Decentralized Identifier WG F2F Sessions

Day 1: January 29, 2020
Chairs: Brent Zundel, Dan Burnett
Location: Microsoft Schiphol

Welcome!

- Logistics
- W3C WG IPR Policy
- Agenda
- IRC and Scribes
- Introductions & Dinner

Logistics

- **Location**: “Spaces”, 6th floor of Microsoft Schiphol
- **WiFi**: SSID Publiek_theOutlook, pwd Hello2020
- **Dial-in information**: +1-617-324-0000, Meeting ID
- **Restrooms**: End of the hall, turn right
- **Meeting time**: 8 am - 5 pm, Jan. 29-31
- **Breaks**: 10:30-11 am, 12:30-1:30 pm, 2:30-3 pm
- **DID WG Agenda**: https://tinyurl.com/didwg-ams2020-agenda (HTML)
- **Live slides**: https://tinyurl.com/didwg-ams2020-slides (Google Slides)
- **Dinner Details**: See the “Dinner Tonight” slide at the end of each day

W3C WG IPR Policy

- This group abides by the W3C patent policy https://www.w3.org/Consortium/Patent-Policy-20040205
- Only people and companies listed at https://www.w3.org/2004/01/pp-impl/117488/status are allowed to make substantive contributions to the specs
- Code of Conduct https://www.w3.org/Consortium/cepc/

Today's agenda

8:00	Breakfast	
8:30	Welcome, Introductions, and Logistics	Chairs
9:00	Level setting	Chairs
9:30	Security issues	Brent
10:15	DID and IoT	Sam Smith
10:45	Break	
11:00	Multiple Encodings/Different Syntaxes: what might we want to support	Markus
11:30	Different encodings: model incompatibilities	Manu
12:00	Abstract data modeling options	Dan Burnett
12:30	Lunch (brief "Why Are We Here?" presentation)	Christopher Allen
13:30	DID Doc Extensibility via Registries	Mike
14:00	DID Doc Extensibility via JSON-LD	Manu
14:30	Break	
14:45	Extensibility of DID Documents Discussion	Chairs/Group
15:15	Metadata	Ganesh Annan
16:15	Open Slot	

IRC and Scribes

- Meeting discussions will be documented
  - Text Chat: http://irc.w3.org/?channels=did
  - IRC://irc.w3.org:6665/#did
- Telecon info
  - https://lists.w3.org/Archives/Member/member-did-wg/2020Jan/0001.html
  - meeting 646 787 287
  - Phone number

	Wednesday	Thursday	Friday
AM 1	Manu	Markus	Kaliya
AM 2	Joe	Oskar	David E
PM1	Ken		Drummond
PM2	Yancy		Juan (in Dvorak)

<JoeAndrieu> q+ to comment on biometrics
<brent> ack JoeAndrieu
<Zakim> JoeAndrieu, you wanted to comment on biometrics

Introductions


```
thread
   Z = X+Y
   {Browse Z}
end
thread X = 40 end
  thread Y = 2 end
```

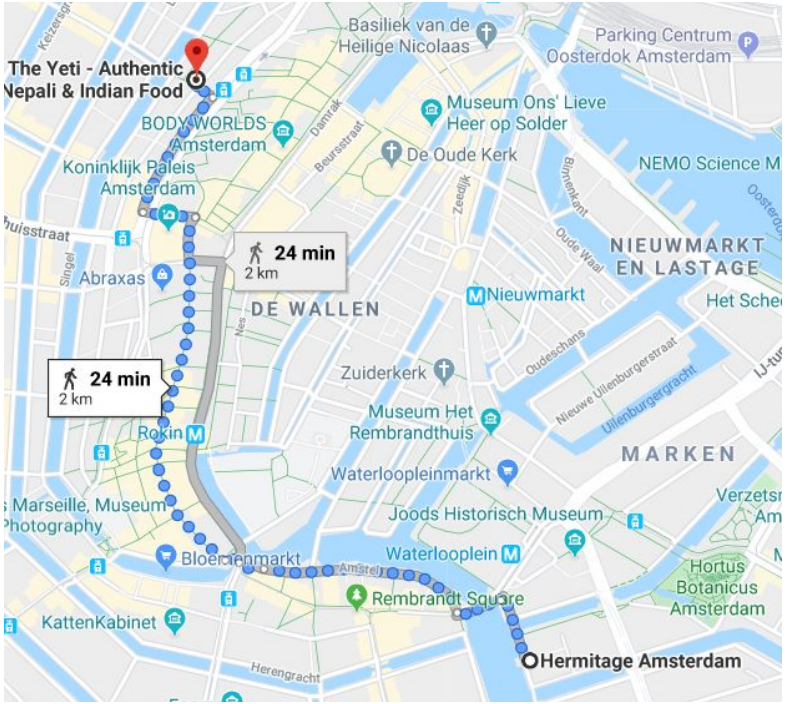
Dinner 30 Jan - 19:00 (Thursday)

Restaurant: The Yeti - Authentic Nepali & Indian Food

Location: Spuistraat 54D, 1012TV Amsterdam

Website: http://theyeti.nl/

Booking Status: Booking confirmed.



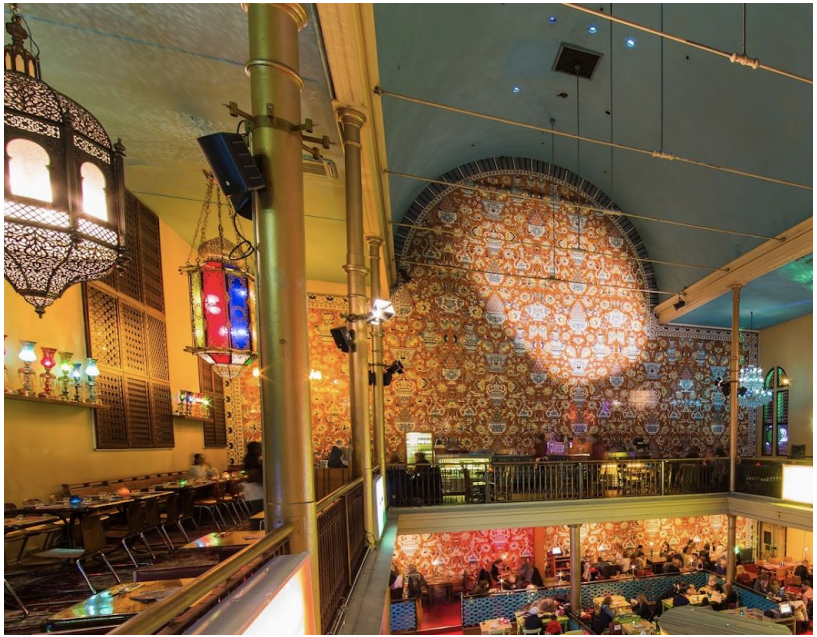
Dinner Wednesday

Time: **8pm** Expected Count: 21

The Bazar, (300 bus to **Schipol**, intercity train to **Zuid**, Metro 52 Noord to **De Pijp Station**)
Albert Cuypstraat 182, Amsterdam +31 20 675 0544

This beautiful Arabic restaurant is housed inside of an enormous renovated church. Its dining area is spread over two massive floors that are covered in extravagant motifs and stained glass artwork. Bazar's couscous dishes are absolutely delicious and the restaurant also offers many other traditional Middle Eastern meals. Famously, its waiters carry orders on huge silver platters that they keep elevated above their heads. NO ALA CARTE, 5 CHOICES (including vegetarian)

https://goo.gl/maps/atGhM9jwzFjSo52J9



Potential topics for the “Open Topics” sessions

- Service endpoints
- Liaisons with other W3C groups:
  - Web auth
  - WoT
  - Privacy IG
  - Web Payment
  - others?
- Metadata - identify categories of data/metadata by reviewing potential fields, where they come from, etc.
- Review and revise Manu’s “5 points” until we have group agreement
- Explicit Cryptographic support?
- Nuances of extension interoperability

Level Setting (Dan, 30 min)

DID WG Mission and Goals

- “... standardize the DID URI scheme, the data model and syntax of DID Documents, which contain information related to DIDs that enable the aforementioned initial use cases, and the requirements for DID Method specifications.”

DID WG Scope

- Define the DID URI scheme.
- Recommend a data model and syntax(es) for the expression of Decentralized Identifier Documents, including one or more core vocabularies.
- Recommend a set of requirements for DID Method specifications that are conformant with the data model and syntax(es).
- Provide a rubric of decentralized characteristics for DID Method specifications.
- Concentrate their efforts on the initial use cases with a particular focus on enabling future specification and implementation of Identity and Access Management.
- Define extension points enabling authentication, signing and cryptography mechanisms.
- With the initial use cases document as input, the WG will produce a NOTE at the end of the process that is a refined Use Cases document.
- Establish a deterministic mapping between DID method identifiers and the resolution process used to resolve that DID method.

DID WG Out of Scope

- Authentication or Authorization Protocols
- Browser APIs
- Specific DID Method specifications or Protocol specifications
- "Solving Identity" on the Web
- Defining specific authentication, signing, or cryptography mechanisms. Scope is limited to defining extension points for these mechanisms.

