SCITT – COSE

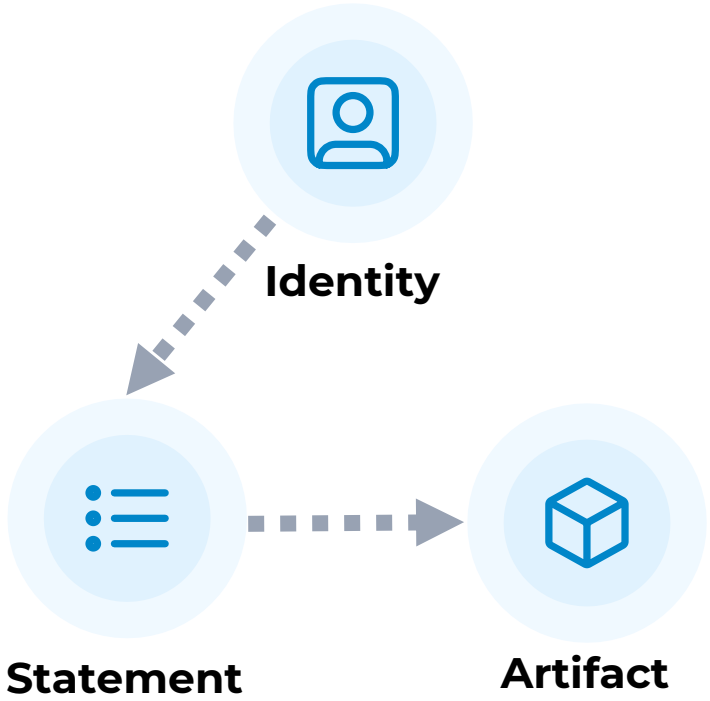
Managing SCITT Statements as COSE Payloads

Attached | *Detached* | *Hashed*

Steve Lasker
SteveLasker.blog
Director of Ecosystem
DataTrails
DataTrails.ai

Reference: https://or13.github.io/draft-steele-cose-hash-envelope/draft-steele-cose-hash-envelope.html

