IETF SCITT Architecture

IETF 117
July 24th, 2023
San Franciso, USA

Concerns going into 117

With such a general technology we run a high risk of running into dead ends.

Several of the stickier conversations we've had in recent interims have been concrete instances of a small number of abstract problems.

Giving these a lot of thought and discussion, we have identified a couple of key areas that promise resolution to a lot of the group's issues.

Henk identifying a potential dead-end

Theme for 117

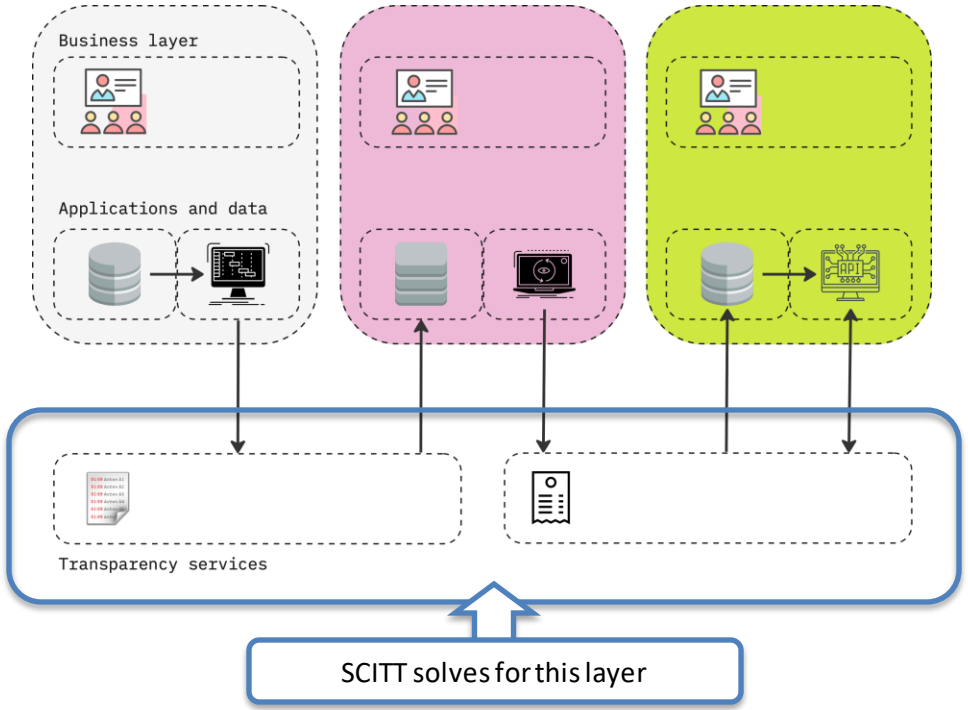
"We are not the curators of a thousand semantics" 😄

Architectural context

SCITT is all about maintaining long term integrity and trust in artifacts that move around between participants.

The business layer contains a lot of real-world messiness, and the applications that businesses use necessarily embody very specific and non-portable implementations of general concepts. But good news: if they need semantic interoperability of the data they exchange, chances are they've already worked that out!

What we MUST solve for is the general case and the 3 promises of SCITT at the data layer. What we MUST avoid is trying to encode all possible application use cases directly into the standard.



Resulting activity at 117

Spec topics

- *Get to grips with identity:* What do we actually NEED to define? What is fundamentally understood and enforced in SCITT vs simply 'supported' and faithfully transported?
- *Feeds emerge as an important common factor:* How do we locate them, and what properties do they need in order to support our various locator/access control use cases?

Hackathon code proof

- Prove that the very generic (but reliable) SCITT model can support end to end application use cases without directly encoding application semantics
- Harmonize terminology/commands in the emulator with latest clarity issues.
- Make some progress towards interoperable claim generation
- Make progress towards indexing of data fields and metadata fields (reinforce where the architectural layer should be)

Recent Updates I

- **Improved** significantly on **terminology consistency** and general **context consistency**
  - e.g., added the core term of "**append-only log**"
- More focus on **Producer & Consumers**
- **Removed** the existing **use-case placeholders** ("fish & chips") and refer to separate **use-case I-D**:
- https://datatracker.ietf.org/doc/draft-ietf-scitt-software-use-cases/00/
- BTW, the use cases seem to have consensus and are done. WGLC?

Recent Updates II

- Simplified text on **registration policies**
  - Checks that are performed before a Signed Statement is registered given a set of input values
  - Is Implementation is **left to the provider** of the Transparency Services and its users?
  - Current minimal baseline: **Gatekeeping by Issuer**

- Is this the way to go?

