one reference to a status mechanism.

  - status_list: REQUIRED when the status mechanism defined in this specification is used. It MUST specify a JSON Object that contains a reference to a Status List Token. It MUST at least contain the following claims:

    o idx: REQUIRED. The idx (index) claim MUST specify a non-negative Integer that represents the index to check for status information in the Status List for the current Referenced Token.

    o uri: REQUIRED. The uri (URI) claim MUST specify a String value that identifies the Status List Token containing the status information for the Referenced Token. The value of uri MUST be a URI conforming to [RFC3986].

Application of additional restrictions and policies are at the discretion of the Relying Party.

The following is a non-normative example of a decoded header and payload of a Referenced Token:

```
{
  "alg": "ES256",
  "kid": "11"
}
.
{
  "status": {
    "status_list": {
      "idx": 0,
      "uri": "https://example.com/statuslists/1"
    }
  }
}
```

SD-JWT-based Verifiable Credentials (Section 3.2.2.2 of [SD-JWT.VC]) introduces the usage of the status mechanism defined in this section. Therefore, an SD-JWT VC can be considered a Referenced Token. The following is a non-normative example of a Referenced Token in SD-JWT VC serialized form as received from an Issuer:

```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImV4YW1wbGUrc2Qtand0In0.eyJfc2QiOiBb
Ikh2cktYNmZQVjB2OUtfeUNWRkJpTEZIc01heGNEXzExNEVtNlZUOHgxbGciXSwgImlz
cyI6ICJodHRwczovL2V4YW1wbGUuY29tL2lzc3VlciIsICJpYXQiOiAxNjgzMDAwMDAw
LCAiZXhwIjogMTg4MzAwMDAwMCwgInN1YiI6ICI2YzVjMGE0OS1iNTg5LTQzMWQtYmFl
Ny0yMTkxMjJhOWVjMmMiLCAic3RhdHVzIjogeyJzdGF0dXNfbGlzdCI6IHsiaWR4Ijog
MCwgInVyaSI6ICJodHRwczovL2V4YW1wbGUuY29tL3N0YXR1c2xpc3RzLzEifX0sICJf
c2RfYWxnIjogInNoYS0yNTYifQ.-kgS-R-Z4DEDlqb8kb6381_gHHNatsoF1fcVKZk3M
06CrnV8F8k9d2w2V_YAOvgcb0f11FqDFezXBXH30d4vcw~WyIyR0xDNDJzS1F2ZUNmR2
ZyeU5STjl3IiwgInN0cmVldF9hZGRyZXNzIiwgIlNjaHVsc3RyLiAxMiJd~WyJlbHVWN
U9nM2dTTklJOEVZbnN4QV9BIiwgImxvY2FsaXR5IiwgIlNjaHVscGZvcnRhIl0~WyI2S
Wo3dE0tYTVpVlBHYm9TNXRtdlZBIiwgInJlZ2lvbiIsICJTYWNoc2VuLUFuaGFsdCJd~
WyJlSThaV205UW5LUHBOUGVOZW5IZGhRIiwgImNvdW50cnkiLCAiREUiXQ~WyJRZ19PN
jR6cUF4ZTQxMmExMDhpcm9BIiwgImFkZHJlc3MiLCB7Il9zZCI6IFsiNnZoOWJxLXpTN
EdLTV83R3BnZ1ZiWXp6dTZvT0dYcm1OVkdQSFA3NVVkMCIsICI5Z2pWdVh0ZEZST0NnU
nJ0TmNHVVhtRjY1cmRlemlfNkVyX2o3NmttWXlNIiwgIktVUkRQaDRaQzE5LTN0aXotR
GYzOVY4ZWlkeTFvVjNhM0gxRGEyTjBnODgiLCAiV045cjlkQ0JKOEhUQ3NTMmpLQVN4V
GpFeVc1bTV4NjVfWl8ycm8yamZYTSJdfV0~
```

The resulting payload of the example above:

```
{
  "_sd": [
    "HvrKX6fPV0v9K_yCVFBiLFHsMaxcD_114Em6VT8x1lg"
  ],
  "iss": "https://example.com/issuer",
  "iat": 1683000000,
  "exp": 1883000000,
  "sub": "6c5c0a49-b589-431d-bae7-219122a9ec2c",
  "status": {
    "status_list": {
      "idx": 0,
      "uri": "https://example.com/statuslists/1"
    }
  },
  "_sd_alg": "sha-256"
}
```

6.3. Referenced Token in COSE

The Referenced Token MAY be encoded as a "CBOR Web Token (CWT)" object according to [RFC8392] or other formats based on COSE. Referenced Tokens in CBOR should share the same core data structure for a status list reference:

* The Status CBOR structure is a Map that MUST include at least one data item that refers to a status mechanism. Each data item in the Status CBOR structure comprises a key-value pair, where the key must be a CBOR text string (major type 3) specifying the identifier of the status mechanism and the corresponding value defines its contents.

  - status_list (status list): REQUIRED when the status mechanism defined in this specification is used. It has the same definition as the status_list claim in Section 6.2 but MUST be encoded as a StatusListInfo CBOR structure with the following fields:

    o idx: REQUIRED. Unsigned integer (major type 0) The idx (index) claim MUST specify a non-negative Integer that represents the index to check for status information in the Status List for the current Referenced Token.

    o uri: REQUIRED. Text string (major type 3). The uri (URI) claim MUST specify a String value that identifies the Status List Token containing the status information for the Referenced Token. The value of uri MUST be a URI conforming to [RFC3986].

6.3.1. CBOR Web Token (CWT)

The following content applies to the CWT Claims Set:

* 65535 (status): REQUIRED. The status claim contains the Status CBOR structure as described in Section 6.3.

Application of additional restrictions and policies are at the discretion of the Relying Party.

The following is a non-normative example of a Referenced Token in CWT format in Hex:

```
d28443a10126a1044231325866a5026531323334350173687474707 33a2f2f657861
6d706c652e636f6d061a648c5bea041a8898dfea19ffffa16b7374617475735f6c69
7374a2636964780063757269782168747470733a2f2f6578616d706c652e636f6d2f
7374617475736c697374732f315840340f7efea10f1a36dc4797636a17b4dd4848b6
8997d1d10e8cceb3a38ff33b3dda72964a83989f6cf98560c2fc97a08bc8977cc6b0
f84cfedab93d3e4481e938
```

The following is the CBOR Annotated Hex output of the example above:

```
d2                                # tag(18)
  84                              #   array(4)
    43                            #     bytes(3)
      a10126                      #       "\x01&"
    a1                            #     map(1)
      04                          #       uint(4)
      42                          #       bytes(2)
        3132                      #         "12"
    58 66                         #     bytes(102)
      a5026531323334350173687 4   #       "¥\x02e12345\x01sht"
      7470733a2f2f6578616d706c    #       "tps://exampl"
      652e636f6d061a648c5bea04    #       "e.com\x06\x1ad\x8c[\x04"
      1a8898dfea19ffffa16b7374    #       "\x1a\x88\x98\x19kst"
      617475735f6c697374a26369    #       "atus_list¢ci"
      6478006375726978216874 74   #       "dx\x00curix!htt"
      70733a2f2f6578616d706c65    #       "ps://example"
      2e636f6d2f7374617475736c    #       ".com/statusl"
      697374732f31                #       "ists/1"
    58 40                         #     bytes(64)
      340f7efea10f1a36dc479763    #       "4\x0f~\x0f\x1a6G\x97c"
      6a17b4dd4848b68997d1d10e    #       "j\x17´HH¶\x89\x97\x0e"
      8cceb3a38ff33b3dda72964a    #       "\x8c£\x8f;=r\x96J"
      83989f6cf98560c2fc97a08b    #       "\x83\x98\x9fl\x85‘\x97\xa0\x8b"
      c8977cc6b0f84cfedab93d3e    #       "\x97|°L=>"
      4481e938                    #       "D\x818"
```

6.3.2. ISO mdoc

ISO mdoc [ISO.mdoc] may utilize the Status List mechanism by introducing the status parameter in the Mobile Security Object (MSO) as specified in Section 9.1.2 of [ISO.mdoc]. The status parameter contains the Status CBOR structure as described in Section 6.3.

It is RECOMMENDED to use status for the label of the field that contains the Status CBOR structure.

Application of additional restrictions and policies are at the discretion of the Relying Party.

The following is a non-normative example of an IssuerAuth as specified in ISO mDL (also referred to as signed MSO) in Hex:

```
8443a10126a118215901f3308201ef30820195a00302010202140bfec7da97e048e
15ac3dacb9eafe82e64fd07f5300a06082a8648ce3d0403023023311430120603550
04030c0b75746f7069612069616361310b3009060355040613025553301e170d323
431303031303030303030305a170d3235313030313030303030305a30213112301006
035504030c0975746f706961206473310b3009060355040613025553305930130600
72a8648ce3d020106082a8648ce3d03010703420004ace7ab7340e5d9648c5a72a9
a6f56745c7aad436a03a43efea77b5fa7b88f0197d57d8983e1b37d3a539f4d5883
65e38cbbf5b94d68c547b5bc8731dcd2f146ba381a83081a5301c0603551d1f0415
30133011a00fa00d820b6578616d706c652e636f6d301e0603551d1204173015811
36578616d706c65406578616d706c652e636f6d301d0603551d0e0416041414e290
17a6c35621ffc7a686b7b72db06cd12351301f0603551d2304183016801454fa238
3a04c28e0d930792261c80c4881d2c00b300e0603551d0f0101ff04040302078030
150603551d250101ff040b3009060728818c5d050102300a06082a8648ce3d04030
203480030450221 00b7103fd4b90529f50bd6f70c5ae5ce7f4f3d4d15a4e082812f
9fa1f5c2e5aa0a0220070b2822ec7ce6c56804923a85b2cfbffd054cf9a915f070c
fef7179a4bc6569590320d81859031ba766737461747573a16b7374617475735f6c
697374a26369647819019c6375726978216874747073 3a2f2f6578616d706c652e6
36f6d2f7374617475736c697374732f3167646f6354797065756f72672e69736f2e
31383031332e352e312e6d444c6776657273696f6e63312e306c76616c696469747
9496e666fa3667369676e6564c074323032342d31302d30315431333a33303a3032
5a6976616c696446726f6dc074323032342d31302d30315431333a33303a30325a6
a76616c6964556e74696cc074323032352d31302d30315431333a33303a30325a6c
76616c7565446967657373473a1716f72672e69736f2e31383031332e352e31ac005
820a81d65ed5075fbd7ee19fa66e2bb3047ed826e2769873e7ef07c923da7a6f243
01582048701a9546492284d266ed81d439230a582d0e1f17a08ab1859a3efe98069
0a4025820d11fe48c8835b30bfb3895c3905436ddfb63f59ab9eee181b110985329
2a8f62035820a741bf05e20a8bc359e32426106ed0899b2c60262cc3acc637ddc99
41095fb7a045820ab67cb9a8f20a8572f77f02727367d08dc8e57fb89deb46b9c62
6e94457b7d8b055820bacddb4142b3842bd555206eb5acb27ded063294995c7e7fe
fbf93ece522604d065820bfd02b3aebdc05b53b5539226c38088d6d784b0ea0fab6
9eb9311650a48d325307582027dab70fe71da63e5e5d199e8ae5b79cbe8904bc30c
5b7544fb809e02ccb3e6a0858200dbd7ccc9c7727d3d17295f1b6f1914071670ee2
3d4d33530c31f1f406b8e3b7095820a5beb5efadf37f21637209abc519830681cc5
1f334818a823fec13b29552f5ba0a5820d8047c95f9272d7d07b2c13a9f5ac2ee02
380ab272a165e569391d89a2152c3c0b582004939930ffb4911ef03487a153605a3
0368b69f2437d6d21b4c90f92bc144c3e6d6465766963654b6579496e666fa16964
65766963654b6579a401022001215820963 13d6c63e24e3372742bfdb1a33ba2c89
7dcd68ab8c753e4fbd48dca6b7f9a2258201fb3269edd418857de1b39a4e4a44b92
fa484caa722c228288f01d0c03a2c3d66f646967657374416c676f726974686d675
348412d3235365840b7c2d4abe85aa5ba814ef95de0385c71c802be8ac33a4a971a
85ed800ba7acb59cb21035f4a68fc0caa450cbefd3b255aec72f83595f0ae7b7d50
fe8a1c4cafe
```