Charter Deliverables

- Recommendation-track Specification
  - Decentralized Identifiers v1.0
- W3C Notes
  - Decentralized Identifier Use Cases v1.0
  - Decentralized Characteristics Rubric v1.0
- Other Deliverables
  - Test Suite and Implementation Report

Documents and Background

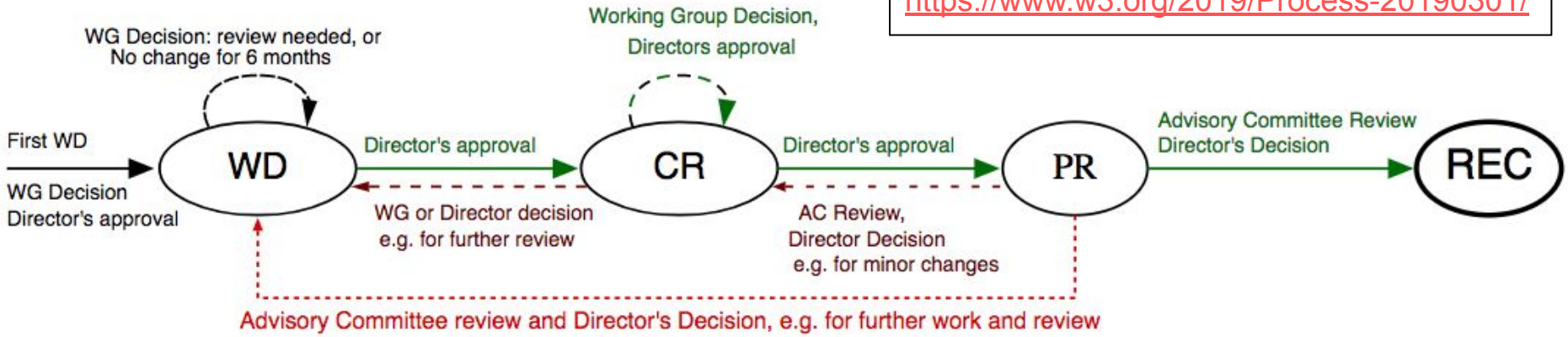
- Home: https://www.w3.org/2019/did-wg/
- Charter: https://www.w3.org/2019/09/did-wg-charter.html
- Primer: https://w3c-ccg.github.io/did-primer/

W3C Technical Report Process

- Working Draft (WD) - does not imply consensus
- Candidate Recommendation (CR)
  - Entry - to publish as CR, the document is expected to be feature complete, have had wide review, and must specify the implementation requirements needed to exit
  - Exit - to exit CR (and move to PR), the document must satisfy the stated implementation requirements; it must also not have made any substantive change not warned about upon entry
- Proposed Recommendation (PR)
  - Basically a one-month sanity check during which the AC is encouraged to have any final review and discussion, but if anything major happens it’s a fail (requiring a move back to CR or earlier)
- Recommendation - Done
  - But errata are possible

Timing

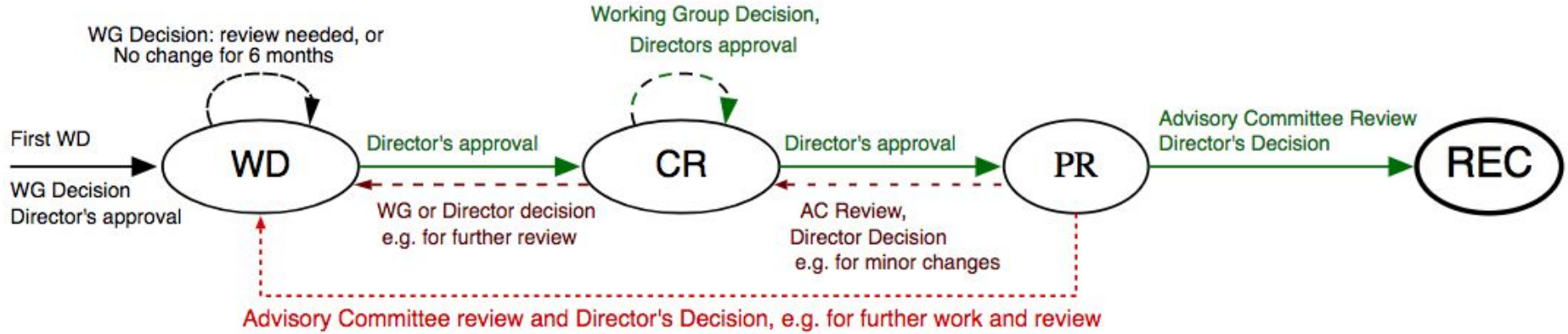
https://www.w3.org/2019/Process-20190301/

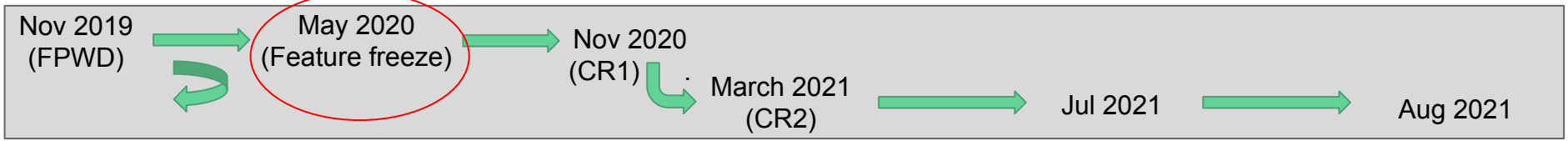


2.3 Timeline

Specification	FPWD	CR	PR	Rec
Decentralized Identifier Use Cases & Requirements (NOTE)	November 2019			August 2021
Decentralized Characteristics Rubric (NOTE)	December 2019			September 2021
Decentralized Identifiers Data Model and Syntax(es)	November 2019	November 2020	July 2021	August 2021
Note: The group will document significant changes from this initial schedule on the group home page.				

Timing of our primary spec





Decentralized Identifiers Data Model and Syntax(es)	November 2019	November 2020	July 2021	August 2021

https://www.w3.org/2019/Process-20190301/

Goals for this meeting

- JSON/JSON-LD/Abstract data model agreement and plan
  - Day 1: relevant factors - making sure we all have all the knowledge to make a good decision
  - Day 2: understanding what we need to decide, then deciding
- Other goals:
  - Progress on metadata
  - Progress on matrix parameters and query parameters
  - Progress on Rubric and Use Cases and Requirements

A story

Security Issues (Brent, 45 min)

What are some security concerns?

- Each DID Method has a different solution for verification of control over a DID
  - With different protocols
  - and different infrastructure
- What are others?
  - Implementation complexity vs. extensibility
  - cryptographic advancements and not limiting ourselves to current tech
  - conflating control of DID vs authentication with the DID
- Outside of our scope, but things we must be aware of:
  - DID Resolution has its own issues
    - method-specific code
    -
  - Trusting the code that verifies

What should we do about them?

- Require all DID methods to follow a common practice for establishment of control authority?
- Guide for how to layer security?
- Recommend best practices in the security considerations section?
- Warn against bad practices in the security considerations section?
- A Security Rubric for determining the security characteristics of different DID methods?
- A Security section of the Test Suite?
- Design an ideal secure DID method as an example

DID and IOT (Sam, 30 min)

DID and IoT

2020/01/29
Discussion W3C DID WG
Samuel M. Smith Ph.D.
https://github.com/SmithSamuelM/Papers
sam@samuelsmith.org

IoT Characteristics

Very large number of devices (80 Billion by 2025) (IPv6 P2P 5G)

Most are limited resource devices (memory, compute, power, network)

CoAP (Constrained Application Protocol) RFC 7252, 7959, 8323

Small Footprint IP/Web Stack UDP Alternative to HTTP

CBOR for Serialization, DTLS for security (datagram TLS)

https://coap.technology

IIoT vs IoT (commercial vs home)

Operational integration in the cloud (siloed vendor/function device families)

Future operational integration in the edge

Data integration and analytics in the cloud

Future data integration and analytics in the edge

IoT Characteristics

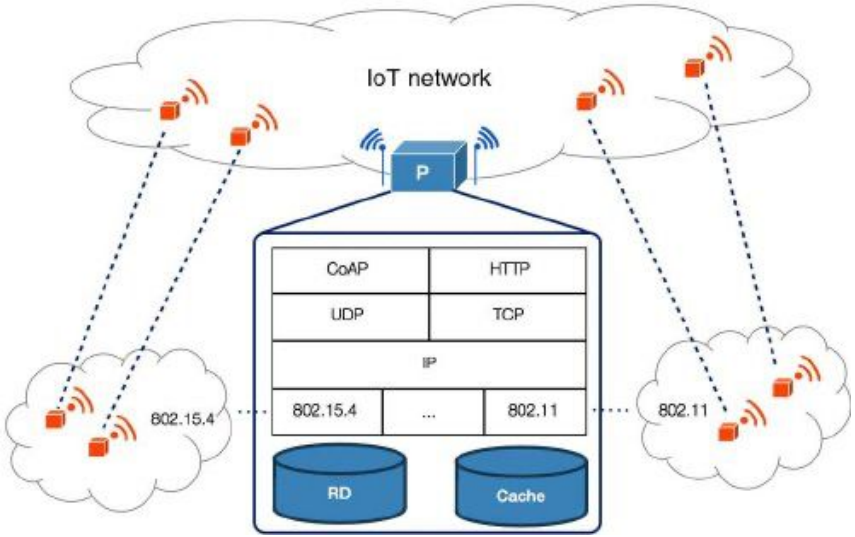
Direct IP enabled: IP Stack on top of communications channel. Typically wifi, low power wifi.

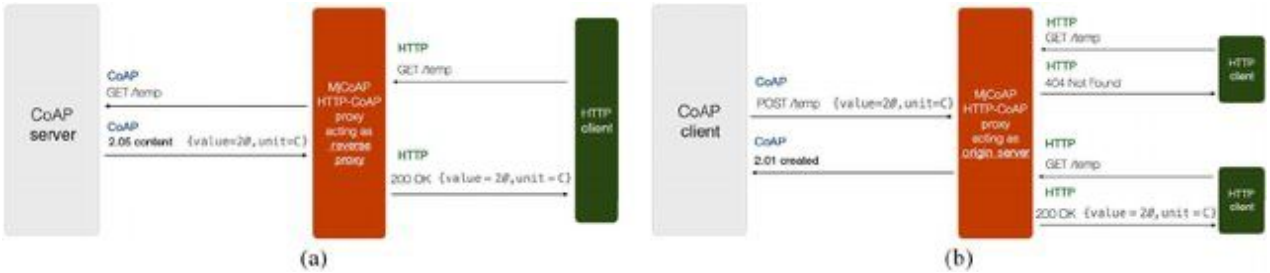
Indirect IP enabled: Non-IP Stack through IP Gateway device as IP Proxy. Bluetooth, Zigbee, ZWave, CoAP, 6LowPAN etc.