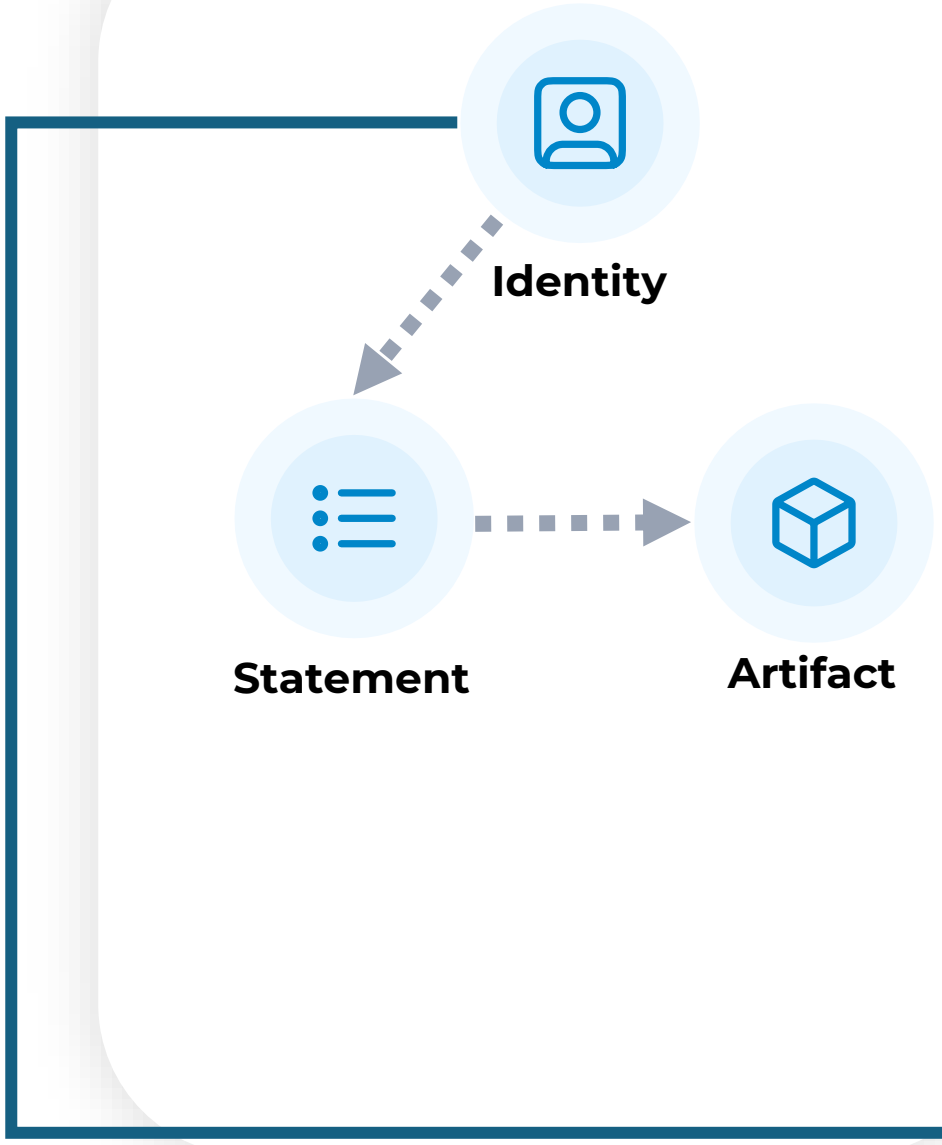
IETF SCITT



SCITT records

Who made
an immutable Statement
about an Artifact
recorded “when”

IETF SCITT



Who are the Who's

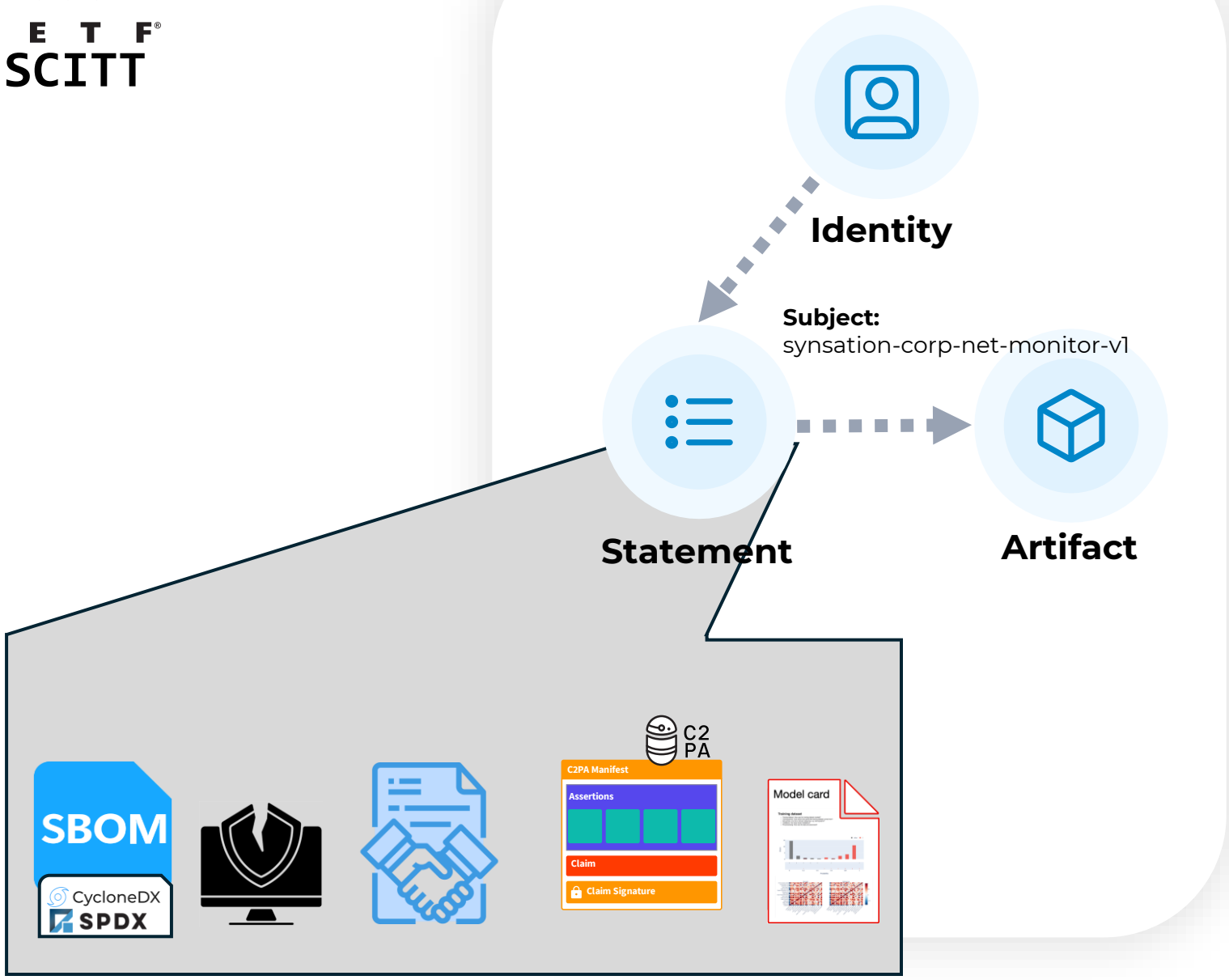
- People (Whoville who's)
- Services
- Processes
- Companies
- Groups
- Anything with any type of identity