The following is the CBOR Diagnostic Notation of the example above:

```
   [
     << {
       1: -7
     } >>,
     {
       33: h'308201ef30820195a00302010202140bfec7da97e048e15ac3dacb9ea
       fe82e64fd07f5300a06082a8648ce3d040302302331143012060355040030c0b
       75746f706961206961636131 0b3009060355040613025553301e170d3234313
       0303130303030305a170d323531303031303030303030305a30213112301006
       035504030c0975746f706961206473310b3009060355040613025553059301
       306072a8648ce3d020106082a8648ce3d03010703420004ace7ab7340e5d964
       8c5a72a9a6f56745c7aad436a03a43efea77b5fa7b88f0197d57d8983e1b37d
       3a539f4d588365e38cbbf5b94d68c547b5bc8731dcd2f146ba381a83081a530
       1c0603551d1f041530133011a00fa00d820b6578616d706c652e636f6d301e0
       603551d120417301581136578616d706c65406578616d706c652e636f6d301d
       0603551d0e0416041414e29017a6c35621ffc7a686b7b72db06cd12351301f0
       603551d2304183016801454fa2383a04c28e0d930792261c80c4881d2c00b30
       0e0603551d0f0101ff040403020780301506035 51d250101ff040b300906072
       8818c5d050102300a06082a8648ce3d040302034800304502 2100b7103fd4b9
       0529f50bd6f70c5ae5ce7f4f3d4d15a4e082812f9fa1f5c2e5aa0a0220070b2
       822ec7ce6c56804923a85b2cfbffd054cf9a915f070cfef7179a4bc6569'
     },
     << 24( << {
       "status": {
         "status_list": {
           "idx": 412,
           "uri": "https://example.com/statuslists/1"
         }
       },
       "docType": "org.iso.18013.5.1.mDL",
       "version": "1.0",
       "validityInfo": {
         "signed": 2024-10-01 13:30:02+00:00,
         "validFrom": 2024-10-01 13:30:02+00:00,
         "validUntil": 2025-10-01 13:30:02+00:00
       },
       "valueDigests": {
         "org.iso.18013.5.1": {
           0: h'a81d65ed5075fbd7ee19fa66e2bb3047ed826e2769873e7ef07c92
           3da7a6f243',
           1: h'48701a9546492284d266ed81d439230a582d0e1f17a08ab1859a3e
           fe980690a4',
           2: h'd11fe48c8835b30bfb3895c3905436ddfb63f59ab9eee181b11098
           53292a8f62',
           3: h'a741bf05e20a8bc359e32426106ed0899b2c60262cc3acc637ddc9
           941095fb7a',
           4: h'ab67cb9a8f20a8572f77f02727367d08dc8e57fb89deb46b9c626e
           94457b7d8b',
```

```
          5: h'bacddb4142b3842bd555206eb5acb27ded063294995c7e7fefbf93
          ece522604d',
          6: h'bfd02b3aebdc05b53b5539226c38088d6d784b0ea0fab69eb93116
          50a48d3253',
          7: h'27dab70fe71da63e5e5d199e8ae5b79cbe8904bc30c5b7544fb809
          e02ccb3e6a',
          8: h'0dbd7ccc9c7727d3d17295f1b6f1914071670ee23d4d33530c31f1
          f406b8e3b7',
          9: h'a5beb5efadf37f21637209abc519830681cc51f334818a823fec13
          b29552f5ba',
          10: h'd8047c95f9272d7d07b2c13a9f5ac2ee02380ab272a165e569391
          d89a2152c3c',
          11: h'04939930ffb4911ef03487a153605a30368b69f2437d6d21b4c90
          f92bc144c3e'
        }
      },
      "deviceKeyInfo": {
        "deviceKey": {
          1: 2,
          -1: 1,
          -2: h'96313d6c63e24e3372742bfdb1a33ba2c897dcd68ab8c753e4fbd
          48dca6b7f9a',
          -3: h'1fb3269edd418857de1b39a4e4a44b92fa484caa722c228288f01
          d0c03a2c3d6'
        }
      },
      "digestAlgorithm": "SHA-256"
    } >> ) >>,
    h'b7c2d4abe85aa5ba814ef95de0385c71c802be8ac33a4a971a85ed800ba7acb
    59cb21035f4a68fc0caa450cbefd3b255aec72f83595f0ae7b7d50fe8a1c4cafe'
  ]
```

7. Status Types

This document defines the statuses of Referenced Tokens as Status Type values. A status describes the state, mode, condition or stage of an entity that is represented by the Referenced Token.

A Status List represents exactly one status per Referenced Token. If the Status List contains more than one bit per token (as defined by bits in the Status List), then the whole value of bits MUST describe one value. Status Types MUST have a numeric value between 0 and 255 for their representation in the Status List. The issuer of the Status List MUST choose an adequate bits value (bit size) to be able to describe the required Status Types for its application.

7.1. Status Types Values

The processing rules for Referenced Tokens (such as JWT or CWT) supersede the Referenced Token's status in a TSL. In particular, a Referenced Token that is evaluated as being expired (e.g. through the exp claim) but in a TSL has a status of 0x00 ("VALID"), is considered expired.

This document creates a registry in Section 14.5 that includes the most common Status Type values. Applications SHOULD use registered values for statuses if they have the correct semantics. Additional values may be defined for particular use cases. Status Types described by this document comprise:

* 0x00 - "VALID" - The status of the Referenced Token is valid, correct or legal.

* 0x01 - "INVALID" - The status of the Referenced Token is revoked, annulled, taken back, recalled or cancelled.

* 0x02 - "SUSPENDED" - The status of the Referenced Token is temporarily invalid, hanging, debarred from privilege. This status is usually temporary.

The Status Type value 0x03 and Status Type values in the range 0x0C until 0x0F are permanently reserved as application specific. The processing of Status Types using these values is application specific. All other Status Type values are reserved for future registration.

See Section 12.8 for privacy considerations on status types.

8. Verification and Processing

The fetching, processing and verifying of a Status List Token may be done by either the Holder or the Relying Party. The following section is described from the role of the Relying Party, however the same rules apply to the Holder.

8.1. Status List Request

The Status Provider SHOULD provide Status List Token on an endpoint serving an HTTP GET request to the URI provided in the Referenced Token, unless the Relying Party and the Status Provider have alternative methods of distribution.

The HTTP endpoint SHOULD support the use of Cross-Origin Resource Sharing (CORS) [CORS] and/or other methods as appropriate to enable Browser-based clients to access it, unless ecosystems using this specification choose not to support Browser-based clients.

The Relying Party SHOULD send the following Accept HTTP Header to indicate the requested response type unless the Content-Type of Status List Tokens in the respective ecosystem is known or the Relying Party supports both formats:

* "application/statuslist+jwt" for Status List Token in JWT format

* "application/statuslist+cwt" for Status List Token in CWT format

If the Relying Party does not send an Accept Header, the response type is assumed to be known implicitly or out-of-band.

The following is a non-normative example of a request for a Status List Token with type application/statuslist+jwt:

```
GET /statuslists/1 HTTP/1.1
Host: example.com
Accept: application/statuslist+jwt
```

8.2. Status List Response

A successful response that contains a Status List Token MUST use an HTTP status code in the 2xx range.

A response MAY also choose to redirect the client to another URI using an HTTP status code in the 3xx range, which clients SHOULD follow. See Section 11.4 for security considerations on redirects.

In the successful response, the Status Provider MUST use the following content-type:

* "application/statuslist+jwt" for Status List Token in JWT format

* "application/statuslist+cwt" for Status List Token in CWT format

In the case of "application/statuslist+jwt", the response MUST be of type JWT and follow the rules of Section 5.1. In the case of "application/statuslist+cwt", the response MUST be of type CWT and follow the rules of Section 5.2.

The body of such an HTTP response contains the raw Status List Token, that means the binary encoding as defined in Section 9.2.1 of [RFC8392] for a Status List Token in CWT format and the JWS Compact

Serialization form for a Status List Token in JWT format. Note that while the examples for Status List Tokens in CWT format in this document are provided in hex encoding, this is done purely for readability; CWT format response bodies are "in binary".

The HTTP response SHOULD use gzip Content-Encoding as defined in [RFC9110] for Status List Tokens in JWT format.

If caching-related HTTP headers are present in the HTTP response, Relying Parties MUST prioritize the exp and ttl claims within the Status List Token over the HTTP headers for determining caching behavior.

The following is a non-normative example of a response with a Status List Token with type application/statuslist+jwt:

```
HTTP/1.1 200 OK
Content-Type: application/statuslist+jwt

eyJhbGciOiJFUzI1NiIsImtpZCI6IjEyIiwidHlwIjoic3RhdHVzbGlzdCtqd3QifQ.e
yJleHAiOjIyOTE3MjAxNzAsImlhdCI6MTY4NjkyMDE3MCwiaXNzIjoiaHR0cHM6Ly9le
GFtcGxlLmNvbSIsInN0YXR1c19saXN0Ijp7ImJpdHMiOjEsImxzdCI6ImVOcmJ1UmdBQ
WhjQlhRIn0sInN1YiI6Imh0dHBzOi8vZXhhbXBsZS5jb20vc3RhdHVzbGlzdHMvMSIsI
nR0bCI6NDMyMDB9.2lKUUNG503R9htu4aHAYi7vjmr3sgApbfoDvPrl65N3URUO1EYqq
Ql45Jfzd-Av4QzlKa3oVALpLwOEUOq-U_g
```

8.3. Validation Rules

Upon receiving a Referenced Token, a Relying Party MUST first perform the validation of the Referenced Token - e.g., checking for expected attributes, valid signature and expiration time. The processing rules for Referenced Tokens (such as JWT or CWT) MUST precede any evaluation of a Referenced Token’s status. For example, if a token is evaluated as being expired through the "exp" (Expiration Time) but also has a status of 0x00 ("VALID"), the token is considered expired. As this is out of scope for this document, this validation is not described here, but is expected to be done according to the format of the Referenced Token.

If this validation is not successful, the Referenced Token MUST be rejected. If the validation was successful, the Relying Party MUST perform the following validation steps to evaluate the status of the Referenced Token:

1. Check for the existence of a status claim, check for the existence of a status_list claim within the status claim and validate that the content of status_list adheres to the rules defined in Section 6.2 for JOSE-based Referenced Tokens and Section 6.3 for COSE-based Referenced Tokens. Other formats of Referenced Tokens may define other encoding of the URI and index.

2. Resolve the Status List Token from the provided URI

3. Validate the Status List Token:

    a. Validate the Status List Token by following the rules defined in Section 7.2 of [RFC7519] for JWTs and Section 7.2 of [RFC8392] for CWTs. This step might require the resolution of a public key as described in Section 11.3.

    b. Check for the existence of the required claims as defined in Section 5.1 and Section 5.2 depending on the token type

4. All existing claims in the Status List Token MUST be checked according to the rules in Section 5.1 and Section 5.2

    a. The subject claim (sub or 2) of the Status List Token MUST be equal to the uri claim in the status_list object of the Referenced Token

    b. If the Relying Party has local policies regarding the freshness of the Status List Token, it SHOULD check the issued at claim (iat or 6)

    c. If the expiration time is defined (exp or 4), it MUST be checked if the Status List Token is expired

    d. If the Relying Party is using a system for caching the Status List Token, it SHOULD check the ttl claim of the Status List Token and retrieve a fresh copy if (time status was resolved + ttl < current time)

5. Decompress the Status List with a decompressor that is compatible with DEFLATE [RFC1951] and ZLIB [RFC1950]

6. Retrieve the status value of the index specified in the Referenced Token as described in Section 4. If the provided index is out of bounds of the Status List, no statement about the status of the Referenced Token can be made and the Referenced Token SHOULD be rejected.

7. Check the status value as described in Section 7

If any of these checks fails, no statement about the status of the Referenced Token can be made and the Referenced Token SHOULD be rejected.

8.4. Historical Resolution

By default, the status mechanism defined in this specification only conveys information about the state of Referenced Tokens at the time the Status List Token was issued. The validity period for this information, as defined by the issuer, is explicitly stated by the iat (issued at) and exp (expiration time) claims for JWT and their corresponding ones for the CWT representation. If support for historical status information is desired, this can be achieved by extending with a timestamp the request for the Status List Token as defined in Section 8.1. This feature has additional privacy implications as described in Section 12.7.

To obtain the Status List Token, the Relying Party MUST send an HTTP GET request to the URI provided in the Referenced Token with the additional query parameter time and its value being a unix timestamp. The response for a valid request SHOULD contain a Status List Token that was valid for that specified time or an error.