Authenticated control more important than confidentiality (signing vs encryption)

Gateways provide encryption, proxy, IP mapping

Gateway





Cirani, S., Davoli, L., Ferrari, G. et al., "A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things," IEEE Internet of Things Journal, vol. Vol 1, No. 5, 2014/10/01

IoT Security Issues

Physical security

Default credentials (username, password)

Self-provisioning without unique credentials

Unsigned data

Unencrypted data

LAN vs WAN access

Edge vs Cloud integration

Generally not self-sovereign

IoT Provisioning

Large number of devices make provisioning complex

Pre-loaded or dynamic self-provisioning systems

Physical access provisioning via device labels:

Number sticker, Scratch-off ID, Barcode ID, QR-Code ID

Edge vs Cloud provisioning integration (LAN vs WAN access)

Cloud platform provisioning (phone home)

IoT Provisioning

Sensors: signed telemetry stream

Actuators: authorized input control device

Self-contained bootstrap to signing key-pair for authentication (self-certifying, self-bootstrapping)

Contextual authentication (location, nearby devices, attributes)

Ephemeral (non-rotatable) key-pair sufficient in most cases

Low value per device plus low resources per device = abandon key-pair vs rotate

Signing key and sometimes encryption key to destinations

Rarely pairwise but pairwise useful in the future

IoT Provisioning

Edge Future:

Non-siloed edge integration (site dependent)

Sensors:

Authorized telemetry stream destinations (on-device or on-gateway)

Actuators:

Configuration table of authorized input control sources (on-device or on-gateway)

IETF/TCG Alternative Standards Stack

Implicit Identifier (Device ID, DID)

Limited resource compute devices (inexpensive)

TCG, "Implicit Identity Based Device Attestation," Trusted Computing Group, vol. Version 1.0, 2018/03/05

https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf

Self-certifying root Device ID based on root public/private key pair generated at first power up

Firmware tamper results in new Root Device ID

Aliased IDs (derived public/private key pairs) from root Device ID

IETF/TCG Alternative Standards Stack

Interoperable Remote Device Configuration Attestation

"Remote ATtestation proceduresS (RATS) WG," IETF,

https://datatracker.ietf.org/wg/rats/about/

Working Group, R. A. T. S., "Network Device Attestation Workflow," IETF Internet-Draft, 2019/12/03

https://tools.ietf.org/html/draft-fedorkow-rats-network-device-attestation-01

Interoperable Authentic Device Configuration Attestations

CBOR/JSON CWT JWT FIDO

IoT Interoperability & Extensibility

oBIX, XML based legacy 1990's

Proprietary (numerous walled gardens)

Industry Consortia (numerous)

IoT-EPI (Euro Big tent approach) TagITSmart O-MI O-DF etc https://iot-epi.eu/wp-content/uploads/2018/07/Advancing-IoT-Platform-Interoperability-2018-IoT-EPI.pdf

Project Haystacks: Tag Model

https://project-haystack.org/doc/TagModel

Portable to almost any encoding

Broad industry adoption due to ease of implementation of tag model

HTO (Haystacks Tagging Ontology) Semantic Overlay https://www.vcharpenay.link/publications/2015-iot.pdf

Brick: https://brickschema.org http://www.vcharpenay.link/hto/doc.htm

Semantic overlay approach portable to almost any RDF encoding

ASHRAE Haystacks plus HTO-Brick Standard

Scope and Mission wrt IoT Support

Should W3C DID WG fully embrace the IoT world

Broad IoT adoption may require support for:

- Constrained encodings such as CBOR
- Self-bootstrapping identifiers
- Compatibility with IETF/TCG standards for implicit identity

Discussion

References

CoAP

"CoAP RFC 7252 Constrained Application Protocol," CoAp.technology, https://coap.technology

"The Constrained Application Protocol (CoAP) RFC 7252," IETF, 2014/06 https://tools.ietf.org/html/rfc7252

"Block-Wise Transfers in the Constrained Application Protocol (CoAP) RFC 7959," IETF, 2016/08/01 https://tools.ietf.org/html/rfc7959

"CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets RFC 8323," IETF, 2018/02/01 https://tools.ietf.org/html/rfc8323

Cirani, S., Davoli, L., Ferrari, G. et al., "A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things," IEEE Internet of Things Journal, vol. Vol 1, No. 5, 2014/10/01 https://www.researchgate.net/publication/269702679_A_Scalable_and_Self-Configuring_Architecture_for_Service_Discovery_in_the_Internet_of_Things

IoT-EPI

https://iot-epi.eu/wp-content/uploads/2018/07/Advancing-IoT-Platform-Interoperability-2018-IoT-EPI.pdf

Trusted Computing Group:

TCG, "Implicit Identity Based Device Attestation," Trusted Computing Group, vol. Version 1.0, 2018/03/05 https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf

Working Group, R. A. T. S., "Network Device Attestation Workflow," IETF Internet-Draft, 2019/12/03 https://tools.ietf.org/html/draft-fedorkow-rats-network-device-attestation-01

"Remote ATtestation proceduresS (RATS) WG," IETF, https://datatracker.ietf.org/wg/rats/about/

Morning break (10:45-11:00)

Multiple Encodings/Different Syntaxes (Markus, 30 min)

What might we want to support?

- JSON-LD
- JSON
- IPLD
- CBOR
- XML
- PDF
- ASN.1
- YAML
- XDI
- CSV
- ?

JSON-LD

- Great:
  - Semantic Web
  - permissionless extensibility
  - compatibility with JSON-LD VCs, LD proofs, Solid/WebID, ActivityPub, ZCAP-LD

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
      "id": "did:example:123456789abcdefghi#keys-1",
      "type": "RsaVerificationKey2018",
      "controller": "did:example:123456789abcdefghi",
      "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "service": [{
      "id":"did:example:123456789abcdefghi#vcs",
      "type": "VerifiableCredentialService",
      "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

JSON

- Great:
  - ubiquitous support, familiar to developers
  - no external network dependency (download via HTTPS, IPFS, ..)
  - compatibility with JOSE, OIDC, DIDComm

```
{
  "id": "did:secp256k1:03fdd57adec3d438ea237fe46b33ee1e016eda6b585c3e27ea66686c2ea5358479",
  "publicKey": [{
      "id": "#keys-1",
      "type": "EcdsaSecp256k1VerificationKey2019",
      "controller": "did:secp256k1:03fdd57adec3d438ea237fe46b33ee1e016eda6b585c3e27ea66686c2ea5358479",
      "publicKeyHex": "03fdd57adec3d438ea237fe46b33ee1e016eda6b585c3e27ea66686c2ea5358479"
  }]
}
```

IPLD

- Great:
  - content-addressable, distributed storage
  - location- and protocol-independent
  - censorship-resistant


```
{
  "@context": {
      "/": "zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq"
  },
  "id": "did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew",
  "publicKey": [{
      "curve": "ed25519",
      "expires": "2019-12-01T03:00:00Z",
      "publicKeyBase64": "qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg=",
      "type": "EdDsaPublicKey"
  }],
  "proof": {
      "/": "z43AaGF42R2DXsU65bNnHRCypLPr9sg6D7CUws5raiqATVaB1jj"
  }
}
```


(technically dagCBOR) serialized to JSON

CBOR

- Great:
  - compact data and software
  - great for IoT
  - easy to map JSON to CBOR

But do we want DID documents in plain CBOR, CBOR-LD, or CBOR-IPLD?

What do we mean by data model, format, syntax, encoding, representation, etc.

XML

- Great:
  - data types, namespaces (?)
  - XML Schema, XSLT, XPath, XQuery

```
<diddocument did="did:example:123456789abcdefghi">
  <authentications>
      <authentication type="RsaVerificationKey2018">
            <controller>did:example:123456789abcdefghi</controller>
            <publicKeyPem>-----BEGIN PUBLIC KEY...END PUBLIC KEY-----</publicKeyPem>
      </authentication>
  </authentications>
  <services>
      <service type="VerifiableCredentialService" href="https://example.com/vc/" />
  </services>
</diddocument>
```

PDF

- Great:
  - widely used in legal and other communities
  - human-readable, printable

DID Document

DID: `did:example:123456789abcdefghi`

Public Keys:

```
did:example:123456789abcdefghi#keys-1 (

    → TYPE:        Ed25519VerificationKey2018
    → CONTROLLER:  did:example:123456789abcdefghi

    → H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV
```

Authentication:

```
did:example:123456789abcdefghi#keys-1
```

Services:

```
did:example:123456789abcdefghi

    → TYPE:        Ed25519VerificationKey2018

    → https://example.com/vc/
```

Dependency: Purpose of the DID document

What's the purpose of the DID document?

- DID document describes the subject ➜ JSON-LD preference?
- DID document contains metadata for interacting with subject ➜ JSON preference?

DIDs and the Web?

- DID documents are resources on the Web ➜ JSON-LD preference?
- DID documents are more like DNS records ➜ JSON preference?

Dependency: DID methods

`did:ipid` prefers (requires?) IPLD-COSE

`did:v1` prefers (requires?) JSON-LD

`did:key` may prefer (?) JSON

`did:peer` needs some LD-like features (e.g. IDs for objects)

Many DID methods don't have a preference...

Dependency: Other technical topics

- Public key formats:
  We just added JWK support (great !).
  But do you really want a JWK public key inside an XML or CBOR DID document?

- Fragments in DID URLs:
  Dereferencing fragments is defined by the MIME type.

```
did:example:123456789abcdefghi#keys-1
```

Discussion

- Interop, extensibility, resolver behavior

Different Encodings: Model Incompatibilities (Manu, 30 min)

A session discussing the possibility of compatibility issues between different data model encodings/syntaxes.