Recent Updates III: Receipts

- **SCITT Receipt** is now a **profile** of a Signed Inclusion Proof: a stack of I-Ds in COSE WG:
  - https://datatracker.ietf.org/doc/draft-steele-cose-merkle-tree-proofs/01/
  - https://datatracker.ietf.org/doc/draft-birkholz-cose-cometre-ccf-profile/

```
Receipt = [
    version: int,
    ts_identifier: tstr,
    proof: SignedMerkleTreeProof
]
```

- SCITT defines **additional protected header attributes** for the COSE **SignedMerkleTreeProof** of the CoMETRE I-D

Recent Updates IV: Additional Header Attributes

```
Protected_Header = {
    390 => int                  ; SCITT Receipt Version
    394 => tstr                 ; DID of Transparency Service
    ? 395 => RegistrationInfo   ; Registration policy information
    2=> [+ label]               ; Critical headers
    ? 4 => bstr                 ; Key ID
    ? 33 => COSE_X509           ; X.509 chain
}

RegistrationInfo = {
    ? "registration_time": uint .within (~time),
    * tstr => any
}
```

Recent Updates V: API(s)

- Where do APIs go?
- In SCITT's case: draft-birkholz-scitt-scrapi
  - SCITT Reference APIs
  - Takes into account RFC 9205
  - Starting with REST API
  - All API content will move out of the Architecture I-D

Current Construction Site: Feeds "We are not the curators of a thousand semantics"

- Why?
- How to identify the "topic" you want to register (or re-discover via index services) Transparent Receipts for requires a standardized and reliable mechanism
- Mechanism must be based on and continuously tested using large sets of real-data: e.g., VRF, NVD, STIX, VEX, CyclonDX
- Will try to keep the feed value structure as simple as possible based on lessons learned from real-data sets
- Linked-Data is not out-of-scope, but also not a target

Identity

- Why?
  - Issuers (organizations who create artifacts)
  - Artifacts (products produced over time)
- Challenges
  - How are keys bound to identifiers over time.
  - How are keys represented?
  - How are keys discovered?

Discovering Identifiers

kid -> application/jwk+json

kid -> application/cose-key

x5u -> application/jwk+json
x5u -> application/cose-key
x5u -> application/cose-key-set

- Decision:
  - Will enable interoperability with OIDC, DID and x509 via URIs
  - Will not open the can of worms that is "defining identity" in SCITT, but will absopositivitly support existing envelope formats that support identity representation and identifiers.
    (Note! These are DIFFERENT things /w different applications)

Issuers

Issuer identifiers are not necessarily unique legal entities.

(iss: vendor0.example) and (iss: vendor1.example) could both be owned by owner.example ... that might change over time.

"signed claims" establish "relationships" between "identifiers".

For example:

did:web:gs1.ca (GS1 Canada)
is https://id.gs1.org/01/07541234555551

Identity Providers

Transparency Services can leverage Identity Providers to understand, issuer and subject identifiers.

"identity assertions" establish "relationships" between "identifiers" and "names".

See OpenID Connect and "id_token"s.

See GS1 Licenses.

See Developer Certificates.

Feeds as URLs

Figure 3-2 A Web URI can act both as a globally unambiguous name for something, as well as providing an easy way to retrieve Web resources (e.g. information) relating to the identified thing

Figure 3-3 provides a brief overview of the internal structural elements of a Web URI:

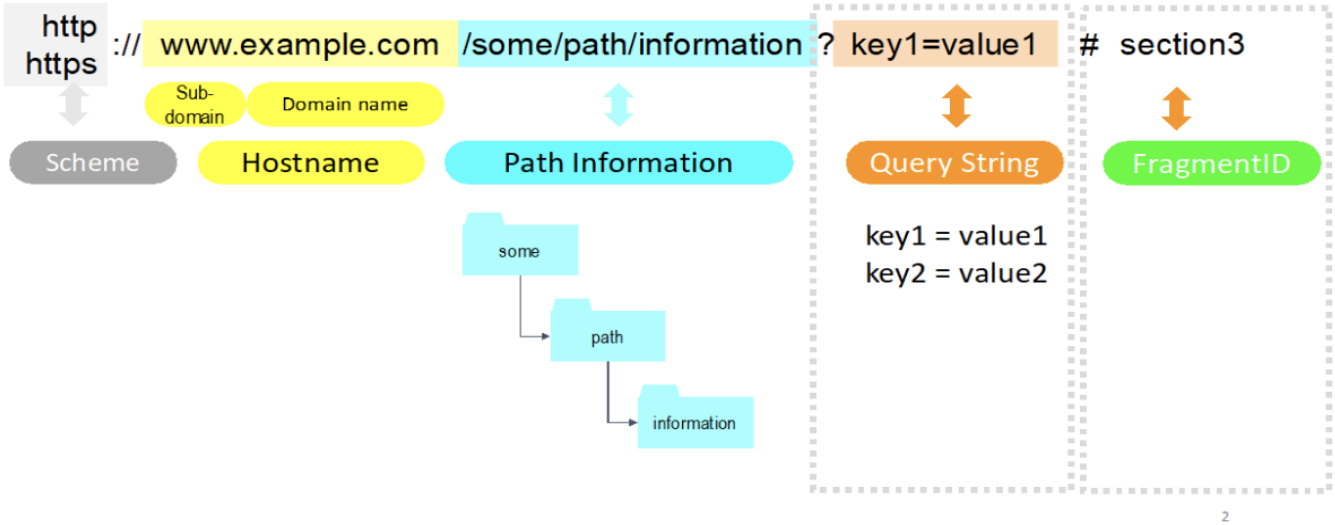


Figure 3-3 Internal structure of a Web URI

https://ref.gs1.org/standards/digital-link/uri-syntax/

Issuer Public Keys over Time

Considering feeds for receipts and artifacts....