SCITT uses x509 as ***an*** example
No intended limitation of identity types
It's up to the SCITT Service to decide what types of identities they'll support

CWT_Claims

```
issuer   : tstr,
subject  : tstr,
*        : int => any
```

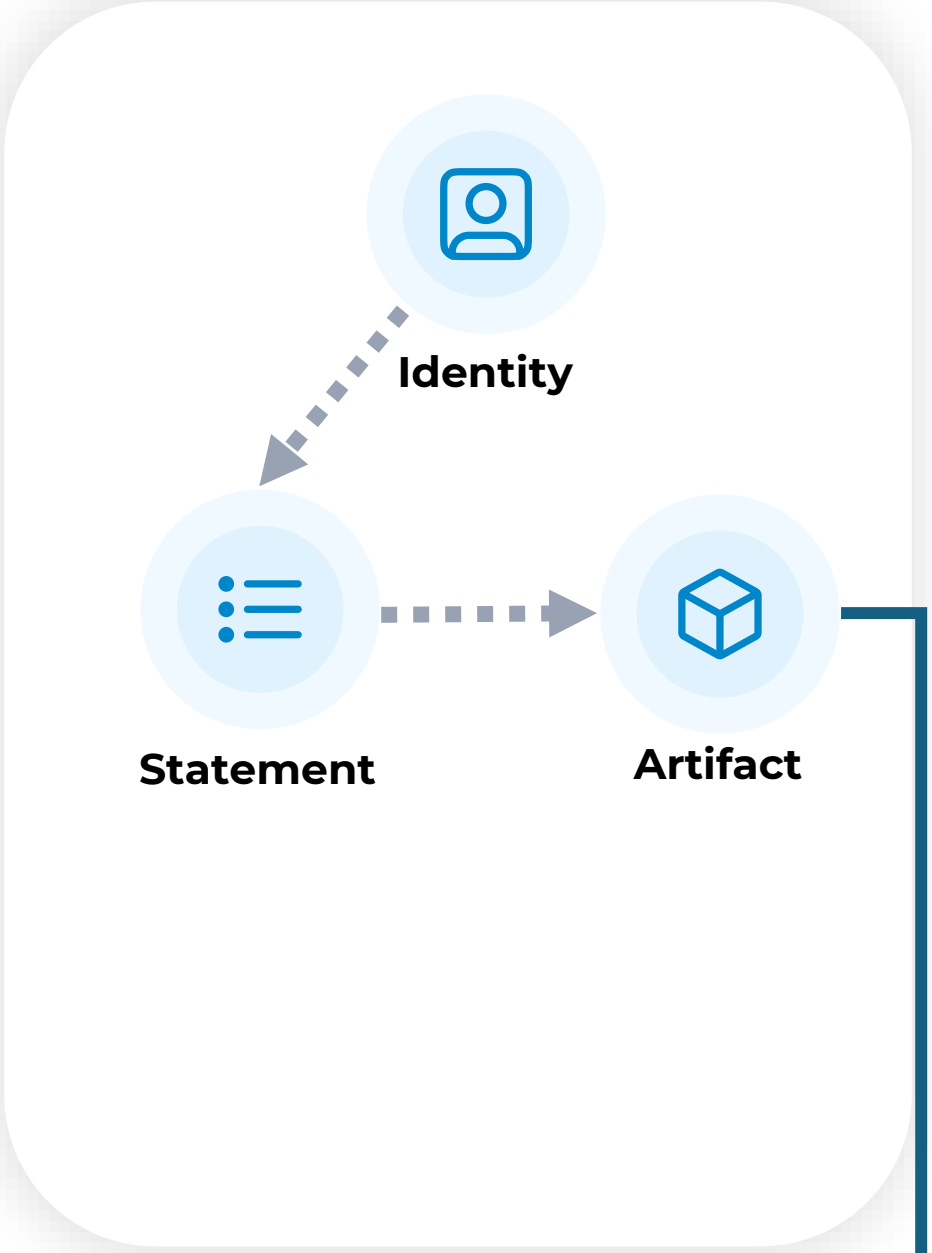
IETF SCITT



What are Statements?

- SBOMs about binaries
  - Test results
  - Compliance to certifications
  - Security Scans
  - VEX Reports
- Contracts about a deal
- C2PA Manifests about digital media
- Responsible AI Claims
  - Model Cards

IETF SCITT



What is an Artifact?

Anything that needs a verifiable statement

- Binary data (software, docker containers)
- AI Models
- vCons
- Digital media (pictures, videos, contracts)
- Physical goods (parts, nuclear waste)
- **`Subject`** is the Artifact Identifier