Potential Data Model Incompatibilities

- Primitives
- Structure
- Canonical Form
- Extensibility

Data Model Primitive Incompatibilities

- CBOR supports integers as keys
  - JSON and JSON-LD do not
- CBOR supports Bignums
  - JSON does not
- CBOR and JSON-LD support byte strings
  - JSON does not
- CBOR and JSON-LD support ordered lists and unordered sets
  - JSON does not
- CBOR and JSON-LD support value types (tagged values)
  - JSON does not
- JSON Numbers are infinite, and 32-bit, or 53-bit, or 64-bit, whatever!
- You get the idea... we will need to find the intersection of Primitives between all encodings

Data Model Structure

- Flat structure
  - Flat name-value pairs
  - JSON, JSON-LD, and CBOR can do this, but limited, and no one is asking for it
- Tree Structure
  - JSON, JSON-LD, and CBOR can do this
  - Does nesting imply something? Example: association
  - Do certain names do something special? Example: id or type
  - Can you have arbitrary references to identifiers? If so, you have a graph.
- Graph structure
  - JSON-LD does this natively
  - JSON and CBOR can do this with extra rules (identifiers and references)
  - DIDDocument.publicKeys is a graph

Data Model Canonical Form Incompatibilities

- CBOR does not have a canonical form
  - One has to be defined by the protocol
  - Non-trivial complexities: IEEE representations, assumed architecture bit width
- JSON does not have a canonical form
  - JSON Canonicalization Scheme (JCS) could be used, but has corner cases
  - Lacks support for global semantics (requirement for some systems)
- JSON-LD does have a canonical form
  - RDF Dataset Canonical Form
  - Works across syntaxes (JSON, CBOR, XML, etc.)
  - Enables one digital signature to work across multiple syntaxes

Data Model Extensibility Incompatibilities

- If we use **@context** from JSON-LD
  - If JSON doesn't, then extending the data model leads to different interpretations
- If JSON-only uses **registry**
  - Does JSON-LD have to support the registry and @context?
  - Do we error on non-registry properties?
  - How does everyone stay up to date with the latest registry?
  - Does being out of date with registry create digital signature attack vectors? Hint: It does.
- If JSON-only uses **method specific bucket**
  - Does JSON-LD have to support the method specific bucket?
  - How do we update non-method specific things at the base-level of the DID Document?

Discussion

- Add slide: What happens when you have multiple signatures?

Abstract Data Modeling Options (Dan Burnett, 30 min)

Abstract Data Model

- Which are we modeling?
  - Data processing (processing of a DID Doc)
    - Processing may be done in various computer languages, and we must remain agnostic. But some format requirements help enforce business rules
  - Data storage (storage of a DID Doc)
    - Storage will likely be in databases, but again we must remain agnostic here. (Database) schema definitions help enforce business rules
  - Data exchange/transmission (transmission of a DID Doc)
    - Format for transmission need not be the same as for processing or storage
    - As with processing and storage, our model needs to remain agnostic with respect to the precise format used for transmission
    - Some ADMs help with structure that enforces business rules

Abstract Data Model

- Which are we modeling?
  - Data processing (processing of a DID Doc)
    - Processing may be done in various computer languages, and we must remain agnostic. But some format requirements help enforce business rules
  - Data storage (storage of a DID Doc)
    - Storage will likely be in databases, but again we must remain agnostic here. (Database) schema definitions help enforce business rules
  - Data exchange/transmission (transmission of a DID Doc)
    - Format for transmission need not be the same as for processing or storage
    - As with processing and storage, our model needs to remain agnostic with respect to the precise format used for transmission
    - Some ADMs help with structure that enforces business rules
- Suggest that transmission/exchange format is primarily what we are modeling

Why do people use JSON today?

- Remember RPC?
- Remember SOA and XHR/AJAX?
- JSON is an alternative to XML for transmission
- Convenient for JavaScript coders, more human-readable than XML and definitely more compact
- JSON is effectively a **wire** format, for message transmission/exchange

Alternatives to JSON for serialization of data for message parsing

- CSV
- XML
- YAML
- CBOR

Why pick one when we are defining a conceptual data format that should be usable in any of these formats? Can help to capture implied ‘business rules’ if we have them.

Which level are we defining at?

From the beginning, spec intended to define an abstract (exchange) data model

- Could define at different levels
- Conceptual, Structural (schema), ~~wire~~

Conceptual

- Describes semantic roles, relationships without getting much into syntax
- Graphical representations available that simplify overview
  - Easier for non-techies to understand
- Examples
  - Object role model (ORM, very high level)
  - Entity relationship model (EXPRESS, UML, IDEF1X)

Structural

- Presents roles/relationships as (sometimes ordered) fields, often with syntactic requirements
- Easier to map to wire/processing/storage formats
- Varying levels of support for business/algorithmic rules
- Varying support for (out of scope) method/procedure ties, possibly helpful for API/protocol definition later
- Some examples
  - EXPRESS
  - UML
  - IDEF1X
  - XML Schema
  - Protobuf
  - Schema.org

Suggestion

We should at least do a structural format, ideally also a format with a graphical representation for ease of explanation

Pros/Cons of some example structural formats

EXPRESS

- ISO 10303-11
- For exchange of product data
- Text format
  - Schema with structural and algorithmic rules
  - For formal verification and input to tools such as SDAI
- Graphical format
  - EXPRESS-G designed for human understanding

UML

- Most well-known data modeling language; for modeling business processes, use cases, and software objects
- Diagrams at the core, so graphical by default
- Not only structural diagrams, but behavioral and interaction as well, so models easy for implementers to extend beyond just structure

Protobuf

- Created by Google
- Designed to be processing-language agnostic, with libraries for most programming languages
- Enforces structural constraints without custom code
- Numbered fields simplify backwards compatibility
- Not as readable as JSON

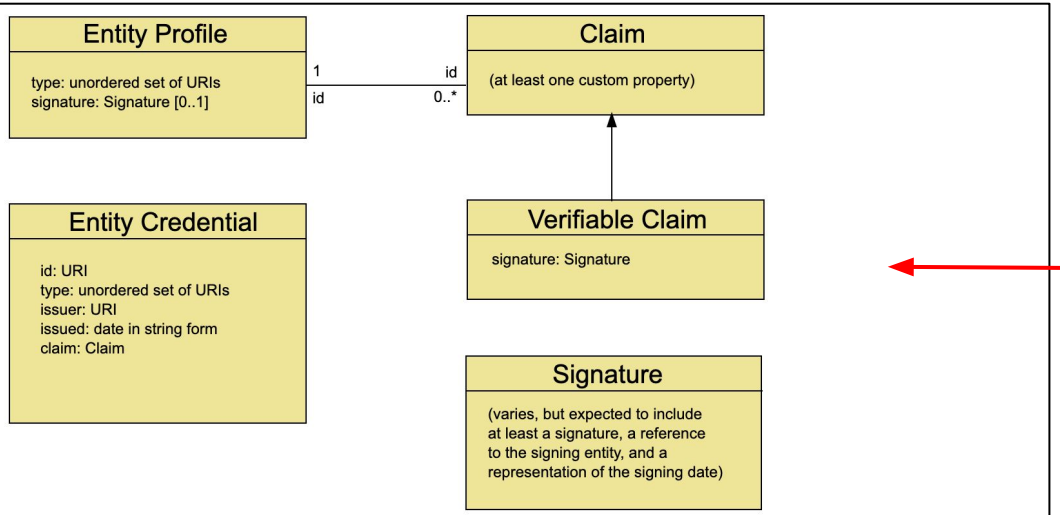
IDEF1X

- Created by USAF for their CAM
- Uses 3 different schemas to ensure proper modeling at each step

Schema.org

- Vocabularies and schema, originally for web data
- JSON-LD format supported natively
- Data model derived from RDF Schema

The original Verifiable Credentials Data Model



UML

3.1 General Characteristics

Both the Entity Profile Model and Entity Credential Model consist of a collection of name-value pairs which will be referred to as ***properties*** in this document. The following subsections describe the required and optional properties for both. The link between the two is in the *id* property. The Entity Profile Model defines a subject identifier in the *id* property, while the claims section of the Entity Credential Model uses the *id* property to refer to that subject identifier.

This document purposely defines the data model without using a concrete syntax such as WebIDL, JSON, or JSON-LD to avoid implying a bias towards any particular one syntax. Section 4. Syntaxes defines how the data model is to be expressed in those representation languages.

Discussion

Lunch (12:30 - 13:30)

Self-Sovereign Identity

Why Are We Here?

Christopher Allen *— Decentralized Identity & Blockchain Architect,*
Co-Chair W3C Credentials Community Group

Presentation

Self-Sovereign Identity: Why We Are Here

Full slide deck at:

https://docs.google.com/presentation/d/1Rtx30fB-U8MStlMrDlc-FneCrr8mel421YNbAdofDwM/edit?usp=sharing

DID Doc Extensibility via Registries (Mike Jones, 30 min)

What do registries accomplish?

- Prevent name collisions
- Publish authoritative links to where to find definitions of registered names
- Enables programmers to have identifiers used have the same meanings

Example Registries used with JSON

- IANA JSON Web Token Claims
  - https://www.iana.org/assignments/jwt/jwt.xhtml
- IANA JSON Web Signature and Encryption Algorithms
  - https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms

How do you add to a registry?

- Publish a specification defining the new entry
- The specification need not be published by the registry authority
    - For instance, a W3C specification is registering secp256k1 with IANA

Observation in DID Spec

- DID Spec prohibits redefining member names defined by spec
- So no dynamic conflict avoidance mechanisms/namespacing needed
- Registries can be used to avoid future name conflicts

DID Doc Extensibility via JSON-LD (Manu, 30 min)

A session covering the ways that
JSON-LD could enable extensibility of DID Documents.

