Token Status List

A simple and scalable credential revocation/status mechanism
[Formerly known as JWT CWT Status List]

Tobias Looker, Paul Bastian, Christian Bormann
A Refresher - The Problem

How to enable the issuer of a token (e.g. CWT or JWT) to communicate dynamic status information about a token after it is issued and before it expires.

Example - An SD-JWT Verifiable Credential where the Issuer would like to communicate whether the credential is revoked or not.
Big picture

1. transmit reference token w/ status info:
   - idx
   - uri

2. GET HTTPS uri

3. response w/ status list

4. verify status list and entry at idx

Issuer ➔ Status Provider ➔ Verifier ➔ Wallet ➔ update status
Changes: -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
Changes: -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 - Change status claim to “cnf” style registry

- We expect different status mechanisms to be used based on use-cases and requirement
- Instead of “status” defining the status list reference, we introduced a mechanism similar to the confirmation claim
- Ability to re-use the general mechanism and add other status mechanisms

```json
{
    "iss": "https://example.com",
    "status": {
        "idx": 0,
        "uri": "https://example.com/statuslists/1"
    }
}

{
    "iss": "https://example.com",
    "status": {
        "status_list": {
            "idx": 0,
            "uri": "https://example.com/statuslists/1"
        }
    }
}
```
01 - Option to return unsigned status list

- Support for a simpler option that places trust in TLS, rather than application level signatures.
- Akin in some ways to the unsigned authorisation metadata option in OAuth2.
- Enable status list repudiation.
In order to convey the intended update interval of a status list, we have defined a new JWT and CWT claim for expressing the *time to live* for a token. When applied to the status list this enables a way for a consumer of a status list to know when to check for updates. The main alternative considered was an absolute timestamp for update checking which was dismissed because:

1. It synchronizes refresh/update requests from consumers.
2. It enables communicating a reoccurring update/refresh interval instead of a single point-of-time without having to resign the status list when there are no updates.
02 - Add COSE/CWT/CBOR encoding

- We have added text and examples for:
  - Status List in CBOR Format
  - Status List Token in CWT Format
  - Referenced Token in CWT Format
  - Referenced Token in other COSE/CBOR Format
02 - Add COSE/CWT/CBOR encoding

- We have added text and examples for:
  - Status List in CBOR Format
  - Status List Token in CWT Format
  - Referenced Token in CWT Format
  - Referenced Token in other COSE/CBOR Format

```json
18(
    [
    / protected / << {
      / alg / 1: -7 / ES256 /
    } >>,
    / unprotected / {
      / kid / 4: h'3132' / '13' /
    },
    / payload / << {
      / iss / 1: "https://example.com",
      / status / 65535: {
        "status_list": {
          "idx": "0",
          "url": "https://example.com/statuslists/1"
        }
      }
    } >>,
    / signature / h'...'
] )
```
Discussion: Provide all StatusLists

- Provide a URL/List/Directory with URLs to all StatusList
  - This enables Verifiers to fetch all relevant StatusLists, e.g. in the morning for potential offline verification throughout the day
- Options
  - Define status list path structure like `<some-host>/<some-path>/<status-list>/<id>` where the parent path contains a list of `<ids>`
  - `.well-known/` path
  - additional URL inside the `status_list` claim
  - URL provided by the Issuer metadata
- Data Structure to be defined
Discussion: Identifier List?

- As we introduce a registry for status mechanisms in JSON/CBOR, two candidates exist:
  - Status-List (this spec)
  - Status-Attestation (Giuseppe’s proposal - OCSP-stapling like)
- We also see increased demand for a simple CRL-like mechanism
  - ISO mdoc folks are interested in this (--> time sensitive)
  - Putting this into the StatusList specification may overload and confuse people
  - Is there consensus for adding another, separate IETF specification for this?
- Call for Feedback:
  - Option 1: separate Draft
  - Option 2: add to Status List
Discussion: status IANA Registration?

- We are currently planning to create a registry for status mechanisms similar to how the confirmation claim (“cnf”) works.
- This should apply for both the JOSE and COSE world (and the mechanisms could be used by mdoc).
- Do we create a registry each in the JWT and CWT IANA documents?
Further Open Topics

- Validation Rules to be more detailed
- Security considerations to be more detailed
- Add more and fix some examples
- Status Object as header?
- Drop authorization considerations for Status List?
- Comparison of status mechanisms -> informational RFC Draft?
Questions?
Example: Referenced Token

```json
{
  "alg": "ES256",
  "kid": "11"
}

{
  "iss": "https://example.com",
  //other claims
  "status": {
    "status_list": {
      "uri": "https://example.com/statuslists/1",
      "idx": 5
    }
  }
}

Extension point for other status mechanisms
URI of the status list token
Index in the status list
Example: Status List in JWT

```json
{
  "alg": "ES256",
  "kid": "12",
  "typ": "statuslist+jwt"
}
```

```json
{
  "exp": 1687517770,
  "iat": 1686912970,
  "iss": "https://example.com",
  // other claims
  "status_list": {
    "bits": 1,
    "lst": "H4sIAMo_jGQC_zvp8hMAZLRLMQMAAAA"
  },
  "sub": "https://example.com/statuslists/1"
}
```
Example: How it fits together

```
"status_list": {
  "status_list": {
    "idx": 5,
    "uri": "https://example.com/statuslists/1",
  }
}

"sub": "https://example.com/statuslists/1"
"status_list": {
  "bits": 1,
  "lst": "H4sIAMo_jGQC_zvp8hMAZLRLMQMAAAA"
}
```

0x0 = VALID
0x1 = INVALID

1 0 0 1 0 1 0 0 0 1 0 0

Deflate zlib
Links

Datatracker -> [https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/](https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/)

Git Repository -> [https://github.com/oauth-wg/draft-ietf-oauth-status-list](https://github.com/oauth-wg/draft-ietf-oauth-status-list)