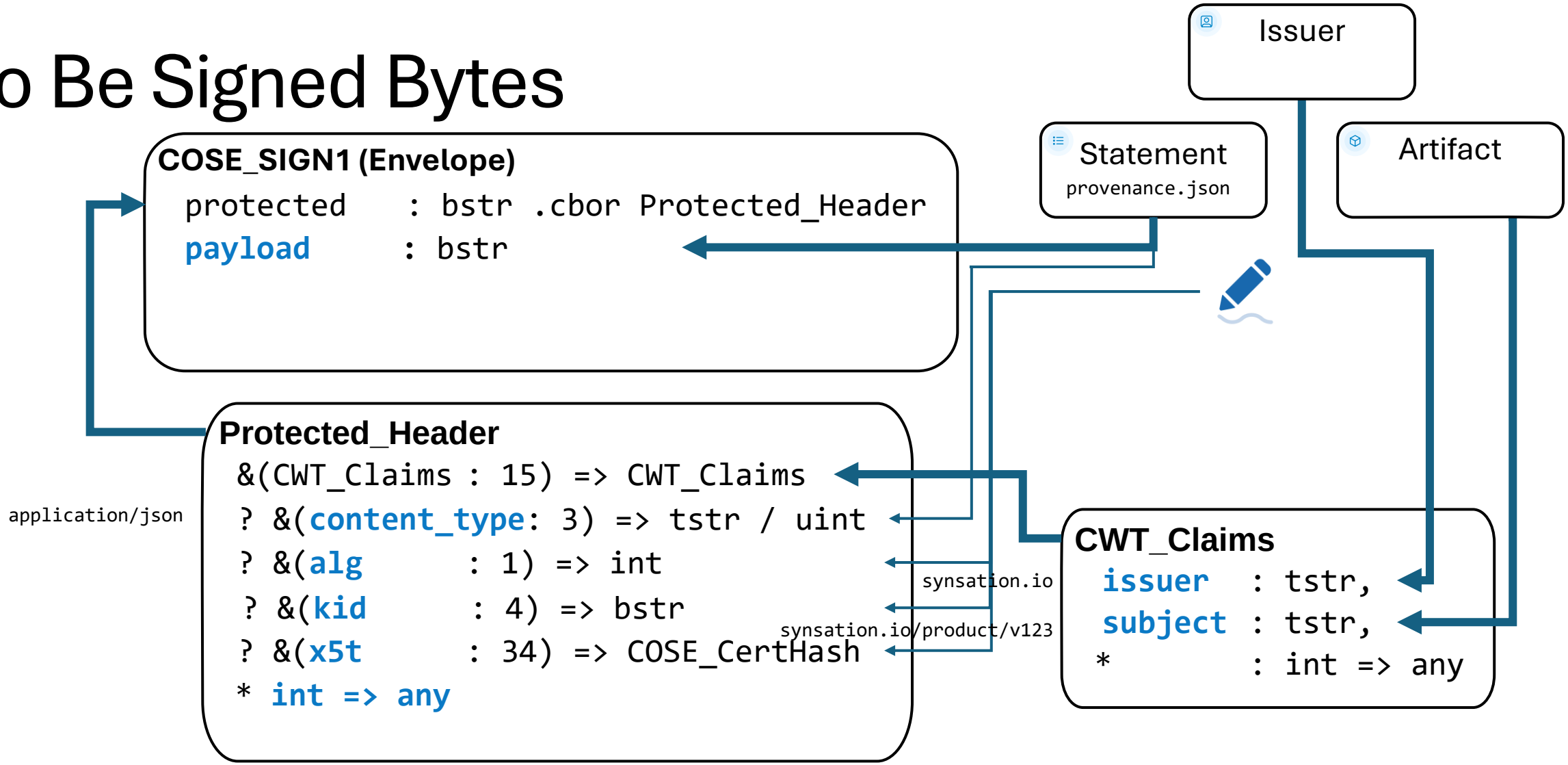
synsation-corp/net-monitor/v1

The format of `Subject` is not part of the SCITT Architecture.
Likely industry specific, and poised for other IETF drafts

CWT_Claims

```
issuer   : tstr,
subject  : tstr,
*        : int => any
```

To Be Signed Bytes



Signed Bytes

COSE_SIGN1 (Envelope)

```
protected   : bstr .cbor Protected_Header
payload     : bstr
```

Issuer

Statement
provenance.json

Artifact

SCITT Statement

COSE_SIGN1 (Envelope)

```
protected   : bstr .cbor Protected_Header
payload     : bstr
signature   : bstr
unprotected : Unprotected_Header
```

Unprotected_Header

```
* int => any
```

Issuer

Statement

Artifact

Registering

COSE_SIGN1 (Envelope)

```
protected   : bstr .cbor Protected_Header
payload     : bstr
signature   : bstr
unprotected : Unprotected_Header
```

DataTrails

SCITT Ledger
Append-only
Log

Transparency Service

How large is the COSE_Sign1 Envelope?

Protected Header	~1k
Unprotected Header	0
Signature	~1k
Payload (Statement)	1k-50gb

2k

~50.002gb

- Is Size the constraint
- Is the Statement already stored somewhere else?
- Do we need to continually pass content for verification?
- What value are we getting by storing the statement in the payload of the Signed Statement

External Storage

Detached Payloads

COSE_SIGN1 (Envelope)

```
protected   : bstr .cbor Protected_Header