If the Server does not support the additional query parameter, it SHOULD return a status code of 501 (Not Implemented) or if the requested time is not supported it SHOULD return a status code of 406 (Not Acceptable). A Status List Token might be served via static file hosting (e.g., leveraging a Content Delivery Network) that ignores query parameters, which would result in the client requesting a historical status list but receiving the current status list. Thus, the client MUST reject a response unless the requested timestamp is within the valid time of the returned token signaled via iat (6 for CWT) and exp (4 for CWT).

The following is a non-normative example of a GET request using the time query parameter:

```
GET /statuslists/1?time=1686925000 HTTP/1.1
Host: example.com
Accept: application/statuslist+jwt
```

The following is a non-normative example of a response for the above Request:

```
HTTP/1.1 200 OK
Content-Type: application/statuslist+jwt

eyJhbGciOiJFUzI1NiIsImtpZCI6IjEyIiwidHlwIjoic3RhdHVzbGlzdCtqd3QifQ.e
yJleHAiOjIyOTE3MjAxNzAsImlhdCI6MTY4NjkyMDE3MCwiaXNzIjoiaHR0cHM6Ly9le
GFtcGxlLmNvbSIsInN0YXR1c19saXN0Ijp7ImJpdHMiOjEsImxzdCI6ImVOcmJ1UmdBQ
WhjQlhRIn0sInN1YiI6Imh0dHBzOi8vZXhhbXBsZS5jb20vc3RhdHVzbGlzdHMvMSIsI
nR0bCI6NDMyMDB9.2lKUUNG503R9htu4aHAYi7vjmr3sgApbfoDvPrl65N3URUO1EYqq
Ql45Jfzd-Av4QzlKa3oVALpLwOEUOq-U_g
```

9. Status List Aggregation

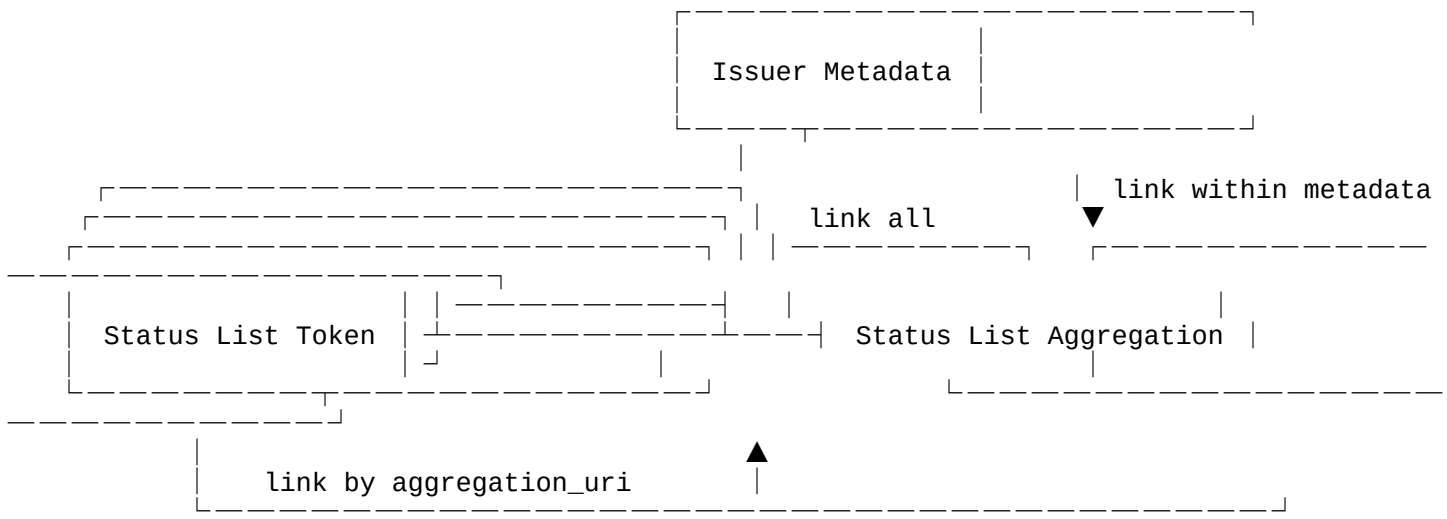
Status List Aggregation is an optional mechanism offered by the Issuer to publish a list of one or more Status List Tokens URIs, allowing a Relying Party to fetch Status List Tokens provided by this Issuer. This mechanism is intended to support fetching and caching mechanisms and allow offline validation of the status of a Referenced Token for a period of time.

If a Relying Party encounters an error while validating one of the Status List Tokens returned from the Status List Aggregation endpoint, it SHOULD continue processing the other Status List Tokens.

There are two options for a Relying Party to retrieve the Status List Aggregation. An Issuer MAY support any of these mechanisms:

* Issuer metadata: The Issuer of the Referenced Token publishes an URI which links to Status List Aggregation, e.g. in publicly available metadata of an issuance protocol

* Status List Parameter: The Status Issuer includes an additional claim in the Status List Token that contains the Status List Aggregation URI.



9.1. Issuer Metadata

The Issuer MAY link to the Status List Aggregation URI in metadata that can be provided by different means like .well-known metadata as is used commonly in OAuth and OpenID Connect, or within Issuer certificates or trust lists (such as VICAL as defined in Annex C of [ISO.mdoc]). If the Issuer is an OAuth Authorization Server according to [RFC6749], it is RECOMMENDED to use the status_list_aggregation_endpoint parameter within its metadata defined by [RFC8414]. The Issuer MAY limit the Status List Tokens listed by a Status List Aggregation to a particular type of Referenced Token.

The concrete implementation details depend on the specific ecosystem and are out of scope of this specification.

9.2. Status List Parameter

The URI to the Status List Aggregation MAY be provided as the optional parameter aggregation_uri in the Status List itself as explained in Section 4.3 and Section 4.2 respectively. A Relying Party may use this URI to retrieve an up-to-date list of relevant Status Lists.

9.3. Status List Aggregation Data Structure

This section defines the structure for a JSON-encoded Status List Aggregation:

* status_lists: REQUIRED. JSON array of strings that contains URIs linking to Status List Tokens.

The Status List Aggregation URI provides a list of Status List Token URIs. This aggregation is in JSON and the returned media type MUST be application/json. A Relying Party can iterate through this list and fetch all Status List Tokens before encountering the specific URI in a Referenced Token.

The following is a non-normative example for media type application/json:

```
{
   "status_lists" : [
      "https://example.com/statuslists/1",
      "https://example.com/statuslists/2",
      "https://example.com/statuslists/3"
   ]
}
```

10. X.509 Certificate Extended Key Usage Extension

[RFC5280] specifies the Extended Key Usage (EKU) X.509 certificate extension for use on end entity certificates. The extension indicates one or more purposes for which the certified public key is valid. The EKU extension can be used in conjunction with the Key Usage (KU) extension, which indicates the set of basic cryptographic operations for which the certified key may be used. A certificate’s issuer explicitly delegates Status List Token signing authority by issuing an X.509 certificate containing the KeyPurposeId defined below in the extended key usage extension. Other specifications MAY choose to re-use this OID for other status mechanisms under the condition that they are registered in the "JWT Status Mechanisms" or "CWT Status Mechanisms" registries.

The following OID is defined for usage in the EKU extension:

```
id-kp  OBJECT IDENTIFIER  ::=
     { iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) kp(3) }

id-kp-oauthStatusSigning OBJECT IDENTIFIER ::= { id-kp TBD }
```

11. Security Considerations

Status List Tokens as defined in Section 5 only exist in cryptographically secured containers which allow checking the integrity and origin without relying on other factors such as transport security or web PKI.

11.1. Correct decoding and parsing of the encoded Status List

Implementers should be particularly careful for the correct parsing and decoding of the Status List. Incorrect implementations might check the index on the wrong data or miscalculate the bit and byte index leading to an erroneous status of the Referenced Token. Beware, that bits are indexed (bit order) from least significant bit to most significant bit (also called "right to left") while bytes are indexed (byte order) in their natural incrementing byte order (usually written for display purpose from left to right). Endianness does not apply here because each status value fits within a single byte.

Implementations SHOULD verify correctness using the test vectors given by this specification.

11.2. Security Guidance for JWT and CWT

A Status List Token in the JWT format SHOULD follow the security considerations of [RFC7519] and the best current practices of [RFC8725].

A Status List Token in the CWT format SHOULD follow the security considerations of [RFC8392].

11.3. Key Resolution and Trust Management

This specification does not mandate specific methods for key resolution and trust management, however the following recommendations are made for specifications, profiles, or ecosystems that are planning to make use of the Status List mechanism:

If the Issuer of the Referenced Token is the same entity as the Status Issuer, then the same key that is embedded into the Referenced Token may be used for the Status List Token. In this case the Status List Token may use:

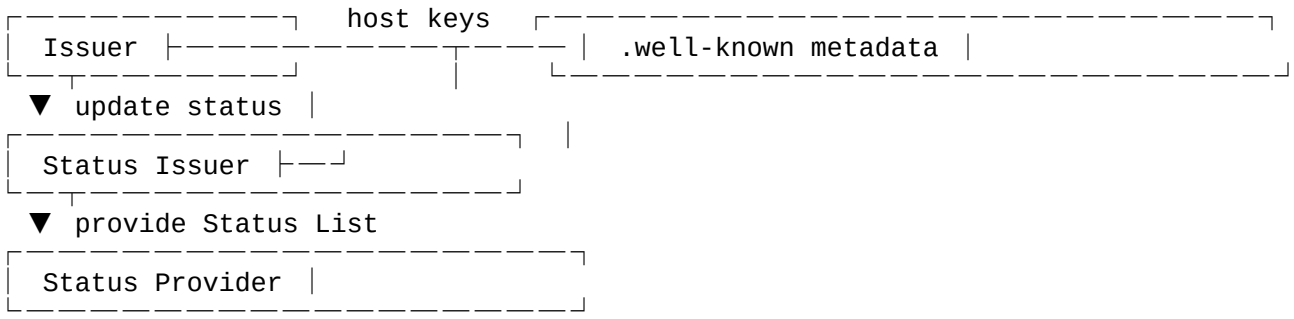
* the same x5c value or an x5t, x5t#S256 or kid parameter referencing to the same key as used in the Referenced Token for JOSE.

* the same x5chain value or an x5t or kid parameter referencing to the same key as used in the Referenced Token for COSE.

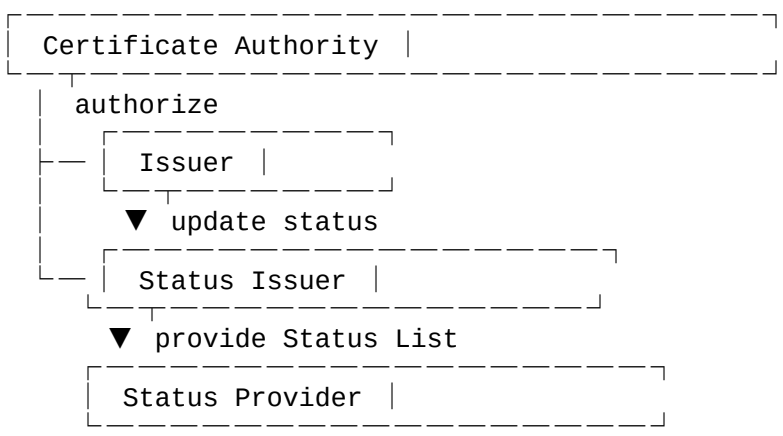
Alternatively, the Status Issuer may use the same web-based key resolution that is used for the Referenced Token. In this case the Status List Token may use:

* an x5u, jwks, jwks_uri or kid parameter referencing to a key using the same web-based resolution as used in the Referenced Token for JOSE.

* an x5u or kid parameter referencing to a key using the same web-based resolution as used in the Referenced Token for COSE.



If the Issuer of the Referenced Token is a different entity than the Status Issuer, then the keys used for the Status List Token may be cryptographically linked, e.g. by a Certificate Authority through an x.509 PKI. The certificate of the Issuer for the Referenced Token and the Status Issuer should be issued by the same Certificate Authority and the Status Issuer's certificate should utilize extended key usage (Section 10).



11.4. Redirection 3xx

HTTP clients that follow 3xx (Redirection) class of status codes SHOULD be aware of the possible dangers of redirects, such as infinite redirection loops, since they can be used for denial of service attacks on clients. A client SHOULD detect and intervene in infinite redirections. Clients SHOULD apply the guidance for redirects given in Section 15.4 of [RFC9110].

