# Token Status List

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

- Refresher
- Updates since IETF 120
- Discussion points
- Q&A

*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.*

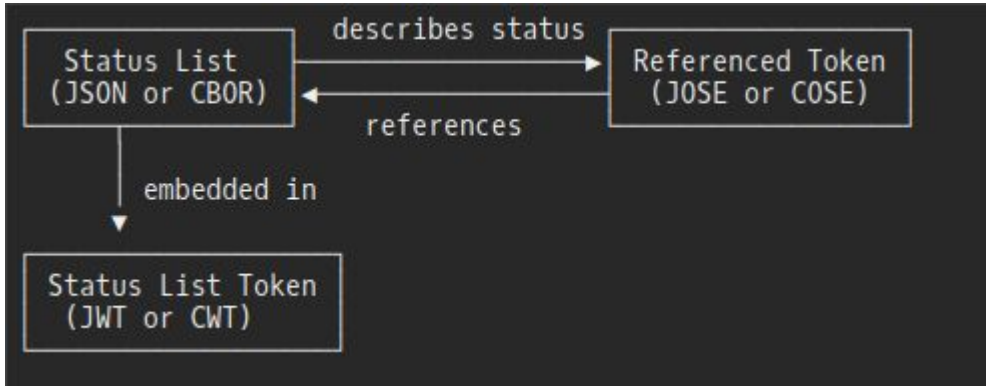
# Key Facts

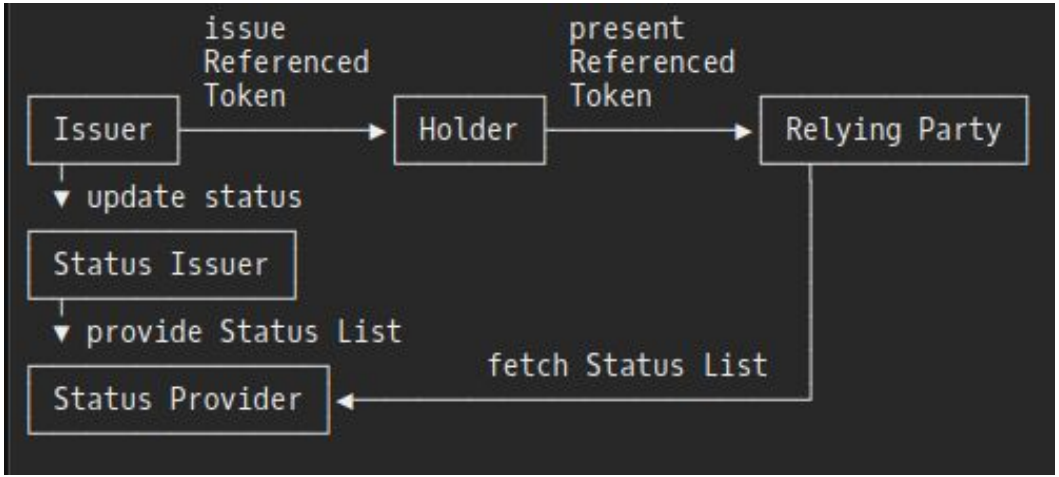
The most important facts:

- Scalable Revocation Mechanism
- Similar privacy properties as CRL
- Simple to understand, easy to implement
- Referenced by SD-JWT VC and ISO 18013-5 mdoc Amendment
- Named by ARF 1.4
- Tested at various interop events and hackathons
- Establishes an extension point and IANA registry for other status mechanisms

# Big Picture

- Architecture of Status List

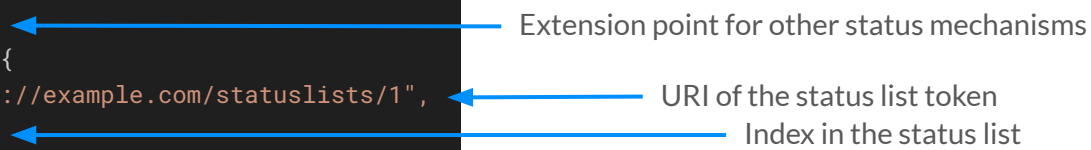



- Architecture of Status List
- Referenced Token = Credential
  - SD-JWT VC
  - ISO mdoc
  - Any other

# Example: Referenced Token

```
{
  "alg": "ES256",
  "kid": "11"
}
.
{
  "iss": "https://example.com",
  … //other claims
  "status": {
    "status_list": {
      "uri": "https://example.com/statuslists/1",
      "idx": 5
    }
  }
}
```