payload     : bstr / nil
signature   : bstr
unprotected : Unprotected_Header
```

DataTrails

SCITT Ledger
Append-only
Log
Ledger

Notary
Verify

Transparency Service

IF

External Storage

Unprotected_Header

```
? &statement_location  => tstr "https://sbom.sh/retrieve/45c86..."
* int => any
```

Detached Payloads

COSE_SIGN1 (Envelope)

```
protected   : bstr .cbor Protected_Header
payload     : bstr / nil
signature   : bstr
unprotected : Unprotected_Header
```

DataTrails

SCITT Ledger
Append-only
Log
Ledger

Notary
Verify

Transparency Service

IF

External Storage

COSE_SIGN1 (Envelope)

```
protected   : bstr .cbor Protected_Header
payload     : bstr / nil
signature   : bstr
unprotected : Unprotected_Header
```

Unprotected_Header

```
? &statement_location  => tstr "https://sbom.sh/retrieve/45c86..."
* int => any
```

Content of a SCITT Statement

What is the size and makeup of the statement

Inline content (binary)

Small File *Define Small*

Large file *Define Large*

Collections of files
large and/or small
Likely packaged in another file (zip/tar) or referenced by a manifest

File by Reference: URI to the location: docker image, npm package, vcon, youtube video

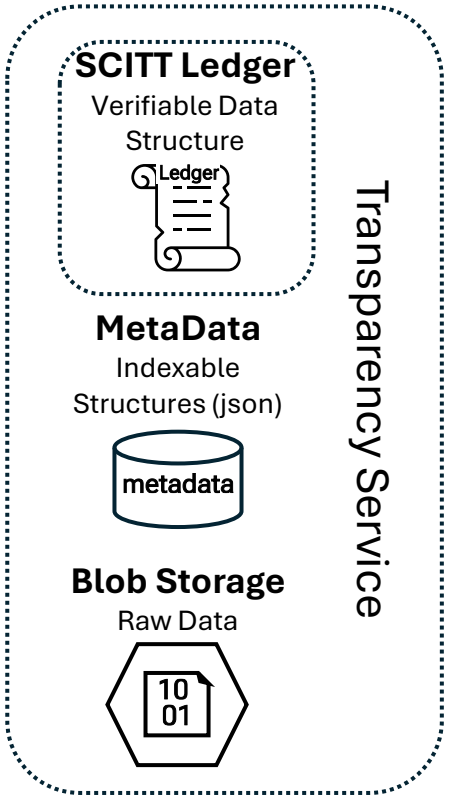
Manifest: Collections of files, each referenced by a unique id (eg: docker image, npm package, vcon, youtube video)