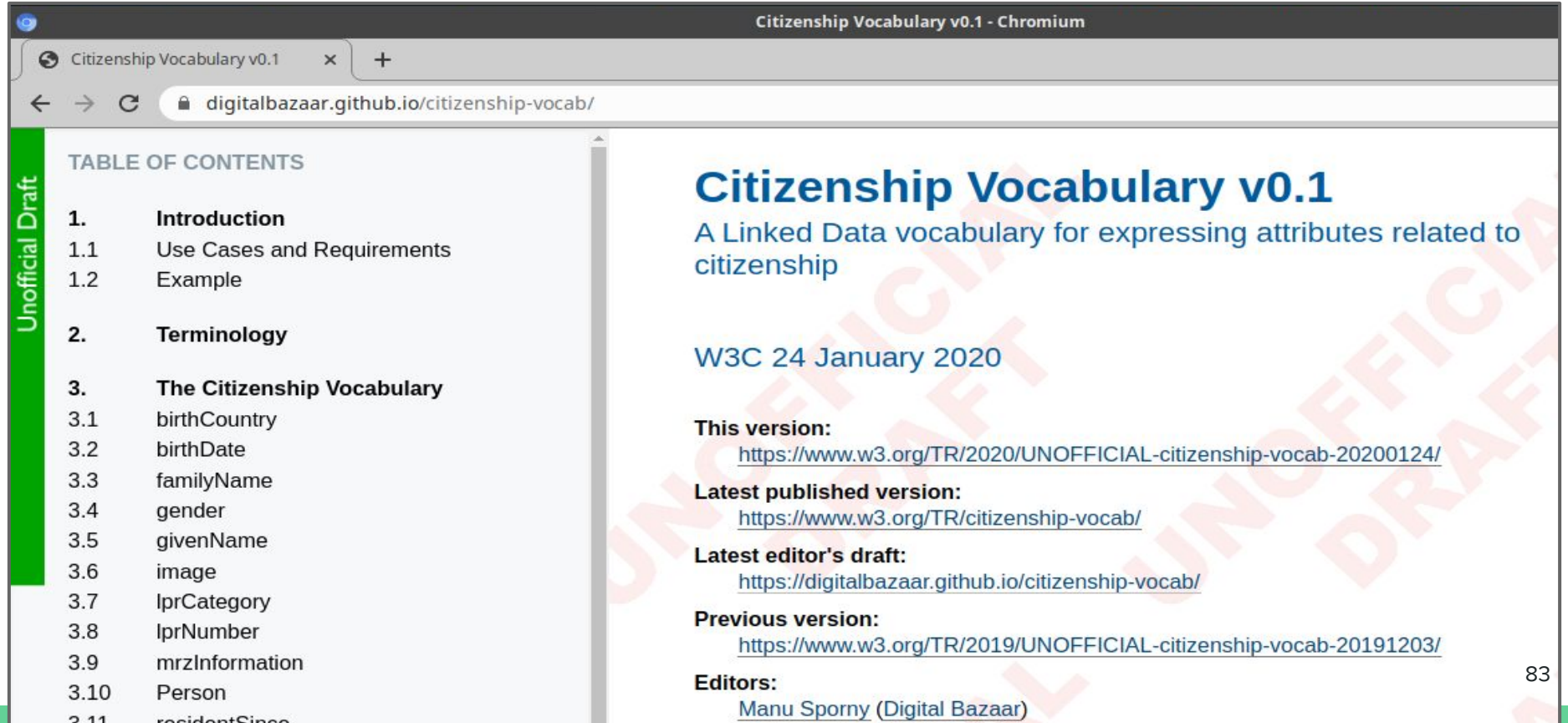
Why extend a DID Document?

- DID method implementers want to add method-specific properties
  - Example: did:web => didDocumentCreated, didDocumentUpdated
- Application developers want to add new service types
  - Example: MicroblogService, PrivateMessagingService, ResumeService
- Application developers want to add new cryptographic mechanisms
  - Example: ethereumAddress, post-quantum eXtended Merkle Signature Scheme
- Merge data with Verifiable Credentials
  - Combine DID Document + 15 Verifiable Credentials => process holistic view of subject

Extending a DID Document with JSON-LD

1. Define a vocabulary
2. Create a JSON-LD Context
3. Append it to **@context** property

Define a Vocabulary (~2 hours)



Create a JSON-LD Context (~1 hour)

```
{
  "@context": {
    "@version": 1.1,
    "@protected": true,

    "myDidFeature": "https://example.com/vocab#myDidFeature"
}
```

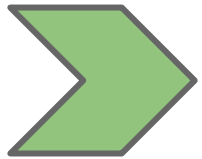
Append it to @context property (~30 seconds)

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://example.com/my-did-method/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [ …  ],
  "service": [ … ],
  "myDidFeature": "Do something amazing!"
}
```

What about cost to JSON-only developers?

1. Update your JSON Schema, or
2. Update validation function in your bespoke library

```
const jsonSchema = {
 type: 'object',
 title: 'DID Document',
 properties: {
   '@context': {
      type: 'array',
      minItems: 1,
      items: [
        'https://www.w3.org/ns/did/v1'
      ]
```