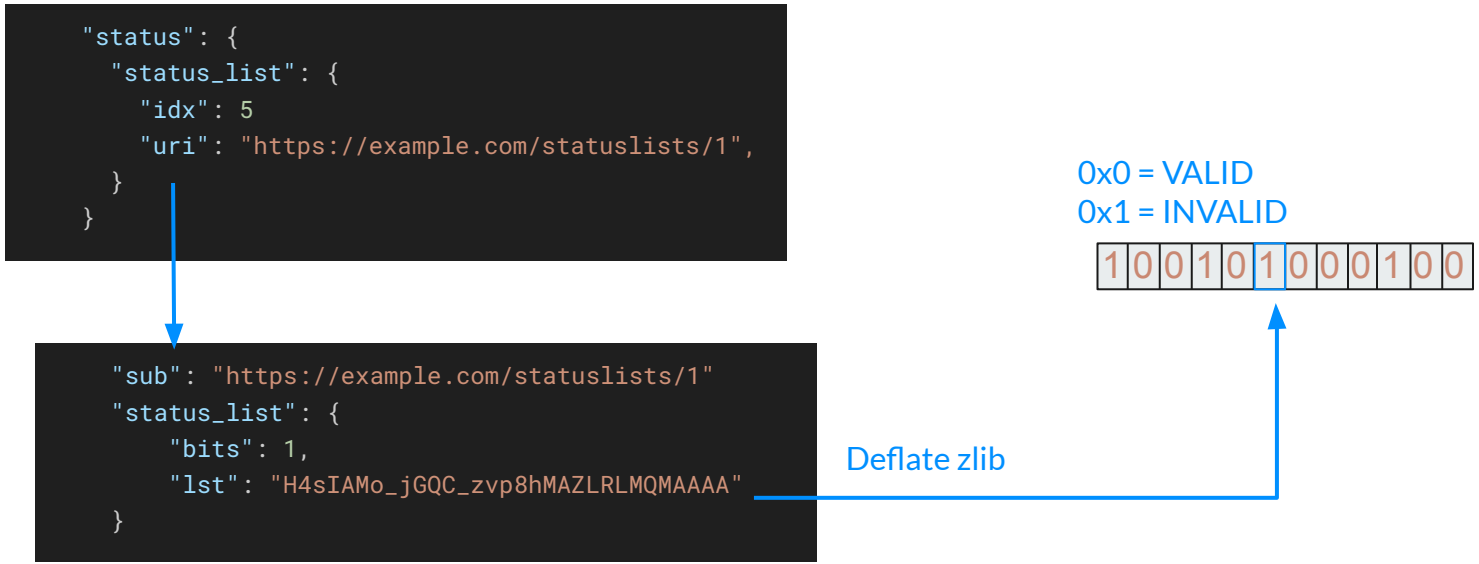


# Example: Status List in JWT

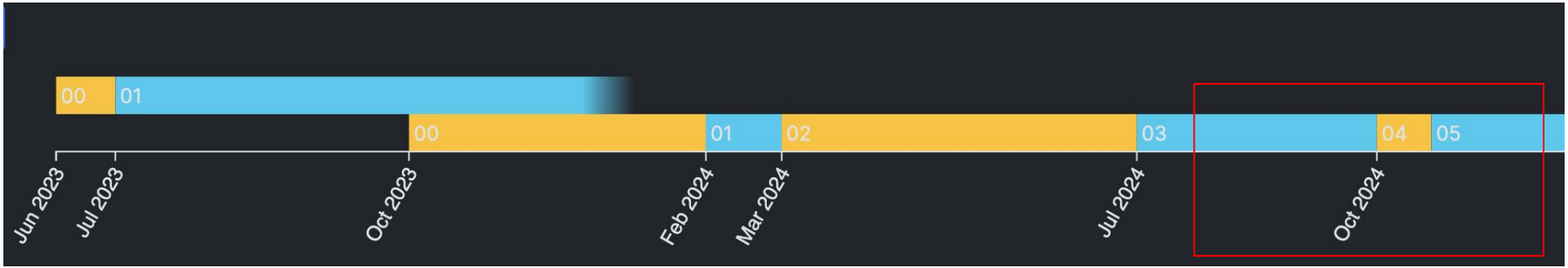
```
eyJhbGciOiJFUzI1NiIsImtpZCI6IjEyIiwidHlwIjoic3RhdHVzbGlzdCtqd3QifQ.eyJleHAiOjE2ODc1MTc3NzAsImlhdCI6MTY4NjkxMjk3MCwiaXNzIjoiaHR0cHM6Ly9leGFtcGxlLmNvbSIsInN0YXR1c19saXN0Ijp7ImJpdHMiOjIsImxzdCI6Ikg0c0lBTW9fakdRQ196dnA4aE1BWkxSTE1RTUFBQUEifSwic3ViIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9zdGF0dXNsaXN0cy8xIn0.8uaUXshaJdGWGjvwPwaa2Gtt0M7-M7dGO9rXaz3x99LCdG5tKb-ARL1ezqguLTs63VeudYWqpdg4HpN-D2hOkg
```