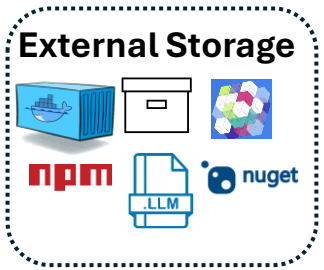
COSE_SIGN1 (Envelope)

```
protected   : bstr .cbor Protected_Header
payload     : bstr / nil
signature   : bstr
unprotected : Unprotected_Header
```

Persistence

Where is the Signed Statement, Metadata and Payload persisted





Content of a SCITT Statement

What is the size and makeup of the statement

Inline content (binary)

Small File

Large file

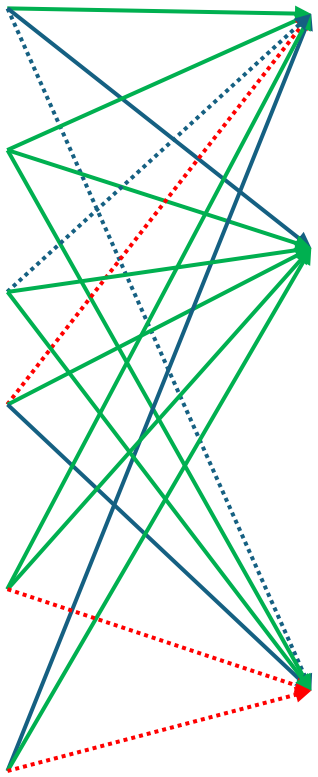
Collections of files

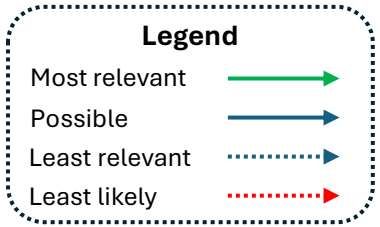
large and/or small

Likely packaged in another file (zip/tar) or referenced by a manifest

File by Reference: URI to the location: docker image, npm package, vcon, youtube video

Manifest: Collections of files, each referenced by a unique id (eg: docker image, npm package, vcon, youtube video)





SCITT Envelope Payload Types

How is the Statement represented within the Signed Statement

[...] Inline:

payload: <statement>
content-type: Type of the payload (application/json, application/bin,)

[###] Hash:

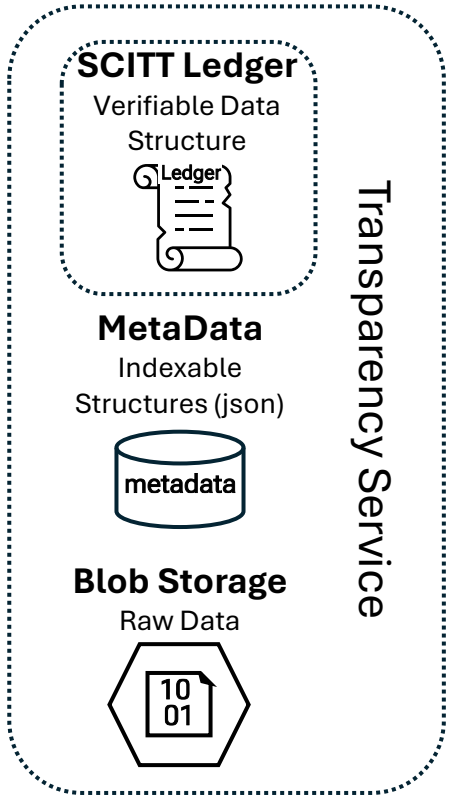
payload: Hash of the content, minimizing the signed-statement size
content-type: Type of the hashed content (application/json, application/bin,)
detached-hash-algorithym: sha-256 | *SHA3-512*
payload-location: added to resolve a possible location for the statement (payload)