```
const jsonSchema = {
 type: 'object',
 title: 'DID Document',
 properties: {
   '@context': {
      type: 'array',
      minItems: 2,
      items: [
        'https://www.w3.org/ns/did/v1',
        'https://example.com/did-method/v1'
      ]
```

Drawbacks of JSON-LD Extensibility

1. Must restrict general JSON-LD processing to ensure security.
    a. Counter argument: Secure systems always require sensible restrictions.
2. Decentralized extensibility can reduce peer reviews; messier systems.
    a. Counter argument: It can also increase decentralized innovation - no gatekeeping.
3. "Too complex for my use case!"
    a. Counter argument: There are other use cases that are not yours that benefit.
    b. Counter argument: What someone else has said about your favorite programming language.

Benefits of JSON-LD Extensibility

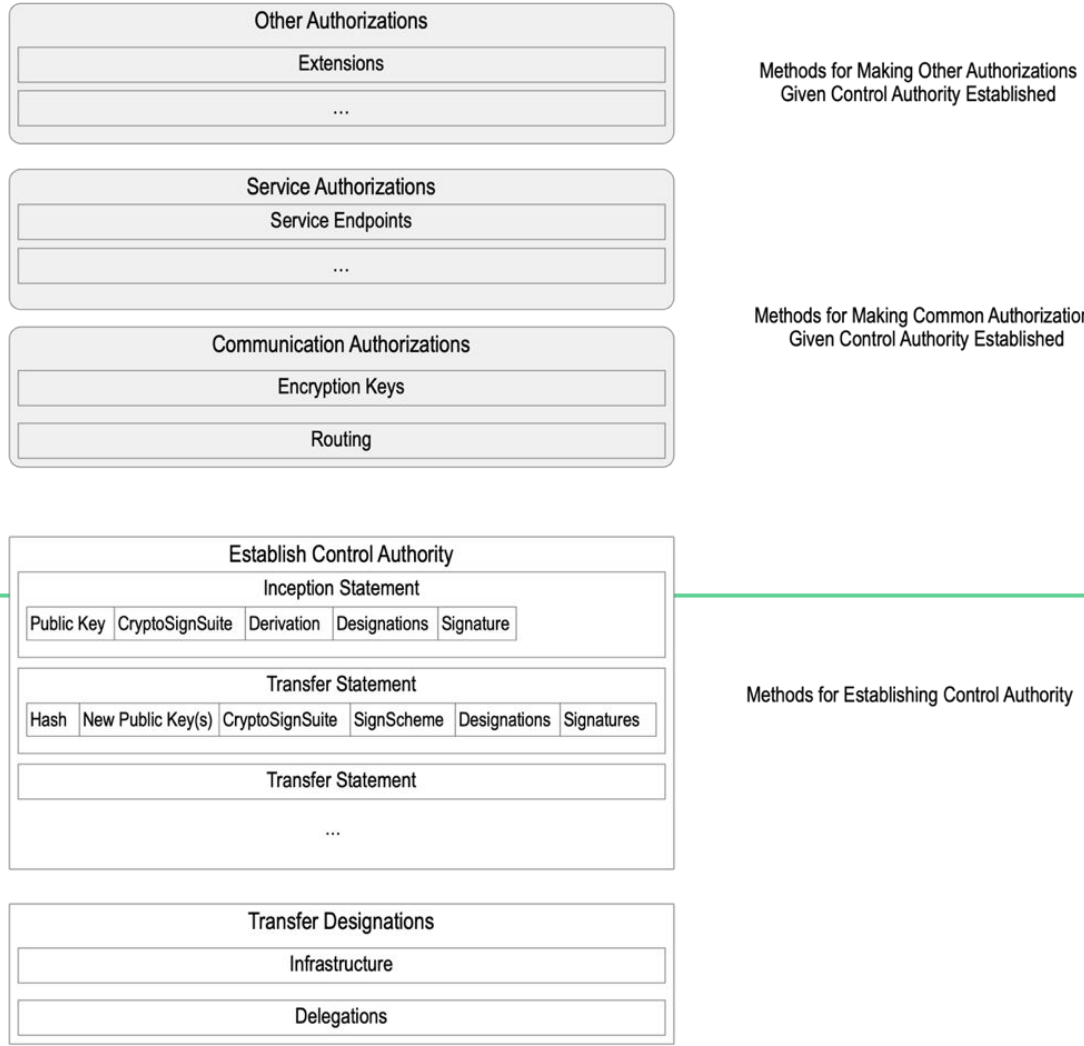
1. No need to consult with a Working Group.
    a. Counter argument: You ***should*** consult with a Working Group to avoid bad work.
2. No need to get approval in a registry.
    a. Counter argument: Someone else needs to check your work to make sure you don't conflict.
3. Global semantics - no property conflicts - guaranteed.
    a. Counter argument: We don't need the complexity, DID Documents should be simple.
4. Compatible with Verifiable Credentials Data Model
    a. Counter argument: DID Documents don't need to be data model compatible with VCs, they are different things.

Discussion

- Commentary...

Afternoon break (14:30 - 14:45)

DID Doc Extensibility: Discussion (Chairs, 30 min)


Other Authorizations
Extensions
...
Service Authorizations
Service Endpoints
...
Communication Authorizations
Encryption Keys
Routing
Establish Control Authority
Inception Statement
Public Key
CryptoSignSuite
Derivation
Designations
Signature
Transfer Statement
Hash
New Public Key(s)
CryptoSignSuite
SignScheme
Designations
Signatures
Transfer Statement
...
Transfer Designations
Infrastructure
Delegations
Methods for Making Other Authorizations
Given Control Authority Established
Methods for Making Common Authorizations
Given Control Authority Established
Methods for Establishing Control Authority

Orig view: DID doc is just a printout of some reso. info

From RFC 3986, the URI spec:

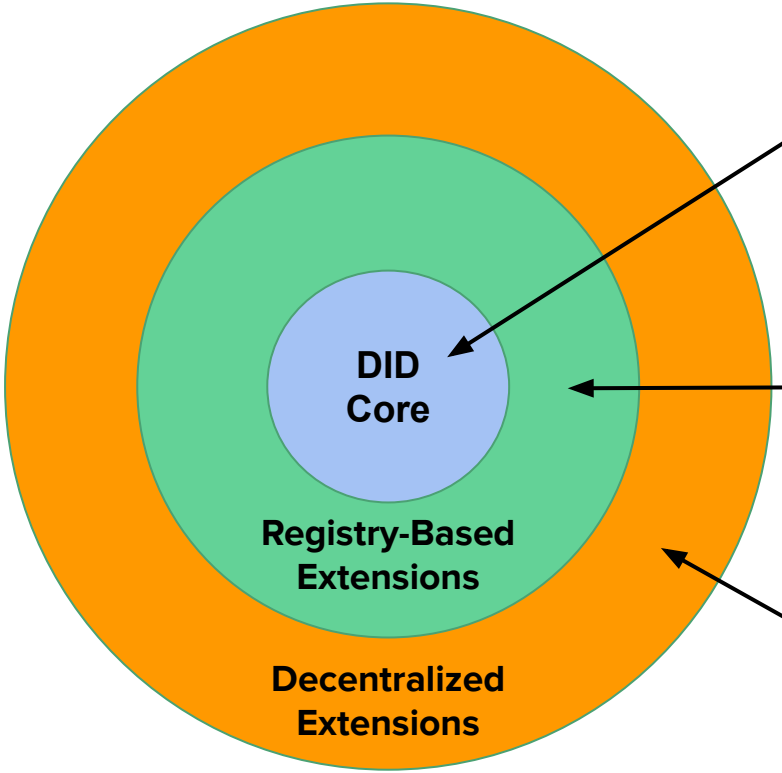
. . . A resource is not necessarily accessible via the Internet; e.g., human beings, corporations, and bound books in a library can also be resources. Likewise, abstract concepts can be resources, such as the operators and operands of a mathematical equation, the types of a relationship (e.g., "parent" or "employee"), or numeric values (e.g., zero, one, and infinity). . . .

. . . An identifier embodies the information required to distinguish what is being identified from all other things within its scope of identification. Our use of the terms "identify" and "identifying" refer to this purpose of distinguishing one resource from all other resources, regardless of how that purpose is accomplished (e.g., by name, address, or context). These terms should not be mistaken as an assumption that an identifier defines or embodies the identity of what is referenced, though that may be the case for some identifiers. Nor should it be assumed that a system using URIs will access the resource identified: in many cases, URIs are used to denote resources without any intention that they be accessed. . . .

1.2.2. Separating Identification from Interaction

> **A common misunderstanding of URIs is that they are only used to refer to accessible resources. The URI itself only provides identification; access to the resource is neither guaranteed nor implied by the presence of a URI. Instead, any operation associated with a URI reference is defined by the protocol element, data format attribute, or natural language text in which it appears.**
>
> **Given a URI, a system may attempt to perform a variety of operations on the resource, as might be characterized by words such as "access", "update", "replace", or "find attributes". Such operations are defined by the protocols that make use of URIs, not by this specification. However, we do use a few general terms for describing common operations on URIs. URI "resolution" is the process of determining an access mechanism and the appropriate parameters necessary to dereference a URI; this resolution may require several iterations. To use that access mechanism to perform an action on the URI's resource is to "dereference" the URI.**

Three-Circle Model of DID Document Extensibility



Universally-agreed semantics for all DID docs—established by DID Core spec

Semantics established by a specified centralized registry mechanism backed by specs

Semantics established by collision-resistant namespaces and independent specs

Metadata (Ganesh Annan, 60 min)

Background

Does **metadata** about a **DID Document** (such as when it was **created**, **updated**, or **who it was signed by**) belong in that DID Document?

https://github.com/w3c/did-core/issues/65

Background - Metadata Definition

Does **metadata** about a **DID Document** (such as when it was **created**, **updated**, or **who it was signed by**) belong in that DID Document?

Metadata (def.)

a set of data that describes and gives information about other data.[1]

Background - DID Document Definition

Does **metadata** about a **DID Document** (such as when it was **created**, **updated**, or **who it was signed by**) belong in that DID Document?

DID Document (def.)

A set of data describing the DID subject, including mechanisms, such as public keys and pseudonymous biometrics, that the DID subject can use to authenticate itself and prove their association with the DID. A DID document might also contain other attributes or claims describing the subject. These documents are graph-based data structures that are typically expressed using [JSON-LD], but can be expressed using other compatible graph-based data formats. [2]

Background - DID Document Explained

Does **metadata** about a **DID Document** (such as when it was **created**, **updated**, or **who it was signed by**) belong in that DID Document?

DID Document

- A DID document contains information associated with the DID, such as ways to cryptographically authenticate the entity in control of the DID, as well as services that can be used to interact with the entity. [3]

Background - DID Document Explained

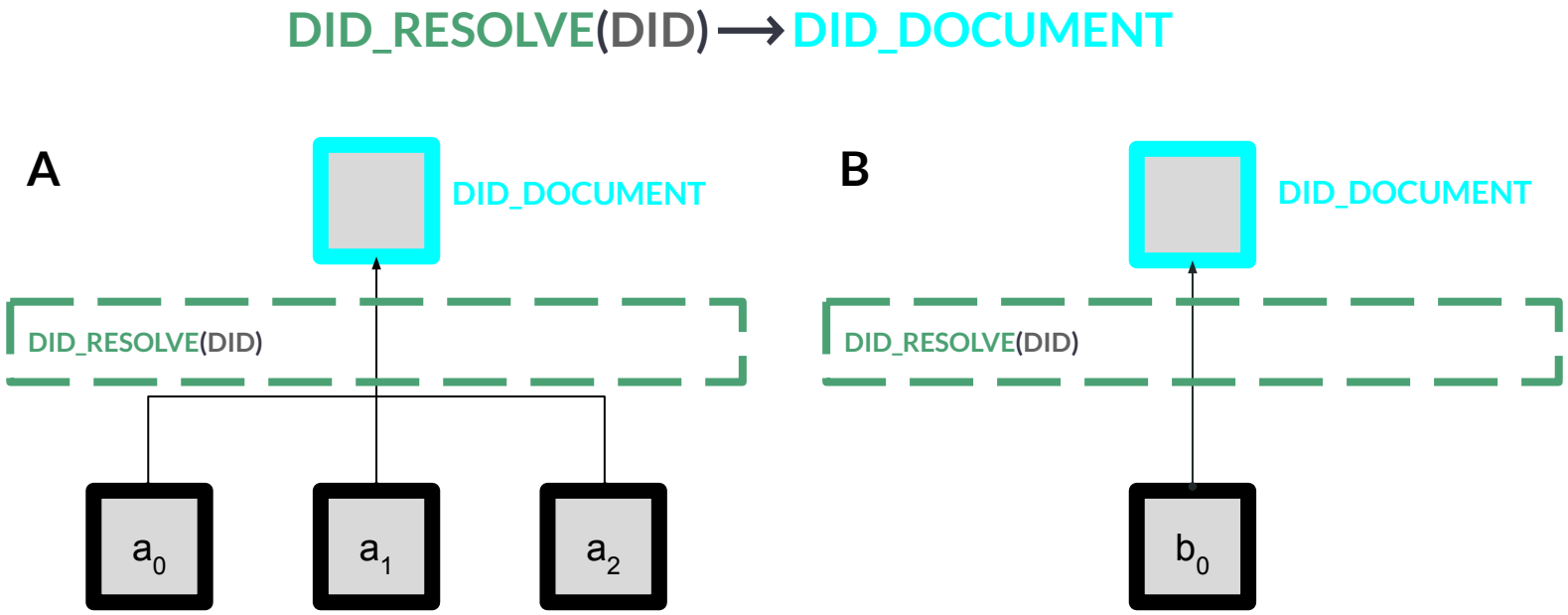
DID_RESOLVE(DID) ⟶ DID_DOCUMENT

Does **metadata** about a **DID Document** (such as when it was **created**, **updated**, or **who it was signed by)** belong in that DID Document?

DID Document

- A DID Document is the **output** of a DID Resolution process
  - This is intentionally abstract as it gives full creative license to DID method implementers to determine how to resolve a DID to a DID Document