11.5. Expiration and Caching

Expiration and caching information is conveyed via the exp and ttl claims as explained in Section 13.7. Clients SHOULD check that both values are within reasonable ranges before requesting new Status List Tokens based on these values to prevent accidentally creating unreasonable amounts of requests for a specific URL. Status Issuers could accidentally or maliciously use this mechanism to effectively DDoS the contained URL of the Status Provider.

Reasonable values for both claims highly depend on the use-case requirements and clients SHOULD be configured with lower/upper bounds for these values that fit their respective use-cases.

11.6. Status List Token Protection

This specification allows both, digital signatures using asymmetric cryptography, and Message Authentication Codes (MAC) to be used to protect Status List Tokens. Implementers SHOULD only use MACs to secure the integrity of Status List Tokens if they fully understand the risks of MACs when compared to digital signatures and especially the requirements of their use-case scenarios. These use-cases typically represent deployments where Status Issuer and Relying Party have a trust relationship and the possibility to securely exchange keys out of band or are the same entity and no other entity needs to verify the Status List Token. We expect most deployments to use digital signatures for the protection of Status List Tokens and implementers SHOULD default to digital signatures if they are unsure.

12. Privacy Considerations

12.1. Observability of Issuers

The main privacy consideration for a Status List, especially in the context of the Issuer-Holder-Verifier model [SD-JWT.VC], is to prevent the Issuer from tracking the usage of the Referenced Token when the status is being checked. If an Issuer offers status information by referencing a specific token, this would enable the Issuer to create a profile for the issued token by correlating the

date and identity of Relying Parties, that are requesting the status.

The Status List approaches these privacy implications by integrating the status information of many Referenced Tokens into the same list. Therefore, the Issuer does not learn for which Referenced Token the Relying Party is requesting the Status List. The privacy of the Holder is protected by the anonymity within the set of Referenced Tokens in the Status List, also called herd privacy. This limits the possibilities of tracking by the Issuer.

The herd privacy is depending on the number of entities within the Status List called its size. A larger size results in better privacy but also impacts the performance as more data has to be transferred to read the Status List.

Additionally, the Issuer may analyse data from the HTTP request to identify the Relying Party, e.g. through the sender’s IP address.

This behaviour may be mitigated by:

* private relay protocols or other mechanisms hiding the original sender like [RFC9458].

* using trusted Third Party Hosting, see Section 12.6.

12.2. Issuer Tracking of Referenced Tokens

An Issuer could maliciously or accidentally bypass the privacy benefits of the herd privacy by either:

* Generating a unique Status List for every Referenced Token. By these means, the Issuer could maintain a mapping between Referenced Tokens and Status Lists and thus track the usage of Referenced Tokens by utilizing this mapping for the incoming requests.

* Encoding a unique URI in each Referenced Token which points to the underlying Status List. This may involve using URI components such as query parameters, unique path segments, or fragments to make the URI unique.

This malicious behavior can be detected by Relying Parties that request large amounts of Referenced Tokens by comparing the number of different Status Lists and their sizes with the volume of Referenced Tokens being verified.

12.3. Observability of Relying Parties

Once the Relying Party receives the Referenced Token, the Relying Party can request the Status List through the provided uri parameter and can validate the Referenced Token’s status by looking up the corresponding index. However, the Relying Party may persistently store the uri and index of the Referenced Token to request the Status List again at a later time. By doing so regularly, the Relying Party may create a profile of the Referenced Token’s validity status. This behaviour may be intended as a feature, e.g. for an identity proofing (e.g. Know-Your-Customer process in finance industry) that requires regular validity checks, but might also be abused in cases where this is not intended and unknown to the Holder, e.g. profiling the suspension of an employee credential.

This behaviour could be mitigated by:

* regular re-issuance of the Referenced Token, see Section 13.2.

12.4. Observability of Outsiders

Outside actors may analyse the publicly available Status Lists to get information on the internal processes of the Issuer and its related business, e.g. number of customers or clients. This data may allow inferences on the total number of issued Referenced Tokens and the revocation rate. Additionally, actors may regularly fetch this data or use the historic data functionality to learn how these numbers change over time.

This behaviour could be mitigated by:

* disabling the historical data feature Section 8.4

* disabling the Status List Aggregation Section 9

* choosing non-sequential, pseudo-random or random indices

* using decoy entries to obfuscate the real number of Referenced Tokens within a Status List

* choosing to deploy and utilize multiple Status Lists simultaneously

12.5. Unlinkability

The tuple of uri and index inside the Referenced Token are unique and therefore is traceable data.

12.5.1. Colluding Relying Parties

Two or more colluding Relying Parties may link two transactions involving the same Referenced Token by comparing the status claims of received Referenced Tokens and therefore determine that they have interacted with the same Holder.

To avoid privacy risks of colluding Relying Parties, it is RECOMMENDED that Issuers provide the ability to issue batches of one-time-use Referenced Tokens, enabling Holders to use in a single interaction with a Relying Party before discarding. See Section 13.2 to avoid further correlatable information by the values of uri and idx, Status Issuers are RECOMMENDED to:

* choose non-sequential, pseudo-random or random indices
* use decoy entries to obfuscate the real number of Referenced Tokens within a Status List
* choose to deploy and utilize multiple Status Lists simultaneously

12.5.2. Colluding Status Issuer and Relying Party

A Status Issuer and a Relying Party Issuer may link two transactions involving the same Referenced Tokens by comparing the status claims of the Referenced Token and therefore determine that they have interacted with the same Holder. It is therefore recommended to use Status Lists for Referenced Token formats that have similar unlinkability properties.

12.6. External Status Provider for Privacy

If the roles of the Status Issuer and the Status Provider are performed by different entities, this may give additional privacy assurances as the Issuer has no means to identify the Relying Party or its request.

Third-Party hosting may also allow for greater scalability, as the Status List Tokens may be served by operators with greater resources, like CDNs, while still ensuring authenticity and integrity of Token Status List, as it is signed by the Status Issuer.

12.7. Historical Resolution

By default, this specification only supports providing Status List information for the most recent status information and does not allow the lookup of historical information like a validity state at a specific point in time. There exists optional support for a query parameter that allows these kind of historic lookups as described in Section 8.4. There are scenarios where such a functionality is necessary, but this feature should only be implemented when the scenario and the consequences of enabling historical resolution are fully understood.

There are strong privacy concerns that have to be carefully taken into consideration when providing a mechanism that allows historic requests for status information - see Section 12.3 for more details. Support for this functionality is optional and Implementers are RECOMMENDED to not support historic requests unless there are strong reasons to do so and after carefully considering the privacy implications.

12.8. Status Types

As previously explained, there is the potential risk of observability by Relying Parties (see Section 12.3) and Outsiders (see Section 12.4). That means that any Status Type that transports information beyond the routine statuses VALID and INVALID about a Referenced Token can leak information to other parties. This document defines one additional Status Type with "SUSPENDED" that conveys such additional information, but in practice all statuses other than VALID and INVALID are likely to contain information with privacy implications.

Ecosystems that want to use other Status Types than "VALID" and "INVALID" should consider the possible leakage of data and profiling possibilities before doing so and evaluate if revocation and re-issuance might a better fit for their use-case.

13. Operational Considerations

13.1. Token Lifecycle

The lifetime of a Status List Token depends on the lifetime of its Referenced Tokens. Once all Referenced Tokens are expired, the Issuer may stop serving the Status List Token.

13.2. Linkability Mitigation

Referenced Tokens may be regularly re-issued to mitigate the linkability of presentations to Relying Parties. In this case, every re-issued Referenced Token MUST have a fresh Status List entry in order to prevent the index value from becoming a possible source of correlation.

Referenced Tokens may also be issued in batches and be presented by Holders in a one-time-use policy to avoid linkability. In this case, every Referenced Token MUST have a dedicated Status List entry and MAY be spread across multiple Status List Tokens. Batch revocation of a batch of Referenced Tokens might reveal that they are all members of the same batch.

Beware, that this mechanism solves linkability issues between Relying Parties, but does not prevent traceability by Issuers.

13.3. Default Values and Double Allocation

The Status Issuer is RECOMMENDED to initialize the Status List byte array with a default value provided as an initialization parameter by the Issuer of the Referenced Token. The Issuer is RECOMMENDED to use a default value that represents the most common value for its Referenced Tokens to avoid an update during issuance (usually 0x00, VALID). This preserves the benefits from compression and effectively hides the number of managed Referenced Tokens since an unused index value can not be distinguished from a valid Referenced Token.

The Status Issuer is RECOMMENDED to prevent double allocation, i.e. re-using the same uri and idx for multiple Referenced Tokens (since uri and idx form a unique identifier that might be used for tracking, see Section 12 for more details). The Status Issuer MUST prevent any unintended double allocation.

13.4. Status List Size

The storage and transmission size of the Status Issuer’s Status List Tokens depend on:

* the size of the Status List, i.e. the number of Referenced Tokens
* the revocation rate and distribution of the Status List data (due to compression, revocation rates close to 0% or 100% create lowest sizes while revocation rates closer to 50% and random distribution create highest sizes)

* the lifetime of Referenced Tokens (shorter lifetimes allows for earlier retirement of Status List Tokens)

The Status List Issuer may increase the size of a Status List if it requires indices for additional Referenced Tokens. It is RECOMMENDED that the size of a Status List in bits is divisible in bytes (8 bits) without a remainder, i.e. size-in-bits % 8 = 0.

The Status List Issuer may divide its Referenced Tokens up into multiple Status Lists to reduce the transmission size of an individual Status List Token. This may be useful for ecosystems where some entities operate in constrained environments, e.g. for mobile internet or embedded devices. The Status List Issuer may organize the Status List Tokens depending on the Referenced Token’s expiry date to align their lifecycles and allow for easier retiring of Status List Tokens, however the Status Issuer must be aware of possible privacy risks due to correlations.

13.5. External Status Issuer

If the roles of the Issuer of the Referenced Token and the Status Issuer are performed by different entities, this may allow for use cases that require revocation of Referenced Tokens to be managed by different entities, e.g. for regulatory or privacy reasons. In this scenario both parties must align on:

* the key and trust management as described in Section 11.3
* parameters for the Status List
  - number of bits for the Status Type as described in Section 4
  - update cycle of the Issuer used for ttl in the Status List Token as described in Section 5

13.6. External Status Provider for Scalability

If the roles of the Status Issuer and the Status Provider are performed by different entities, this may allow for greater scalability, as the Status List Tokens may be served by operators with greater resources, like CDNs. At the same time the authenticity and integrity of Token Status List is still guaranteed, as it is signed by the Status Issuer.

13.7. Status List Update Interval and Caching

Status Issuers have two options to communicate their update interval policy for the status of their Referenced Tokens:

* the exp claim specifies an absolute timestamp, marking the point in time when the Status List expires and MUST NOT be relied upon any longer

* the ttl claim represents a duration to be interpreted relative to the time the Status List is fetched, indicating when a new version of the Status List may be available

Both ttl and exp are RECOMMENDED to be used by the Status Issuer.

When fetching a Status List Token, Relying Parties must carefully evaluate how long a Status List is cached for. Collectively the iat, exp and ttl claims when present in a Status List Token communicate how long a Status List should be cached and should be considered valid for. Relying Parties have different options for caching the Status List:

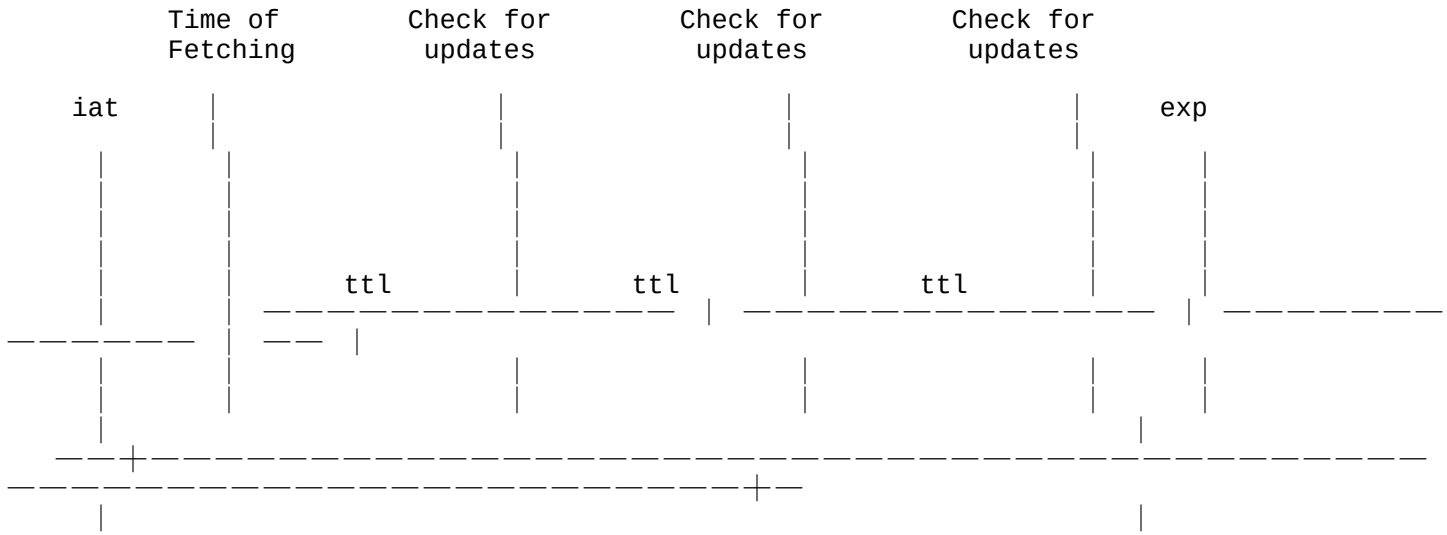
* After time of fetching, the Relying Party caches the Status List for time duration of ttl before making checks for updates. This method is RECOMMENDED to distribute the load for the Status Provider.