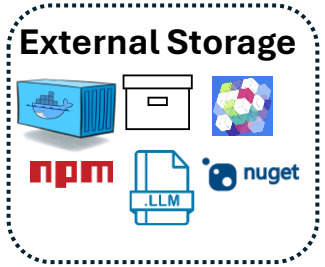
[nul] Detached Payload:

payload: nil
content-type: the type of the detached content (application/json, application/bin,)
payload-location: added to resolve a possible location for the statement (payload)

Persistence

Where is the Signed Statement, Metadata and Payload persisted

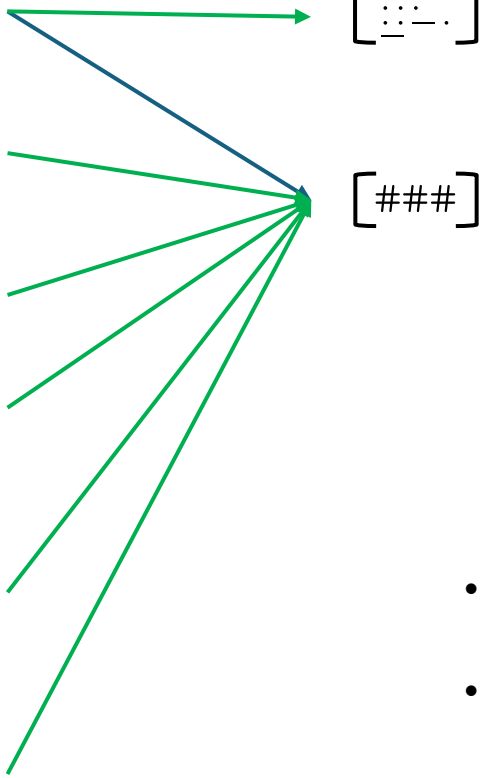




Content of a SCITT Statement

What is the size and makeup of the statement

- Inline content (binary)
- Small File
- Large file
- Collections of files
  large and/or small
  Likely packaged in another file (zip/tar) or referenced by a manifest
- File by Reference: URI to the location: docker image, npm package, vcon, youtube video
- Manifest: Collections of files, each referenced by a unique id (eg: docker image, npm package, vcon, youtube video)



Legend

- Most relevant
- Possible
- Least relevant
- Least likely

SCITT Envelope Payload Types

How is the Statement represented within the Signed Statement

Inline:

`payload:` <statement>
`content-type:` Type of the payload (`application/json`, `application/bin`,)

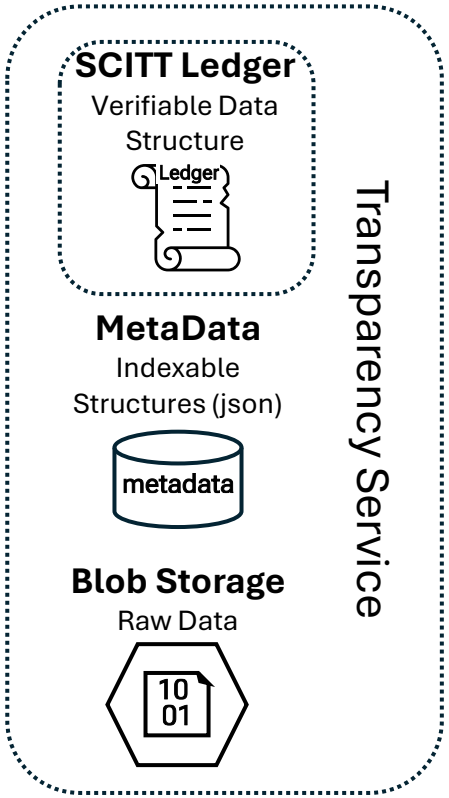
Hash:

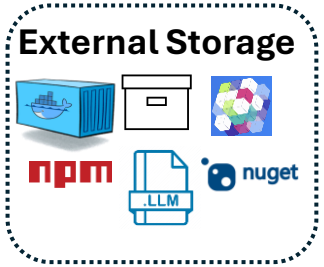
`payload:` Hash of the content, minimizing the signed-statement size
`content-type:` Type of the hashed content (`application/json`, `application/bin`,)
`payload-hash-algorithm:` sha-256 | *SHA3-512*
`payload-location:` added to resolve a possible location for the statement (payload)