Background - DID Document Illustrated



DID Document Created?

Does **metadata** about a **DID Document** (such as when it was **created**, **updated**, or **who it was signed by)** belong in that DID Document?

- It presumes a "DID Document is a file on a file system" point of view, which it MAY NOT BE. That is an implementation detail.
  - A DID Document is not a "file" that has "metadata" like on a file system. It cannot be assumed to work this way.
- Some DID methods generate DID Documents in a totally ephemeral way.

DID Document Created.

What makes sense for "when a DID Document was created" is that it was created DURING a resolution process, i.e., someone passed a DID to a resolver and the DID Document was created as the result of that call.

1. The mental model for this may be equivalent to the creation of an HTTP Response
2. This could be modeled similar to the `date` header on an HTTP Response Header [RFC 7231]
    a. The "Date" header field represents the date and time at which the message was originated, having the same semantics as the Origination Date Field (orig-date) defined in Section 3.6.1 of [RFC5322]. The field value is an HTTP-date, as defined in Section 7.1.1.1.

Identify Core Problem

What is in the DID Document?

Information related to the DID subject. So, if your use case has to do with expressing information related to the DID subject, then it makes sense to express it there. Otherwise, it does not.

Discussion

What information do we want to express?

- What information as it relates to the DID subject do we want to place in the DID Doc?
- Do we want to express that the DID subject has a "DID registration" using DID method X that happened at time t_0?
- What are the use cases driving the need for metadata?

Open (45 min)

Graph of the Meaning of “DID Subject”

Anyone can say anything about anything—and prove they said it

Visual model of DIDs

Layer	Examples
Application	Selective disclosure, music app, rideshare service, extensions, etc.
Implementation	DIF Hubs, Indy Agents, uPort app, etc.
Payload	JSON-LD, JWT, CWT
Encoding	ProtoBuf, Cap'n Proto, MessagePack, JSON, CBOR, etc.
Encryption	Ciphersuites, JWE, etc.
DID AuthN	Key ownership, verification, challenge/response, etc.
Transport	QR Code, HTTP, BLE, NFC, FTP, SMTP, etc.
DID Resolution	DID -> DID Doc / service and key resolution
DID Operations	CRUD support for a DID Doc
Storage	Optional, separate storage of DID metadata, e.g., IPFS
Anchor	Bitcoin, Ethereum, Veres.One, Sovrin, etc.

SSI Stack diagram  crowdsourced at IIW__?
https://medium.com/decentralized-identity/the-self-sovereign-identity-stack-8a2cc95f2d45

Decentralized Identifier WG Face-to-Face meeting

Day 2: January 30, 2020
Chairs: Brent Zundel, Dan Burnett
Location: Microsoft Schiphol

Today’s agenda

8:00	Breakfast	
8:30	Review and Agenda	Chairs
9:00	Levels of Interoperability	Mike/Manu
9:30	Interoperability (discussion, decision)	Chairs
10:00	Extensibility and Interoperability	Manu
10:30	Break	
10:45	JSON format (PR processing)	Drummond
11:45	Spec Next Steps	Chairs
12:30	Lunch (ZKP presentation)	
13:30	Prep for boat tour	
15:00	Amsterdam Canal Boat Tour and Dinner	

Dinner Tonight

Expected Count: roughly 20

Dinner Proposals:

Levels of Interoperability
(Mike Jones and Manu Sporny, 30 min)

Overview of Kinds of Interoperability

1. No interop
    a. Why are we even here!?
2. Interop on Data Model
    a. Some abstract data modelling mechanism that is agreed to
3. Interop on Data Model + Basic Syntax
    a. A basic subset of JSON / JSON-LD / CBOR that is agreed to
4. Interop on Extension Mechanism
    a. Registries, Syntax-based extensibility mechanism, etc.
5. Interop on Canonical Form
    a. Is there a canonical form for the data model?
6. Interop on Cryptography
    a. Can you interoperate when you produce and verify proofs?

Overview of Kinds of Interoperability (cont.)

- Interop on Behavior
  - Are there functions/methods/actions/APIs that are expected to do something in a specific way or transform data in certain ways
- Interop on User Experience
  - Does the user have a consistent experience when using the technology?
- Interop on Transport / Protocol Messages
  - How do you get the data model from one place to another
- DID Method interop
- Protocol Stack from a few IIWs ago - https://medium.com/decentralized-identity/the-self-sovereign-identity-stack-8a2cc95f2d45

Q&A / Crowd Contributions

Kaliya referred us to a diagram Crowdsourced at IIW27-->
And written up by Oliver Terbu

Layer	Examples
Application	Selective disclosure, music app, rideshare service, extensions, etc.
Implementation	DIF Hubs, Indy Agents, uPort app, etc.
Payload	JSON-LD, JWT, CWT
Encoding	ProtoBuf, Cap'n Proto, MessagePack, JSON, CBOR, etc.
Encryption	Ciphersuites, JWE, etc.
DID AuthN	Key ownership, verification, challenge/response, etc.
Transport	QR Code, HTTP, BLE, NFC, FTP, SMTP, etc.
DID Resolution	DID -> DID Doc / service and key resolution
DID Operations	CRUD support for a DID Doc
Storage	Optional, separate storage of DID metadata, e.g., IPFS
Anchor	Bitcoin, Ethereum, Veres.One, Sovrin, etc.

Practical means of promoting interoperability

- Have developers try their implementations with one another
  - See what happens with each other
- Create an interop test suite covering important aspects of specifications
  - Run implementations against test suite and see what happens
- Create a certification program that tests implementations
  - Test suite and legal process enforcing defined set of features required for certification

Virtuous cycle between interop testing and spec work

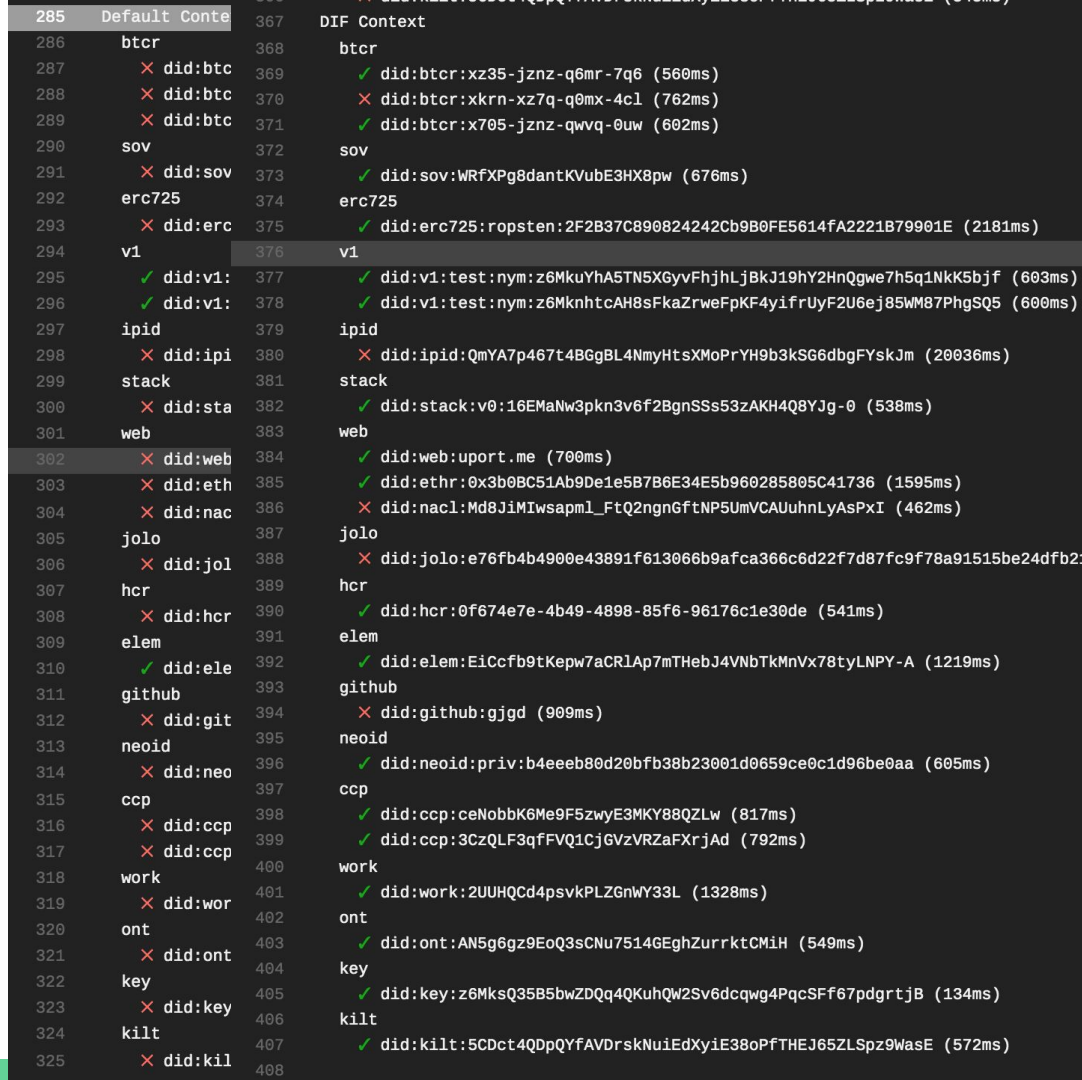
- Interop testing can expose implementation bugs
  - Fix bugs and test again
  - Improves implementations
- Interop testing can expose specification bugs and ambiguities
  - For instance, exposing where different developers interpreted the spec differently
  - Fix spec bugs and ambiguities and test again
  - Improves specifications
- Iterate, improving implementations and specifications until both are solid
  - Proven means to achieve interoperable implementations using unambiguous specifications

What to test to improve interoperability

- Test that the MUSTs and SHOULDs are followed by implementations
- Test positive cases
  - That implementations emit and accept correct messages
  - For instance, that signatures with a key pair can be produced and verified
- Test negative cases
  - That implementations reject incorrect messages
  - For instance, that incorrect signatures are rejected
- Test uses of extensibility points
  - That implementations do not reject messages using extensions that they don’t understand
- Test invariants
  - Verify that implementations progress from legal states to other legal states as specified

Q&A / Crowd Contributions