* After initial fetching, the Relying Party checks for updates at time of iat + ttl. This method ensures the most up-to-date information for critical use cases. The Relying Party should account a minimal offset due to the signing and distribution process of the Status Issuer.

* If no ttl is given, then Relying Party SHOULD check for updates latest after the time of exp.

Ultimately, it’s the Relying Parties decision how often to check for updates, ecosystems may define their own guidelines and policies for updating the Status List information. Clients should ensure that exp and ttl are within reasonable bounds before creating requests to get a fresh Status List Token (see Section 11.5 for more details).

The following diagram illustrates the relationship between these claims and how they are designed to influence caching:



13.8. Relying Parties avoiding correlatable Information

If the Relying Party does not require the Referenced Token or the Status List Token for further processing, it is RECOMMENDED to delete correlatable information, in particular:

* the status claim in the Referenced Token (after the validation)

* the Status List Token itself (after expiration or update)

The Relying Party should instead only keep the needed fields from the Referenced Token.

13.9. Status List Formats

This specification defines 2 different token formats of the Status List:

* JWT

* CWT

This specification states no requirements to not mix different formats like a CBOR based Referenced Token using a JWT for the Status List, but the expectation is that within an ecosystem, a choice for specific formats is made. Within such an ecosystem, only support for those selected variants is required and implementations should know what to expect via a profile.

14. IANA Considerations

14.1. JSON Web Token Claims Registration

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

14.1.1. Registry Contents

* Claim Name: status
* Claim Description: A JSON object containing a reference to a status mechanism from the JWT Status Mechanisms Registry.
* Change Controller: IETF
* Specification Document(s): Section 6.1 of this specification

* Claim Name: status_list
* Claim Description: A JSON object containing up-to-date status information on multiple tokens using the Token Status List mechanism.
* Change Controller: IETF
* Specification Document(s): Section 5.1 of this specification

* Claim Name: ttl
* Claim Description: Time to Live
* Change Controller: IETF
* Specification Document(s): Section 5.1 of this specification

14.2. JWT Status Mechanisms Registry

This specification establishes the IANA "JWT Status Mechanisms" registry for JWT "status" member values and adds it to the "JSON Web Token (JWT)" registry group at https://www.iana.org/assignments/jwt. The registry records the status mechanism member and a reference to the specification that defines it.

JWT Status Mechanisms are registered by Specification Required [RFC8126] after a three-week review period on the jwt-reg-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of names prior to publication, the Designated Expert(s) may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register JWT Status Mechanism: example").

Within the review period, the Designated Expert(s) will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful.

IANA must only accept registry updates from the Designated Expert(s) and should direct all requests for registration to the review mailing list.

14.2.1. Registration Template

Status Mechanism Value:

The name requested (e.g., "status_list"). The name is case sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Experts state that there is a compelling reason to allow an exception.

Status Mechanism Description:

Brief description of the status mechanism.

Change Controller:

For IETF Stream RFCs, list the IETF. For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):

Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

14.2.2. Initial Registry Contents

* Status Mechanism Value: status_list
* Status Mechanism Description: A Token Status List containing up-to-date status information on multiple tokens.
* Change Controller: IETF
* Specification Document(s): Section 6.2 of this specification

14.3. CBOR Web Token Claims Registration

This specification requests registration of the following Claims in the IANA "CBOR Web Token (CWT) Claims" registry [IANA.CWT] established by [RFC8392].

14.3.1. Registry Contents

* Claim Name: status
* Claim Description: A CBOR structure containing a reference to a status mechanism from the CWT Status Mechanisms Registry.
* JWT Claim Name: status
* Claim Key: TBD (requested assignment 65535)
* Claim Value Type: map
* Change Controller: IETF
* Reference: Section 6.1 of this specification

* Claim Name: status_list
* Claim Description: A CBOR structure containing up-to-date status information on multiple tokens using the Token Status List mechanism.
* JWT Claim Name: status_list
* Claim Key: TBD (requested assignment 65533)

* Claim Value Type: map
* Change Controller: IETF
* Specification Document(s): Section 5.2 of this specification

* Claim Name: ttl
* Claim Description: Time to Live
* JWT Claim Name: ttl
* Claim Key: TBD (requested assignment 65534)
* Claim Value Type: unsigned integer
* Change Controller: IETF
* Specification Document(s): Section 5.2 of this specification

14.4. CWT Status Mechanisms Registry

This specification establishes the IANA "CWT Status Mechanisms" registry for CWT "status" member values and adds it to the "CBOR Web Token (CWT) Claims" registry group at https://www.iana.org/assignments/cwt. The registry records the status mechanism member and a reference to the specification that defines it.

CWT Status Mechanisms are registered by Specification Required [RFC8126] after a three-week review period on the cwt-reg-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of names prior to publication, the Designated Expert(s) may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register CWT Status Mechanism: example").

Within the review period, the Designated Expert(s) will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful.

IANA must only accept registry updates from the Designated Expert(s) and should direct all requests for registration to the review mailing list.

14.4.1. Registration Template

Status Mechanism Value:

The name requested (e.g., "status_list"). The name is case sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Experts state that there is a compelling reason to allow an exception.

Status Mechanism Description:

Brief description of the status mechanism.

Change Controller:

For IETF Stream RFCs, list the IETF. For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):

Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

14.4.2. Initial Registry Contents

* Status Mechanism Value: status_list
* Status Mechanism Description: A Token Status List containing up-to-date status information on multiple tokens.
* Change Controller: IETF
* Specification Document(s): Section 6.3 of this specification

14.5. OAuth Status Types Registry

This specification establishes the IANA "OAuth Status Types" registry for Status List values and adds it to the "OAuth Parameters" registry group at https://www.iana.org/assignments/oauth-parameters. The registry records a human readable label, the bit representation and a common description for it.

Status Types are registered by Specification Required [RFC8126] after a two-week review period on the oauth-ext-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of names prior to publication, the Designated Expert(s) may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register Status Type name: example").

Within the review period, the Designated Expert(s) will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful.

IANA must only accept registry updates from the Designated Expert(s) and should direct all requests for registration to the review mailing list.

14.5.1. Registration Template

Status Type Name:

The name is a human-readable case insensitive label for the Status Type that helps to talk about the status of Referenced Token in common language.

Status Type Description:

Brief description of the Status Type and optional examples.

Status Type value:

The bit representation of the Status Type in a byte hex representation. Valid Status Type values range from 0x00-0xFF. Values are filled up with zeros if they have less than 8 bits.

Change Controller:

For IETF Stream RFCs, list the IETF. For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):

Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

14.5.2. Initial Registry Contents

* Status Type Name: VALID
* Status Type Description: The status of the Referenced Token is valid, correct or legal.
* Status Type value: 0x00
* Change Controller: IETF
* Specification Document(s): Section 7 of this specification

* Status Type Name: INVALID
* Status Type Description: The status of the Referenced Token is revoked, annulled, taken back, recalled or cancelled.
* Status Type value: 0x01
* Change Controller: IETF
* Specification Document(s): Section 7 of this specification

* Status Type Name: SUSPENDED
* Status Type Description: The status of the Referenced Token is temporarily invalid, hanging or debarred from privilege. This state is usually temporary.
* Status Type value: 0x02
* Change Controller: IETF
* Specification Document(s): Section 7 of this specification

* Status Type Name: APPLICATION_SPECIFIC

* Status Type Description: The status of the Referenced Token is application specific.

* Status Type value: 0x03

* Change Controller: IETF

* Specification Document(s): Section 7 of this specification

* Status Type Name: APPLICATION_SPECIFIC

* Status Type Description: The status of the Referenced Token is application specific.

* Status Type value: 0x0C-0xOF

* Change Controller: IETF

* Specification Document(s): Section 7 of this specification

14.6. OAuth Parameters Registration

This specification requests registration of the following values in the IANA "OAuth Authorization Server Metadata" registry [IANA.OAuth.Params] established by [RFC8414].

* Metadata Name: status_list_aggregation_endpoint

* Metadata Description: URL of the Authorization Server aggregating OAuth Token Status List URLs for token status management.

* Change Controller: IESG

* Reference: Section 9 of this specification

14.7. Media Type Registration

This section requests registration of the following media types [RFC2046] in the "Media Types" registry [IANA.MediaTypes] in the manner described in [RFC6838].

To indicate that the content is an JWT-based Status List:

* Type name: application

* Subtype name: statuslist+jwt
* Required parameters: n/a
* Optional parameters: n/a
* Encoding considerations: See Section 5.1 of this specification
* Security considerations: See Section 11 of this specification
* Interoperability considerations: n/a
* Published specification: this specification
* Applications that use this media type: Applications using this specification for updated status information of tokens
* Fragment identifier considerations: n/a
* Additional information: n/a
* Person & email address to contact for further information: Paul Bastian, paul.bastian@posteo.de
* Intended usage: COMMON
* Restrictions on usage: none
* Author: Paul Bastian, paul.bastian@posteo.de
* Change controller: IETF
* Provisional registration? No

To indicate that the content is an CWT-based Status List:

* Type name: application
* Subtype name: statuslist+cwt
* Required parameters: n/a
* Optional parameters: n/a
* Encoding considerations: See Section 5.2 of this specification
* Security considerations: See Section 11 of this specification

* Interoperability considerations: n/a

* Published specification: this specification

* Applications that use this media type: Applications using this specification for updated status information of tokens

* Fragment identifier considerations: n/a

* Additional information: n/a

* Person & email address to contact for further information: Paul Bastian, paul.bastian@posteo.de

* Intended usage: COMMON

* Restrictions on usage: none

* Author: Paul Bastian, paul.bastian@posteo.de

* Change controller: IETF

* Provisional registration? No

14.8. CoAP Content-Format Registrations

IANA is requested to register the following Content-Format numbers in the "CoAP Content-Formats" sub-registry, within the "Constrained RESTful Environments (CoRE) Parameters" Registry [IANA.Core.Params]:

* Content Type: application/statuslist+cwt

* Content Coding: -

* ID: TBD

* Reference: this specification

14.9. X.509 Certificate Extended Key Purpose OID Registration

IANA is requested to register the following OID "1.3.6.1.5.5.7.3.TBD" with a description of "id-kp-oauthStatusSigning" in the "SMI Security for PKIX Extended Key Purpose" registry (1.3.6.1.5.5.7.3). This OID is defined in section Section 10.

IANA is requested to register the following OID "1.3.6.1.5.5.7.0.TBD" with a description of "id-mod-oauth-status-signing-eku" in the "SMI Security for PKIX Module Identifier" registry (1.3.6.1.5.5.7.0). This OID is defined in section Appendix A.

15. Acknowledgments

We would like to thank Andrii Deinega, Brian Campbell, Dan Moore, Denis Pinkas, Filip Skokan, Francesco Marino, Giuseppe De Marco, Hannes Tschofenig, Kristina Yasuda, Markus Kreusch, Martijn Haring, Michael B. Jones, Micha Kraus, Michael Schwartz, Mike Prorock, Mirko Mollik, Oliver Terbu, Orie Steele, Rifaat Shekh-Yusef, Rohan Mahy, Takahiko Kawasaki, Timo Glastra and Torsten Lodderstedt

for their valuable contributions, discussions and feedback to this specification.