- Signed Statement Payload consistent across services
- Never wonder what size constraint will fail
- Builds upon existing storage services
- Transparency Services can provide storage services, they just fill the payload-location with their storage url

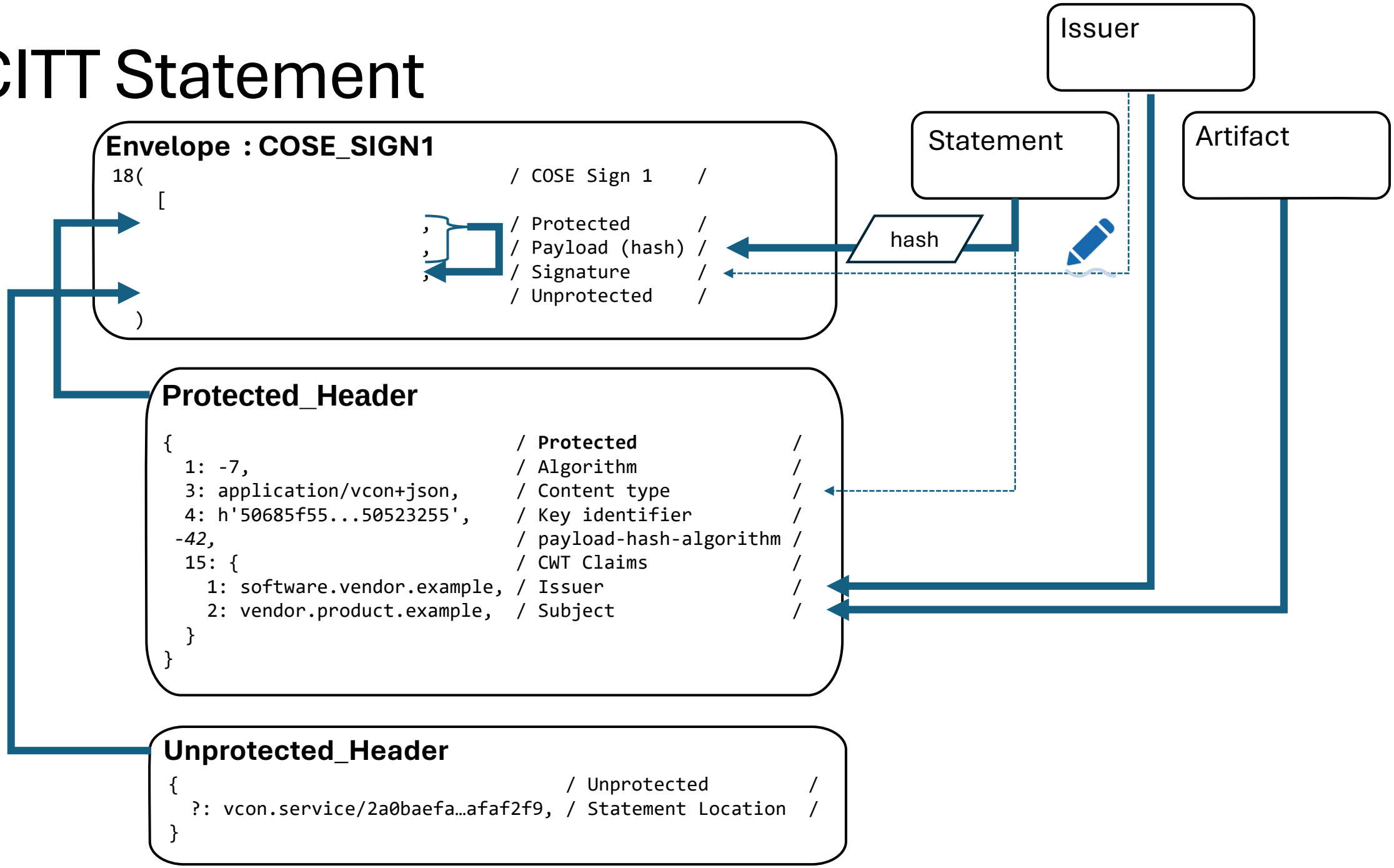
Persistence

Where is the Signed Statement, Metadata and Payload persisted





SCITT Statement



SCITT – COSE

Managing SCITT Statements as COSE Payloads

Hashed Payloads