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



# Changes since IETF 120



# Changes: -04

- 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
- add mDL example as Referenced Token and consolidate CWT and CBOR sections
- add sd-jwt-vc example
- fix reference of Status List in CBOR format
- fix CWT status_list map encoding
- add CORS considerations to the http endpoint
- remove requirement for matching iss claim in Referenced Token and Status List Token
- added status_list CWT claim key assigned
- move base64url definition to terminology

# Changes: -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

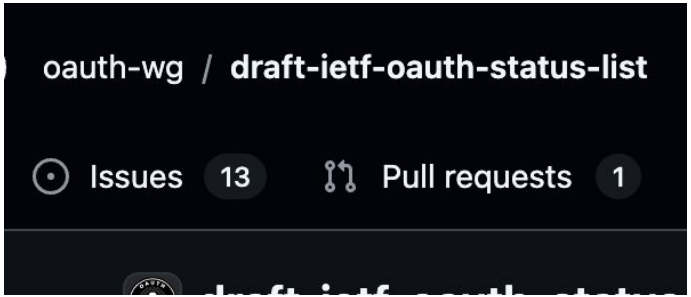
# Add optional support for historical requests

- There were requests to support use-cases that require historical status information
  - “This credential was valid at an earlier point in time”
- We were a bit reluctant to enable this directly in this spec for privacy reasons
- We Decided to add support an optional feature via a query parameter
  - With strong warnings to only use this feature if you are aware of the privacy concerns/risks
  - Better to have a common interface how to query this, than have diverging implementations for the use-cases that need it

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

# Remaining Issues

- We got pretty far in the past few months!




**Open Questions:**

- drop the unsigned option?
- allow other compression algorithms?
- Enforce HTTP Response Content-Types?
- otherwise mainly editorial issues

# Discuss: Unsigned Option

Current situation

- The current way the unsigned option is defined creates complexity (additional definitions, extra handling for verification etc.)
- All implementations we are aware of are using the signed variant
  - But: There were some voices to keep the unsigned option
- There may be use cases that are sufficient to rely on WebPKI/TLS and additional signing may be unnecessary
- Proposal: Drop unsigned option

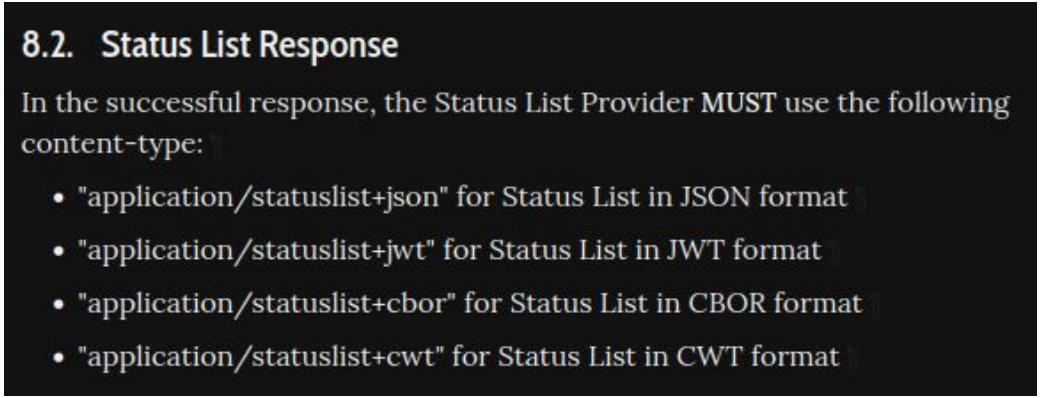
# Allow other compression algorithms?

- right now we always require compression using RFC1951 ZLIB

> 3. The byte array is compressed using DEFLATE [RFC1951] with the ZLIB [RFC1950] data format. Implementations are **RECOMMENDED** to use the highest compression level available.

- does it make sense to allow for other mechanisms?
  - add a registry for allowed algorithms?
  - allow no compression?
- some people argue that compression is already done on HTTP transport level → #156
- could be useful for potential ZK-Circuits that would have a hard time to do decompression
  - however, requirements and solutions for ZKP may be pre-mature?
- potentially create a registry with “zlib” as the only option

# Enforce Content-Type?



### 8.2. Status List Response

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

- "application/statuslist+json" for Status List in JSON format
- "application/statuslist+jwt" for Status List in JWT format
- "application/statuslist+cbor" for Status List in CBOR format
- "application/statuslist+cwt" for Status List in CWT format

- Issuer #168
- we enforce HTTP Content-Type
  - this allows the Relying Party to easily parse the Status List (Token)
- however, some CDNs don't support Content-Type
- Option 1
  - leave things as-is
  - CDNs that don't support this may violate the specification
- Option 2
  - relax MUST -> SHOULD, Status List format may be ecosystem/token specific out-of-band
  - however, then Relying Parties cannot rely on the Content-Type
  - this may make implementations more complicated

# Roadmap & WGLC

- after solving
  - the three issues
  - an editorial pass
- we want to start WGLC before the end of the year
- OAuth Interims Call?
- how to proceed?