16. References

16.1. Normative References

[CORS] WHATWG, "Fetch Living Standard", n.d., <https://fetch.spec.whatwg.org/commit-snapshots/4775fcb48042c8411df497c0b7cf167b4240004f/#http-cors-protocol>.

[RFC1950] Deutsch, P. and J. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, DOI 10.17487/RFC1950, May 1996, <https://www.rfc-editor.org/rfc/rfc1950>.

[RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, DOI 10.17487/RFC1951, May 1996, <https://www.rfc-editor.org/rfc/rfc1951>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/rfc/rfc2046>.

[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/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[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/rfc/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/rfc/rfc7519>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9596] Jones, M.B. and O. Steele, "CBOR Object Signing and Encryption (COSE) "typ" (type) Header Parameter", RFC 9596, DOI 10.17487/RFC9596, June 2024, <https://www.rfc-editor.org/rfc/rfc9596>.

[X.680] International Telecommunications Union, "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", February 2021.

[X.690] International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", February 2021.

16.2. Informative References

[IANA.Core.Params] IANA, "Constrained RESTful Environments (CoRE) Parameters", n.d., <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml>.

[IANA.CWT] IANA, "CBOR Web Token (CWT) Claims", n.d., <https://www.iana.org/assignments/cwt/cwt.xhtml>.

[IANA.JWT] IANA, "JSON Web Token Claims", n.d., <https://www.iana.org/assignments/jwt/jwt.xhtml>.

[IANA.MediaTypes] IANA, "Media Types", n.d., <https://www.iana.org/assignments/media-types/media-types.xhtml>.

[IANA.OAuth.Params]
IANA, "OAuth Authorization Server Metadata", n.d., <https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml#authorization-server-metadata>.

[ISO.mdoc] ISO/IEC JTC 1/SC 17, "ISO/IEC 18013-5:2021 ISO-compliant driving licence", n.d..

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

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

[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/rfc/rfc7800>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/rfc/rfc8414>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC9458] Thomson, M. and C. A. Wood, "Oblivious HTTP", RFC 9458, DOI 10.17487/RFC9458, January 2024, <https://www.rfc-editor.org/rfc/rfc9458>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/rfc/rfc9562>.

[SD-JWT.VC]
Terbu, O., Fett, D., and B. Campbell, "SD-JWT-based Verifiable Credentials (SD-JWT VC)", Work in Progress, Internet-Draft, draft-ietf-oauth-sd-jwt-vc-13, 6 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-sd-jwt-vc-13>.

[smith2020let]
Smith, T., Dickinson, L., and K. Seamons, "Let's revoke: Scalable global certificate revocation", Network and Distributed Systems Security (NDSS) Symposium 2020 , n.d., <https://www.ndss-symposium.org/ndss-paper/lets-revoke-scalable-global-certificate-revocation/>.

[W3C.SL] Longley, D., Sporny, M., and O. Steele, "W3C Bitstring Status List v1.0", December 2024, <https://www.w3.org/TR/vc-bitstring-status-list/>.

Appendix A. ASN.1 Module

The following module adheres to ASN.1 specifications [X.680] and [X.690]. It defines the OID used for OAuth Status Mechanism Key Extended Key Usage.

```
<CODE BEGINS>
  OauthStatusSigning-EKU
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-oauth-status-signing-eku (TBD) }

  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN

  -- OID Arc

  id-kp OBJECT IDENTIFIER ::=
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) kp(3) }

  -- OAuth Extended Key Usage

  id-kp-oauthStatusSigning OBJECT IDENTIFIER ::= { id-kp TBD }

  END
<CODE ENDS>
```

Appendix B. Size Comparison

The following tables show a size comparison for a Status List (compressed byte array as defined in Section 4.1) and a compressed Byte Array of UUIDs [RFC9562] (as an approximation to the list of IDs of Referenced Tokens in a Certificate Revocation List). Readers must be aware that these are not sizes for complete Status List Tokens in JSON/CBOR nor Certificate Revocation Lists (CRLs), as they don't contain metadata, certificates, and signatures.

If no further metadata is provided in Status List Tokens or CRLs, then the size of Status Lists or arrays of Certificate ids (represented as UUIDs) will be the main factors deciding on the overall size of a Status List Token or CRL, respectively.

Size of Status Lists for varying amount of entries and revocation rates

Size	0.01%	0.1%	1%	2%	5%	10%	25%	50%	75%	100%
100k	81 B	252 B	1.4 KB	2.3 KB	4.5 KB	6.9 KB	10.2 KB	12.2 KB	10.2 KB	35 B
1M	442 B	2.2 KB	13.7 KB	23.0 KB	43.9 KB	67.6 KB	102.2 KB	122.1 KB	102.4 KB	144 B
10M	3.8 KB	21.1 KB	135.4 KB	230.0 KB	437.0 KB	672.9 KB	1023.4 KB	1.2 MB	1023.5 KB	1.2 KB
100M	38.3 KB	213.0 KB	1.3 MB	2.2 MB	4.3 MB	6.6 MB	10.0 MB	11.9 MB	10.0 MB	11.9 KB

Table 1: Status List Size examples for varying amount of entries and revocation rates

Size of compressed array of UUIDv4 (128-bit UUIDs) for varying amount of entries and revocation rates

This is a simple approximation of a CRL using an array of UUIDs without any additional metadata (128-bit UUID per revoked entry).

Size	0.01%	0.1%	1%	2%	5%	10%	25%	50%	75%	100%
100k	219 B	1.6 KB	15.4 KB	29.7 KB	78.1 KB	154.9 KB	392.9 KB	783.1 KB	1.1 MB	1.5 MB
1M	1.6 KB	16.4 KB	157.7 KB	310.4 KB	781 KB	1.5 MB	3.8 MB	7.6 MB	11.4 MB	15.3 MB
10M	15.3 KB	155.9 KB	1.5 MB	3.1 MB	7.6 MB	15.2 MB	38.2 MB	76.3 MB	114.4 MB	152.6 MB
100M	157.6 KB	1.5 MB	15.3 MB	30.5 MB	76.3 MB	152.6 MB	381.4 MB	762.9 MB	1.1 GB	1.5 GB

Table 2: Size examples for 128-bit UUIDs for varying amount of entries and revocation rates

Appendix C. Test vectors for Status List encoding

All examples here are given in the form of JSON or CBOR payloads. The examples are encoded according to Section 4.2 for JSON and Section 4.3 for CBOR. The CBOR examples are displayed as hex values.

All values that are not mentioned for the examples below can be assumed to be 0 (VALID). All examples are initialized with a size of 2^20 entries.

C.1. 1-bit Status List

The following example uses a 1-bit Status List (2 possible values):

```
status[0] = 0b1
status[1993] = 0b1
status[25460] = 0b1
status[159495] = 0b1
status[495669] = 0b1
status[554353] = 0b1
status[645645] = 0b1
status[723232] = 0b1
status[854545] = 0b1
status[934534] = 0b1
status[1000345] = 0b1
```

JSON encoding:

```
{
  "bits": 1,
  "lst": "eNrt3AENwCAMAEGogklACtKQPg9LugC9k_ACvreiogE
  AAKkeCQAAAAAAAAAAAAAAAAAAAIBylgQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAXG9IAAAAAAAAPwsJAAAAAAAAAAAAAAvhsSAAAAAAAAAAAA
  A7KpLAAAAAAAAAAAAAAAAAAAAAJsLCQAAAAAAAAAAAADjelAAAAAAAAAAAKjDMAQAAA
  ACAZC8L2AEb"
}
```


CBOR encoding:


```
a2646269747301636c737458bd78daeddc010dc0200c0041a88249400ad2903e0f4b
ba00bd93f002beb7a2a2010000a91e09000000000000000000000000000000807296
04000000000000000000000000000000000000000000000000000000000000000000
0000000000005c6f480000000000000000fc2c240000000000000000000000be1b12
000000000000000000ecaa4b000000000000000000000000000000009b0b09000000
00000000000038de9400000000000000002a30cc010000000080642f0bd8011b
```


C.2. 2-bit Status List

The following example uses a 2-bit Status List (4 possible values):

```
status[0] = 0b01
status[1993] = 0b10
status[25460]= 0b01
status[159495] = 0b11
status[495669] = 0b01
status[554353] = 0b01
status[645645] = 0b10
status[723232] = 0b01
status[854545] = 0b01
status[934534] = 0b10
status[1000345] = 0b11
```

JSON encoding:


```
{
  "bits": 2,
  "lst": "eNrt2zENACEQAEEuoaBABP5VIO01fCjIHTMStt9ovGV
  IAAAAAABAbiEBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEB5WwIAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAID0ugQAAAAAAAAAAAAAAAAAQG12SgAAA
  AAAAAAAAAAAAAAAAAAAAAAAAOCSIQEAAAAAAAAAAAAAAAAAAAAAAAD8ExIAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwJEuAQAAAAAAAAAAAAAAAAAAAAAAAMB9S
  wIAAAAAAAAAAAAAAAAAAACoYUoAAAAAAAAAAAAAAEBqH81gAQw"
}
```


CBOR encoding:

```
a2646269747302636c737459013d78daeddb310d00211000412ea1a04004fe5520ed
357c28c81d3312b6df68bc6548000000000406e2101000000000000000000000000
0000000000000000000000000000000000000040795b020000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0080f4ba040000000000000000000000000406d764a000000000000000000000000
0000000000000000e092210100000000000000000000000000000000000fc1312
0000000000000000000000000000000000000000000000000000000000c0912e
0100000000000000000000000000000000000c07d4b020000000000000000000000
00000000a8614a0000000000000000000000406a1fcd60010c
```

C.3. 4-bit Status List

The following example uses a 4-bit Status List (16 possible values):

```
status[0] = 0b0001
status[1993] = 0b0010
status[35460] = 0b0011
status[459495] = 0b0100
status[595669] = 0b0101
status[754353] = 0b0110
status[845645] = 0b0111
status[923232] = 0b1000
status[924445] = 0b1001
status[934534] = 0b1010
status[1004534] = 0b1011
status[1000345] = 0b1100
status[1030203] = 0b1101
status[1030204] = 0b1110
status[1030205] = 0b1111
```

JSON encoding:

```
{
  "bits": 4,
  "lst": "eNrt0EENgDAQADAIHwImkIIEJEwCUpCEBBQRHOy35Li
  1EjoOQGabAgAAAAAAAAAAAAAAAAAACC1SQEAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABADrsCAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAADoxaEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIIoCgAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACArpwKAAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAGhqVkAzlwIAAAAiGVRAAAAAAAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAABx3AoAgLpVAQAAAAAAAAAAAAwM89rwMAAAAAAAAA
  AjsA9xMBMA"
}
```

CBOR encoding:

```
a2646269747304636c737459024878daedd0410d8030100030081f0226908204244c
025290840414111cecb7e4b8b5123a0e40669b020000000000000000000000000000
0020b549010000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000400ebb020000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
000000000000e8c5a100000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000082280a0000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000080ae9c0a
0000000000000000000000000000000000000000000000000000000000000000000000
000000686a56403397020000000886551000000000000000000000000000000000000
0000000000000000000000000000071dc0a0080ba55010000000000000000000000000
c0cf3daf0300000000000000008ec03dc4c04c0
```