transparency1/issuer/vendor1/public-keys

-> application/jwk-set+json with application/scitt.receipt+cose

transparency1/issuer/vendor2/public-keys

-> application/cose-key-set with application/scitt.receipt+cose

... receipts for witnessed public keys for an issuer... tells an auditor which keys have been authoritative for an issuer over time.

Endorsements over Time

Considering feeds for receipts and artifacts...

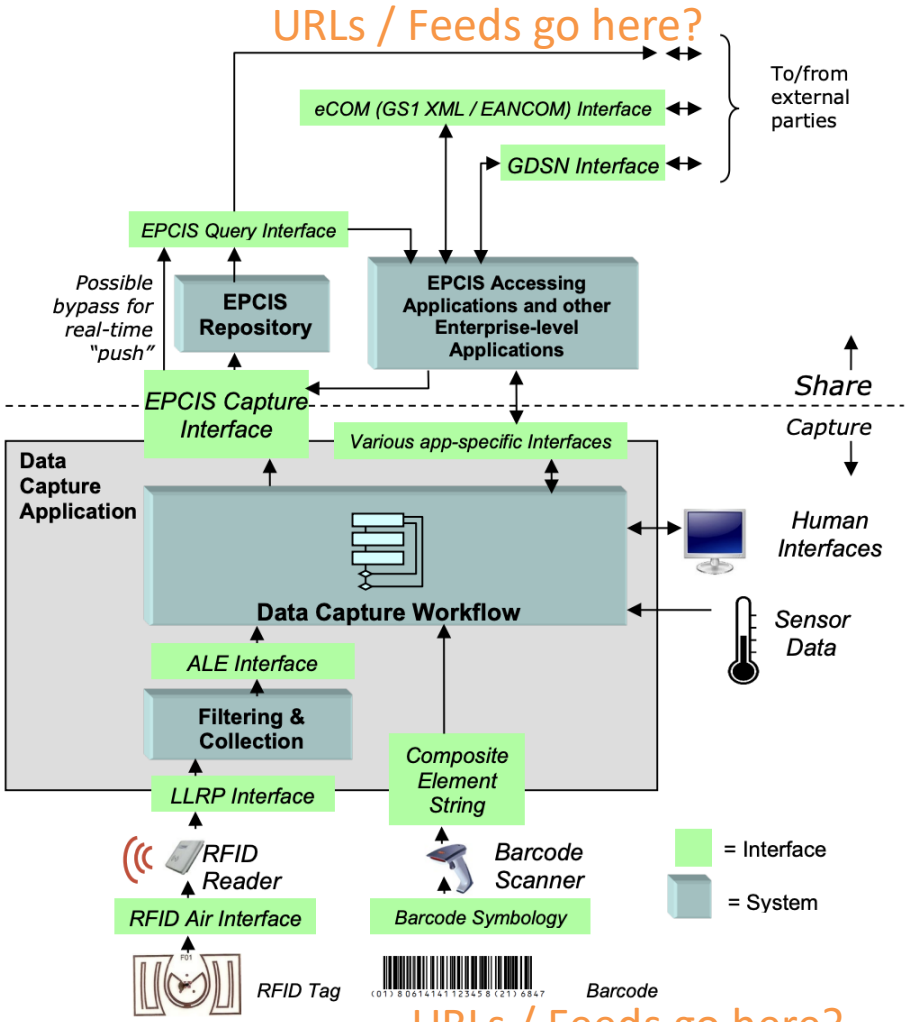
transparency1/issuer/vendor1/products/product1
transparency1/issuer/vendor2/products/product1

... like CPE ? Not exactly...

"criteria": "cpe:2.3:a:*:openssl:version:*:*:*:*:*:*:*:*",

<xsd:pattern value="cpe:2\.3:[aho*\-](:(((\?*|*?)([a-zA-Z0-9\-\._]|(\\[*\?!"#$$%&'\(\)\+,/:;<=>@\[\]\^`\{\|}~]))+(\?*|*?))|[*\-])){5}(:(([a-zA-Z]{2,3}(-([a-zA-Z]{2}|[0-9]{3}))?)|[*\-]))(:(((\?*|*?)([a-zA-Z0-9\-\._]|(\\[\\*\?!"#$$%&'\(\)\+,/:;<=>@\[\]\^`\{\|}~]))+(\?*|*?))|[*\-])){4}"/>

Feed Workflows?



https://ref.gs1.org/standards/epcis/