C.4. 8-bit Status List

The following example uses a 8-bit Status List (256 possible values):

```
status[233478] = 0b00000000
status[52451] = 0b00000001
status[576778] = 0b00000010
status[513575] = 0b00000011
status[468106] = 0b00000100
status[292632] = 0b00000101
```

```
   status[214947] = 0b00000110
   status[182323] = 0b00000111
   status[884834] = 0b00001000
   status[66653] = 0b00001001
   status[62489] = 0b00001010
   status[196493] = 0b00001011
   status[458517] = 0b00001100
   status[487925] = 0b00001101
   status[55649] = 0b00001110
   status[416992] = 0b00001111
   status[879796] = 0b00010000
   status[462297] = 0b00010001
   status[942059] = 0b00010010
   status[583408] = 0b00010011
   status[13628] = 0b00010100
   status[334829] = 0b00010101
   status[886286] = 0b00010110
   status[713557] = 0b00010111
   status[582738] = 0b00011000
   status[326064] = 0b00011001
   status[451545] = 0b00011010
   status[705889] = 0b00011011
   status[214350] = 0b00011100
   status[194502] = 0b00011101
   status[796765] = 0b00011110
   status[202828] = 0b00011111
   status[752834] = 0b00100000
   status[721327] = 0b00100001
   status[554740] = 0b00100010
   status[91122] = 0b00100011
   status[963483] = 0b00100100
   status[261779] = 0b00100101
   status[793844] = 0b00100110
   status[165255] = 0b00100111
   status[614839] = 0b00101000
   status[758403] = 0b00101001
   status[403258] = 0b00101010
   status[145867] = 0b00101011
   status[96100] = 0b00101100
   status[477937] = 0b00101101
   status[606890] = 0b00101110
   status[167335] = 0b00101111
   status[488197] = 0b00110000
   status[211815] = 0b00110001
   status[797182] = 0b00110010
   status[582952] = 0b00110011
   status[950870] = 0b00110100
   status[765108] = 0b00110101
```

```
   status[341110] = 0b00110110
   status[776325] = 0b00110111
   status[745056] = 0b00111000
   status[439368] = 0b00111001
   status[559893] = 0b00111010
   status[149741] = 0b00111011
   status[358903] = 0b00111100
   status[513405] = 0b00111101
   status[342679] = 0b00111110
   status[969429] = 0b00111111
   status[795775] = 0b01000000
   status[566121] = 0b01000001
   status[460566] = 0b01000010
   status[680070] = 0b01000011
   status[117310] = 0b01000100
   status[480348] = 0b01000101
   status[67319] = 0b01000110
   status[661552] = 0b01000111
   status[841303] = 0b01001000
   status[561493] = 0b01001001
   status[138807] = 0b01001010
   status[442463] = 0b01001011
   status[659927] = 0b01001100
   status[445910] = 0b01001101
   status[1046963] = 0b01001110
   status[829700] = 0b01001111
   status[962282] = 0b01010000
   status[299623] = 0b01010001
   status[555493] = 0b01010010
   status[292826] = 0b01010011
   status[517215] = 0b01010100
   status[551009] = 0b01010101
   status[898490] = 0b01010110
   status[837603] = 0b01010111
   status[759161] = 0b01011000
   status[459948] = 0b01011001
   status[290102] = 0b01011010
   status[1034977] = 0b01011011
   status[190650] = 0b01011100
   status[98810] = 0b01011101
   status[229950] = 0b01011110
   status[320531] = 0b01011111
   status[335506] = 0b01100000
   status[885333] = 0b01100001
   status[133227] = 0b01100010
   status[806915] = 0b01100011
   status[800313] = 0b01100100
   status[981571] = 0b01100101
```

```
   status[527253] = 0b01100110
   status[24077] = 0b01100111
   status[240232] = 0b01101000
   status[559572] = 0b01101001
   status[713399] = 0b01101010
   status[233941] = 0b01101011
   status[615514] = 0b01101100
   status[911768] = 0b01101101
   status[331680] = 0b01101110
   status[951527] = 0b01101111
   status[6805] = 0b01110000
   status[552366] = 0b01110001
   status[374660] = 0b01110010
   status[223159] = 0b01110011
   status[625884] = 0b01110100
   status[417146] = 0b01110101
   status[320527] = 0b01110110
   status[784154] = 0b01110111
   status[338792] = 0b01111000
   status[1199] = 0b01111001
   status[679804] = 0b01111010
   status[1024680] = 0b01111011
   status[40845] = 0b01111100
   status[234603] = 0b01111101
   status[761225] = 0b01111110
   status[644903] = 0b01111111
   status[502167] = 0b10000000
   status[121477] = 0b10000001
   status[505144] = 0b10000010
   status[165165] = 0b10000011
   status[179628] = 0b10000100
   status[1019195] = 0b10000101
   status[145149] = 0b10000110
   status[263738] = 0b10000111
   status[269256] = 0b10001000
   status[996739] = 0b10001001
   status[346296] = 0b10001010
   status[555864] = 0b10001011
   status[887384] = 0b10001100
   status[444173] = 0b10001101
   status[421844] = 0b10001110
   status[653716] = 0b10001111
   status[836747] = 0b10010000
   status[783119] = 0b10010001
   status[918762] = 0b10010010
   status[946835] = 0b10010011
   status[253764] = 0b10010100
   status[519895] = 0b10010101
```

```
   status[471224] = 0b10010110
   status[134272] = 0b10010111
   status[709016] = 0b10011000
   status[44112] = 0b10011001
   status[482585] = 0b10011010
   status[461829] = 0b10011011
   status[15080] = 0b10011100
   status[148883] = 0b10011101
   status[123467] = 0b10011110
   status[480125] = 0b10011111
   status[141348] = 0b10100000
   status[65877] = 0b10100001
   status[692958] = 0b10100010
   status[148598] = 0b10100011
   status[499131] = 0b10100100
   status[584009] = 0b10100101
   status[1017987] = 0b10100110
   status[449287] = 0b10100111
   status[277478] = 0b10101000
   status[991262] = 0b10101001
   status[509602] = 0b10101010
   status[991896] = 0b10101011
   status[853666] = 0b10101100
   status[399318] = 0b10101101
   status[197815] = 0b10101110
   status[203278] = 0b10101111
   status[903979] = 0b10110000
   status[743015] = 0b10110001
   status[888308] = 0b10110010
   status[862143] = 0b10110011
   status[979421] = 0b10110100
   status[113605] = 0b10110101
   status[206397] = 0b10110110
   status[127113] = 0b10110111
   status[844358] = 0b10111000
   status[711569] = 0b10111001
   status[229153] = 0b10111010
   status[521470] = 0b10111011
   status[401793] = 0b10111100
   status[398896] = 0b10111101
   status[940810] = 0b10111110
   status[293983] = 0b10111111
   status[884749] = 0b11000000
   status[384802] = 0b11000001
   status[584151] = 0b11000010
   status[970201] = 0b11000011
   status[523882] = 0b11000100
   status[158093] = 0b11000101
```

```
   status[929312] = 0b11000110
   status[205329] = 0b11000111
   status[106091] = 0b11001000
   status[30949] = 0b11001001
   status[195586] = 0b11001010
   status[495723] = 0b11001011
   status[348779] = 0b11001100
   status[852312] = 0b11001101
   status[1018463] = 0b11001110
   status[1009481] = 0b11001111
   status[448260] = 0b11010000
   status[841042] = 0b11010001
   status[122967] = 0b11010010
   status[345269] = 0b11010011
   status[794764] = 0b11010100
   status[4520] = 0b11010101
   status[818773] = 0b11010110
   status[556171] = 0b11010111
   status[954221] = 0b11011000
   status[598210] = 0b11011001
   status[887110] = 0b11011010
   status[1020623] = 0b11011011
   status[324632] = 0b11011100
   status[398244] = 0b11011101
   status[622241] = 0b11011110
   status[456551] = 0b11011111
   status[122648] = 0b11100000
   status[127837] = 0b11100001
   status[657676] = 0b11100010
   status[119884] = 0b11100011
   status[105156] = 0b11100100
   status[999897] = 0b11100101
   status[330160] = 0b11100110
   status[119285] = 0b11100111
   status[168005] = 0b11101000
   status[389703] = 0b11101001
   status[143699] = 0b11101010
   status[142524] = 0b11101011
   status[493258] = 0b11101100
   status[846778] = 0b11101101
   status[251420] = 0b11101110
   status[516351] = 0b11101111
   status[83344] = 0b11110000
   status[171931] = 0b11110001
   status[879178] = 0b11110010
   status[663475] = 0b11110011
   status[546865] = 0b11110100
   status[428362] = 0b11110101
```

```
   status[658891] = 0b11110110
   status[500560] = 0b11110111
   status[557034] = 0b11111000
   status[830023] = 0b11111001
   status[274471] = 0b11111010
   status[629139] = 0b11111011
   status[958869] = 0b11111100
   status[663071] = 0b11111101
   status[152133] = 0b11111110
   status[19535] = 0b11111111
```

JSON encoding:

```
{
  "bits": 8,
  "lst": "eNrt0WOQM2kYhtGsbdu2bdu2bdu2bdu2bdu2jVnU1my
  -SWYm6U5enFPVf7ue97orFYAo7CQBAACQuuckAABStqUEAAAAAAAAtN6wEgAE71QJA
  AAAAIrwhwQAAAAAAdtAAgAAAAAAACLwkAQAAAAAAAAAAACUaFcJAACAeJwkAQAAAAA
  AAABQvL4kAAAAWmJwCQAAAAAAAAjAwBIAAAB06ywJoDKQBARpfgkAAAAAAAAAAAAA
  AAAAACo50sJAAAAAAAAAOiRcSQAAAAAgAJNKgEAAG23mgQAAAAAAECw3pUAQvegBAA
  AAAAAAADduE4CAAAAyjSvBAAQiw8koHjvSABAb-wlARCONyVoxtMSZOd0CQAAAOjWD
  RKQmLckAAAAAACysLYEQGcnSAAAAAAQooUlAABI15kSAIH5RAIgLB9LABC4_SUgGZN
  IAABAmM6RoLbTJIASzCIBAEAhfpcAAAAAAABquk8CAAAAAAAAaJl9SvvzBOICAFWmk
  IBgfSgBAAAANOgrCQAAAAAAADStK8EAAC03gASAAAAAAAAAADFWFUCAAAAMjOaBEA
  DHpYAQjCIBADduFwCAAAAGitMSSI3BUSAECOHpAA6IHrJQAAAAAAsjeVBAAAKRpVA
  orWvwQAAAAAAAAkKRtJAAAAAAAgCbcLAF0bXUJAAAAoF02kYDg7CYBAAAAAEB6NpQ
  AAAAAAAAAAAAAAEr1uQQAAF06VgIAAAAAAAAAqDaeBAAQqgMkAAAAAABogQMlAAAAA
  AAa87MEAAAQiwslAAAAAAAAAAAAAAAAMrOyBAAAiekv-hcsY0Sgne6QAAAAAAAgaUt
  JAAAAAAAAAAAAAAAAAAAAAAAAAADwt-07vjVkAAAAgDy8KgFAUEaSAAAAAJL3vgQAW
  dhcAgAAoBHDSUDo1pQAAACI2o4SAABZm14CALoyuwQAAPznGQkgZwdLAAAQukclAAA
  AAAAAAAAAgKbMKgEAAAAAAAAAAAAAAAAAAECftpYAAAAAAAAAAAAACnaXBAAAAADk7
  iMJAAAAAAAAAABqe00CAnGbBBG4TAIAgFDdKgFAXCaWAAAAAAAAAAAAAAAAAKAJQwR
  72XbGAQAAAKAhh0sAAAAAAABQgO8kAAAAAAAAAAAAACAaM0kAAAC5W0QCAIJ3mAQAx
  GwxCQAA6nhSAsjZBRIAANEbWQIAAAAAaJE3JACAwA0qAUBIVpKAlphbAiAPp0iQnKE
  kAAAAAAAgBP1KAAAAdOl4CQAAAAAAAPjLZBIAAG10RtrPm8_CAEBMTpYAAAAAAIjQY
  BL8z5QSAAAAAEDYPpUAACAsj0gAAADQkHMlAAjHDxIA0Lg9JQAAgHDsLQEAAABAQS6
  WAAAAgLjNFs2l_RgLAIAEfCEBlGZZCQAAaIHjJACgtlskAAAozb0SAAAAVFtfAgAAA
  AAAAAAAAAAAAAAAAAAAAKDDtxIAAAAAVZaTAKB5W0kAANCAsSUgJ0tL0GqHSNBbL0g
  AZflRAgCARG0kQXNmlgCABiwkAQAAAEB25pIAAAAAAAAAAAAAoFh9SwAAAAAADWNm
  OSrpjFsEoaRgDKcF9Q1dxsEAAAAAAAAAAAAAAAAgPZ6SQIAAAAAAAAAgChMLgEAAAA
  AAAAAqZlQAsK2qQQAAAAAAAD06XUJAAAAqG9bCQAAgLD9IgEAAAAAAAAAAAAAAAAAA
  EBNe0gAAAAAAAAAAEBPHSEBAAAAlOZtCYA4fS8B0GFRCQAo0gISAOTgNwmC840EAAA
  AAAAAAAAAAAAAAAAUJydJfjXPBIAAAAAAAAAAAAAAABk6WwJAAAAAAAAAAAAAAAAq
  G8UCQAAgPpOlAAAIA83SQAANWwc9HUjGAgAAAAAACAusaSAAAAAAAAAAAAAAAAAAA
  AAAAAAAAAqHKVBACQjxklAAAAAAAAAKBHxpQAAAAACBME0lAdlaUAACyt7sEAAAA0
  Nl0EgAAAAAAAAAAABA-8wgAQAAAAAAKU4SgKgUtlBAgAAAAAAAAAgMCMLwEE51k
  JICdzSgCJGl2CsE0tAQAA0L11JQAAAAAAAjUOhIAAAAAAAAAAAAAAGTqeQkAAAAAA
  AAAAAAAKM8SEjTrJwkAAAAAACocqQEULgVJAAAACjDUxJUKgtKAAAAqbpRAgCA0n0
  mAQAAAABAGzwmAUCTLpUAAAAAAAAAEjZNRIAAAAAAAAAAAAAAAAAAAAA8I-vJaAlh
  pQAAAAAHrvzjJ-OqCuuVlLAojP8BJAr70sQZVDJYAgXS0BAAAAAAAAAAAAtMnyEgA
  AAAAAFONKCQAAAAAAADorc0kAAAAAAAAgDqOlgAAAAAAAAAAAADIwv0SAAAAAAAAA
  AAAAADBuV0CIFVDSwAAAABAAI6RAAAAAGIwrQSEZAsJAABouRclAAAAAKDDrxIAAAA
  0bkkJgFiMKwEAAAAAAHQyhwRk7h4JAAAAAAAAAAAAgatdKAACUYj0JAAAAAAAAAAAAQ
  nORBLTFJRIAAAAAkIaDJAAAAJryngQAAAAAAAAAAAA98oQEAAAAAAAAAEC2zpcgWY9
  LQKL2kwAgGK9IAAAAAPHaRQIAAAAAAAAAAADIxyoSAAAAAAAAAAAAAAADQFotLAECz_
  gQ1PX-B"
}
```

CBOR encoding:

```
a2646269747308636c73745907b078daedd1639033691886d1ac6ddbb66ddbb66ddb
b66ddbb66ddbb68d59d4d66cbe496626e94e5e9c53d57fbb9ef7ba2b158028ec2401
000090bae724000052b6a504000000000000b4deb0120004ef5409000000008af087
040000000001db40020000000000022f0900400000000000000000946857090000
80789c2401000000000000050bcbe240000005a62700900000000000008c0c01200
000074eb2c09a032900404697e0900000000000000000000000000000a8e74b0900
000000000000e89171240000000080024d2a0100006db79a04000000000040b0de95
0042f7a00400000000000000ddb84e02000000ca34af0400108b0f24a078ef480040
6fec2501108e372568c6d31264e77409000000e8d60d129098b7240000000000b2b0
b6044067274800000000010a28525000048d799120081f94402202c1f4b0010b8fd25
2019934800004098ce91a0b6d3248012cc22010040217e970000000000006aba4f02
0000000000068997d4afbf304e2020055a69080607d280100000034e82b09000000
00000000d2b4af040000b4de0012000000000000000c558550200000032339a0440
031e96004230880400ddb85c020000000068ad312488dc151200408e1e9000e881eb
250000000000b23795040000291a55028ad6bf040000000000000090a46d24000000
00008026dc2c01746d7509000000a05d369180e0ec260100000000407a3694000000
0000000000000004af5b90400005d3a56020000000000000000a8369e040010aa0324
00000000006881032500000000001af3b3040000108b0b250000000000000000000000
000032b3b204000089e92ffa172c6344a09dee900000000000020694b490000000000
00000000000000000000000000000000f0b7ed3bbe3564000000803cbc2a0140504692
0000000092f7be040059d85c020000a011c34940e8d69400000088da8e120000599b
5e0200ba32bb040000fce719092067074b000010ba472500000000000000000080a6
cc2a0100000000000000000000000000000409fb696000000000000000000000a7697
0400000000e4ee2309000000000000000006a7b4d0202719b0411b84c02008050dd2a
01405c2696000000000000000000000000000a00943047bd976c601000000a021874b
0000000000005080ef2400000000000000000000201a3349000000b95b4402008277
980400c46c31090000ea785202c8d905120000d11b5902000000006891372400080c0
0d2a01404856928096985b02200fa748909ca12400000000002004fd4a00000074e9
78090000000000000f8cb641200006d7446dacf9bcfc200404c4e96000000000088d0
6012fccf94120000000040d83e950000202c8f48000000d09073250008c70f1200d0
b83d2500008070ec2d0100000040412e9600000080b8cd16cda5fd180b0080047c21
019466590900006881e32400a0b65b24000028cdbd12000000545b5f020000000000
0000000000000000000000000000a0c3b71200000000055969300a0795b490000d080b1
2520274b4bd06a8748d05b2f480065f951020080446d24417366960080062c240100
00004076e69200000000000000000000a0587d4b000000000000358d98e4aba6316c
12869180329c17d435771b040000000000000000000000000080f67a49020000000000
000080284c2e0100000000000000a9995002c2b6a9040000000000000f4e975090000
00a86f5b09000080b0fd22010000000000000000000000000000404d7b4800000000
00000000404f1d210100000094e66d0980387d2f01d06151090028d2021200e4e037
0982f38d040000000000000000000000000000000509c9d25f8d73c12000000000000
00000000000064e96c0900000000000000000000000a86f1409000080fa4e940000
200f37490000356c1cf4752318080000000000000080bac69200000000000000000000
0000000000000000000000a872950400908f19250000000000000000a047c694000000
0000204c1349407656940000b2b7bb04000000d0d97412000000000000000000000040
fbcc200100000000000000a5384a02a052d941020000000000000000080c08c2f0104e7
59092027734a00891a5d82b04d2d010000d0bd752500000000000000008d43a12000000
0000000000000000064ea790900000000000000000000028cf121234eb270900000000
0000a872a40450b8152400000028c35312542a0b4a000000a9ba51020080d27d2601
```

```
00000000401b3c260140932e9500000000000000000048d935120000000000000000000
00000000000000f08faf25a02586940000000000007aefce327e3aa0aeb9594b0288cf
f01240afbd2c4195432580205d2d0100000000000000000000b4c9f212000000000014
e34a0900000000000000e8adcd24000000000000803a8e960000000000000000000000
c8c2fd120000000000000000000000c1b95d022055434b0000000040008e91000000
006230ad0484640b09000068b9172500000000a0c3af12000000346e490980588c2b
0100000000007432870464ee1e0900000000000000000206ad74a000094623d090000
0000000000000042739104b4c525120000000090868324000000 9af29e0400000000
00000000003df2840400000000000000040b6ce9720598f4b40a2f693002018af4800
000000f1da450200000000000000000c8c72a120000000000000000000000d0168b
4b0040b3fe04353d7f81
```

Document History

[[To be removed from the final specification]]

-15

* limit Status List Token CWT COSE message to Sign1/Mac0

* be explicit about tagging and re-add cose_sign1 tag to example

* add description field to EKU iana registration request

* fix typos in referenced token

* fix typos

* make IANA references informative

* remove unused iana.jose reference

-14

* use binary value encoding for all test vectors (display purposes only)

* removed bytes from graphic that were intepreted as padding bytes

* removed 0x0B from application-specific Status Type

* reemphasized that expired tokens with status "VALID" are still expired

* renamed section "Status List Aggregation in JSON Format" to "Status List Aggregation Data Structure"

* slightly restructure/clarify referenced token cose section

* Add ASN.1 module
* many nits and improvements from genart review
* remove cose_sign1 tag from statuslist in cwt form examples
* slightly restructure/clarify referenced token cose section
* Add ASN.1 module
* removed DL suspension example

-13

* add definition of client to terminology
* Make exp and ttl recommended in claim description (fixes inconsistency, was recommended in other text)
* Add short security consideraiton on redirects and ttl
* fix CORS spec to specific version
* explain KYC
* link implementation guidance to exp and ttl in Status List Token definition
* reference RFC7515 instead of IANA:JOSE
* add a note that cwt is encoded in raw/binary.
* added further privacy consideration around issuer tracking using unique URIs

-12

* Allow for extended key usage OID to be used for other status mechanisms
* add Paul’s affiliation
* add feedback from Dan Moore
* change JSON Status List structure to only contain JSON object
* further nitpicks

* clarifying status and status_list IANA descriptions for JWT/CWT
* clarifying description texts for status and status_list in CBOR
* splitting Linkability Mitigation from Token Lifecycle section in Implementation Consideration
* relax the accept header from must to should

-11

* incorporate feedback from shepherd review
* some nitpicks
* even more nitpicks

-10

* improve caching guidelines and move them to implementaiton considerations
* Add CoAP Content-Format ID and IANA registration
* Add size comparison for status list and compressed uuids
* Change Controller IESG for OAuths Parameters Registration

-09

* update acknowledgments
* introduce dedicated section for compressed byte array of the Status List
* fix Status List definitions
* Add CDDL for CBOR StatusList encoding
* add diagram for Status List Aggregation for further explanation
* rename "chunking" of Status List Tokens (for scalability reasons) into "divide .. up"

-08

* Fix cwt typ value to full media type

* Holders may also fetch and verify Status List Tokens
* Update terminology for referenced token and Status List Token

-07

* add considerations about External Status Issuer or Status Provider
* add recommendations for Key Resolution and Trust Management
* add extended key usage extensions for x509
* Relying Parties avoiding correlatable Information
* editorial changes on terminology and Referenced Tokens
* clarify privacy consideration around one time use referenced tokens
* explain the Status List Token size dependencies
* explain possibility to chunk Status List Tokens depending on Referenced Token’s expiry date
* add short-lived tokens in the Rationale
* rename Status Mechanism Methods registry to Status Mechanisms registry
* changes as requested by IANA review
* emphasize that security and privacy considerations only apply to Status List and no other status mechanisms
* differentiate unlinkability between Issuer-RP and RP-RP
* add more test vectors for the status list encoding
* add prior art
* updated language around application specific status type values and assigned ranges for application specific usage
* add short security considerations section for mac based deployments
* privacy considerations for other status types like suspended

* fix aggregation_uri text in referenced token
* mention key resolution in validation rules

-06

* iana registration text updated with update procedures
* explicitly mention that status list is expected to be contained in cryptographically secured containers
* reworked and simplified introduction and abstract
* specify http status codes and allow redirects
* add status_list_aggregation_endpoint OAuth metadata
* remove unsigned options (json/cbor) of status list
* add section about mixing status list formats and media type
* fixes from IETF review
* update guidance around ttl
* add guidance around aggregation endpoint

-05

* add optional support for historical requests
* update CBOR claim definitions
* improve section on Status Types and introduce IANA registry for it
* add Status Issuer and Status Provider role description to the introduction/terminology
* add information on third party hosting to security consideration
* remove constraint that Status List Token must not use a MAC

-04

* add mDL example as Referenced Token and consolidate CWT and CBOR sections

* add implementation consideration for Default Values, Double Allocation and Status List Size
* add privacy consideration on using private relay protocols
* add privacy consideration on observability of outsiders
* add security considerations on correct parsing and decoding
* remove requirement for matching iss claim in Referenced Token and Status List Token
* add sd-jwt-vc example
* fix CWT status_list map encoding
* editorial fixes
* add CORS considerations to the http endpoint
* fix reference of Status List in CBOR format
* added status_list CWT claim key assigned
* move base64url definition to terminology

-03

* remove unused reference to RFC9111
* add validation rules for status list token
* introduce the status list aggregation mechanism
* relax requirements for status_list claims to contain other parameters
* change cwt referenced token example to hex and annotated hex
* require TLS only for fetching Status List, not for Status List Token
* remove the undefined phrase Status List endpoint
* remove http caching in favor of the new ttl claim
* clarify the sub claim of Status List Token

* relax status_list iss requirements for CWT

* Fixes missing parts & iana ttl registration in CWT examples

-02

* add ttl claim to Status List Token to convey caching

* relax requirements on referenced token

* clarify Deflate / zlib compression

* make a reference to the Issuer-Holder-Verifier model of SD-JWT VC

* add COSE/CWT/CBOR encoding

-01

* Rename title of the draft

* add design consideration to the introduction

* Change status claim to in referenced token to allow re-use for other mechanisms

* Add IANA Registry for status mechanisms

* restructure the sections of this document

* add option to return an unsigned Status List

* Changing compression from gzip to zlib

* Change typo in Status List Token sub claim description

* Add access token as an example use-case

-00

* Initial draft after working group adoption

* update acknowledgments

* renamed Verifier to Relying Party

* added IANA consideration

[draft-ietf-oauth-status-list]

-01

* Applied editorial improvements suggested by Michael Jones.

-00

* Initial draft

Authors' Addresses

Tobias Looker
MATTR
Email: tobias.looker@mattr.global

Paul Bastian
Bundesdruckerei
Email: paul.bastian@posteo.de

Christian Bormann
SPRIND
Email: chris.bormann@gmx.de