Manu mentioned that DIF is open to some kinds of test-suite hosting and/or certification. Here's a screengrab Orie shared on DIF Slack this week of a preliminary LD-proof/sig test script he ran, for reference:



Interoperability: Discussion & Decision (Chairs, 30 min)

Extensibility and Interoperability (Manu, 30 min)

Proposal #1

1. The DID Core specification will define an abstract data model that can be cleanly represented in at least JSON, JSON-LD, and CBOR. There will also be a graphical depiction of the abstract data model. There must be lossless conversion between multiple syntaxes (modulo signatures and verification).
2. In general, the registry mechanism is the one that will be used for globally interoperable extensions.
3. The governance of the registry mechanism will be defined by the W3C DID Working Group.
4. Extension authors must provide references to specifications for new entries and a valid JSON-LD Context to be associated with each entry to ensure lossless conversion between serializations for both producers and consumers. This is partly being done to ensure semantic interoperability.

Proposal #2

1. The DID Core specification will define an abstract data model that can be cleanly represented in at least JSON, JSON-LD, and CBOR. There will also be a graphical depiction of the abstract data model. There must be lossless conversion between multiple syntaxes (modulo signatures and verification).
2. The extension mechanism will be @context and DID Document publishers MUST use it to express the base context and do extensions.
3. JSON-only processors only need to check @context (basic string compare against expected contexts) when programming against the specification or any extension.
4. There is no need for a registry or a maintenance group or expert approval.

Morning break (10:30 - 10:45)

JSON Format (PR Processing) (Drummond, 60 min)

Overview of “JSON-Only” PRs

1. #145: removed "context" from requirements
    - Justin, 2019-12-07
2. #146: Simplifies spec by using pure JSON instead of JSON-LD
    - Mike Lodder, 2019-12-20

#145—removed "context" from requirements

In order to facilitate the description of DID documents as pure JSON structures, this pull request removes requirements to use `@context` in the spec for all processors. Note that `@context` is still allowed to be used by JSON-LD providers, it's now just classified as an extension. The extensive examples using JSON-LD are left in as they are still valid, but additional plain-JSON extensions examples still need to be added.

Additionally, a formal requirement for how to process fields you don't know (MUST ignore, MAY ignore, etc) still needs to be agreed upon and added, I just wasn't sure quite where to put it so it's in the extensibility section.

#145 Example

6. DID Documents

A DID points to a DID document. DID documents are the serialization of the § 3. Data Model. The following sections define the properties in a DID document, including whether these properties are required or optional.

6.1 Contexts When two software systems need to exchange data, they need to use terminology and a protocol that both systems understand. The @context property ensures that two systems operating on the same DID document are using mutually agreed terminology. DID documents MUST include the @context property. Note : The JSON-LD Context More information about the JSON-LD Context in general can be found in the [JSON-LD] specification. @context The value of the @context property MUST be one or more URIs , where the value of the first URI is https://www.w3.org/ns/did/v1 . If more than one URI is provided, the URIs MUST be interpreted as an ordered set. It is RECOMMENDED that dereferencing the URIs results in a document containing machine-readable information about the context. Example: Example 10 { "@context": "https://www.w3.org/ns/did/v1" } DID method specifications MAY define their own JSON-LD contexts. However it is NOT RECOMMENDED to define a new context unless necessary to properly impleme

#146—Simplifies spec by using pure JSON instead of JSON-LD

- Removes sections of the spec that mention JSON-LD
- Removes @context from examples
- TallTed suggested in a comment that the deleted JSON-LD content be moved into a separate section or Appendix that defines a JSON-LD representation

#146 Example edit

EXAMPLE 2 : Minimal self-managed DID document

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "service": [{
    // used to retrieve Verifiable Credentials associated with the DID
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

Suggested requirements for a JSON-Only representation

#1: No confusion

- A JSON-only representation MUST NOT include an `@context` statement
- This is a clear and unambiguous signal to any consumer of a DID document that JSON-LD processing should not be performed

#2: MIME Type

- A JSON-only representation SHOULD have a MIME type that indicates it is pure JSON
- Resolvers SHOULD be able to request a JSON-only representation by using this MIME type

#3: Encoding Rules

- A JSON-only representation MUST be valid JSON
- It MUST NOT redefine any property defined in the Core Data Model

#4: Fragment Processing Rules

- Fragments are processed by matching them to the value of the `id` property defined in the Core Data Model
- Matching of both relative and absolute URIs to the value of the `id` property MUST be supported
- Absolute values MUST begin with `did:`
- Relative values MUST begin with `#`

#5: Extensibility

- Extensions of the JSON-only format MUST follow the extensibility requirements defined in the DID Core spec

Spec Next Steps (Chairs, 45 min)

Lunch (12:30 - 13:30)
Presentation on ZKPs (Brent, 45 mins)

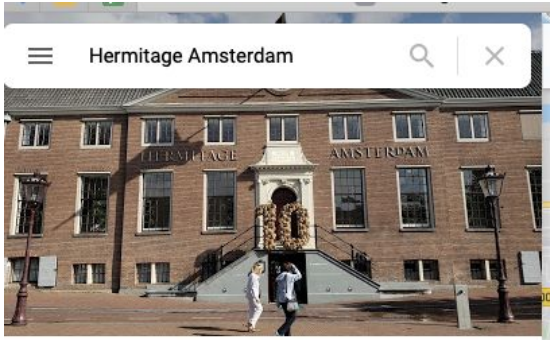
https://docs.google.com/presentation/d/1hGEpWIpl9hp8QoTIXjlozY7Gzy6zXY9IAL2x6U1b7Fk/edit#slide=id.g409d465df3_2_25

Boat Tour Logistics (Ivan)

Boat tour

- Meeting time: 3:30 pm (16:00)
- Meeting location: Amstel 51F, by the Hermitage Museum on the Amstel river
  - See https://tinyurl.com/vgzvdkd
- Tour timing: 90 minutes, from 16:00-17:30
- Cost: EU 25
- Boat return address: Stadthouderkade, somewhere around Leidseplein
- There are some drinks (beer, juice) for everyone
- If I am knocked over by a bus:-)
  - Company name BoatAmsterdam,
  - Booking number 368407

Hermitage Amsterdam



Hermitage Amsterdam

4,4 ★★★★½ (5199)

Musée

Itinéraires | Enregistrer | À proximité | Envoyer vers votre téléphone | Partager

ACHETER DES BILLETS

Amstel 51, 1018 EJ Amsterdam

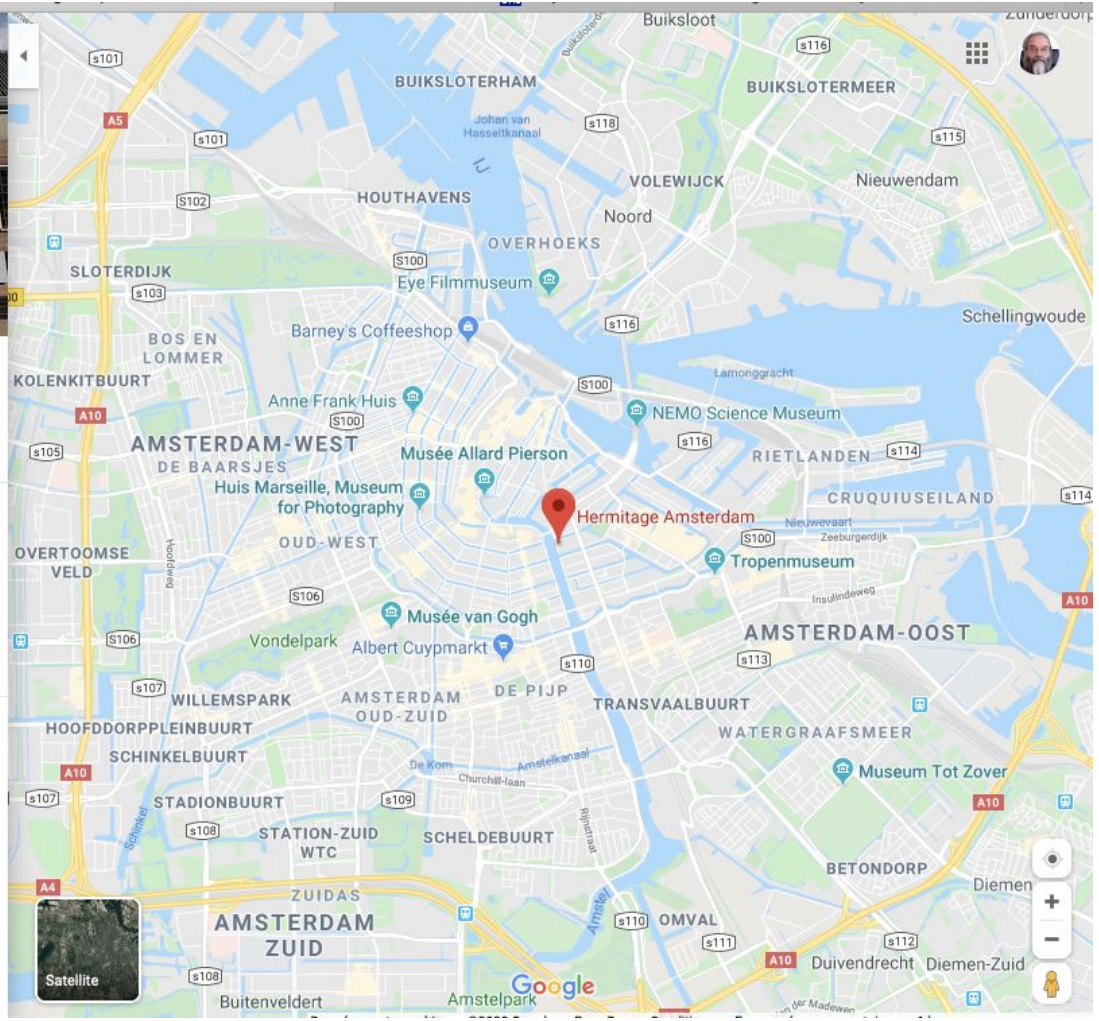
9W82+4X Amsterdam

hermitage.nl

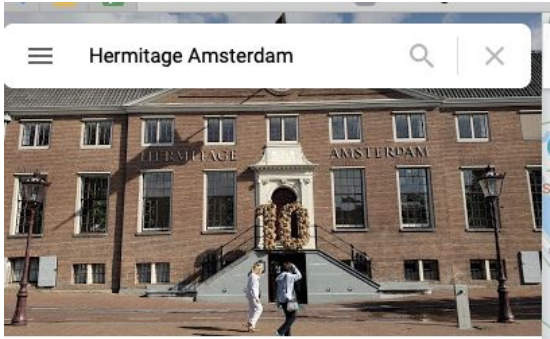
020 530 8755

Établissement ouvert : 10:00–17:00

Ajouter un libellé



Hermitage Amsterdam



Hermitage Amsterdam

4,4 ★★★★½ (5199)

Musée

Itinéraires | Enregistrer | À proximité | Envoyer vers votre téléphone | Partager

ACHETER DES BILLETS

Amstel 51, 1018 EJ Amsterdam

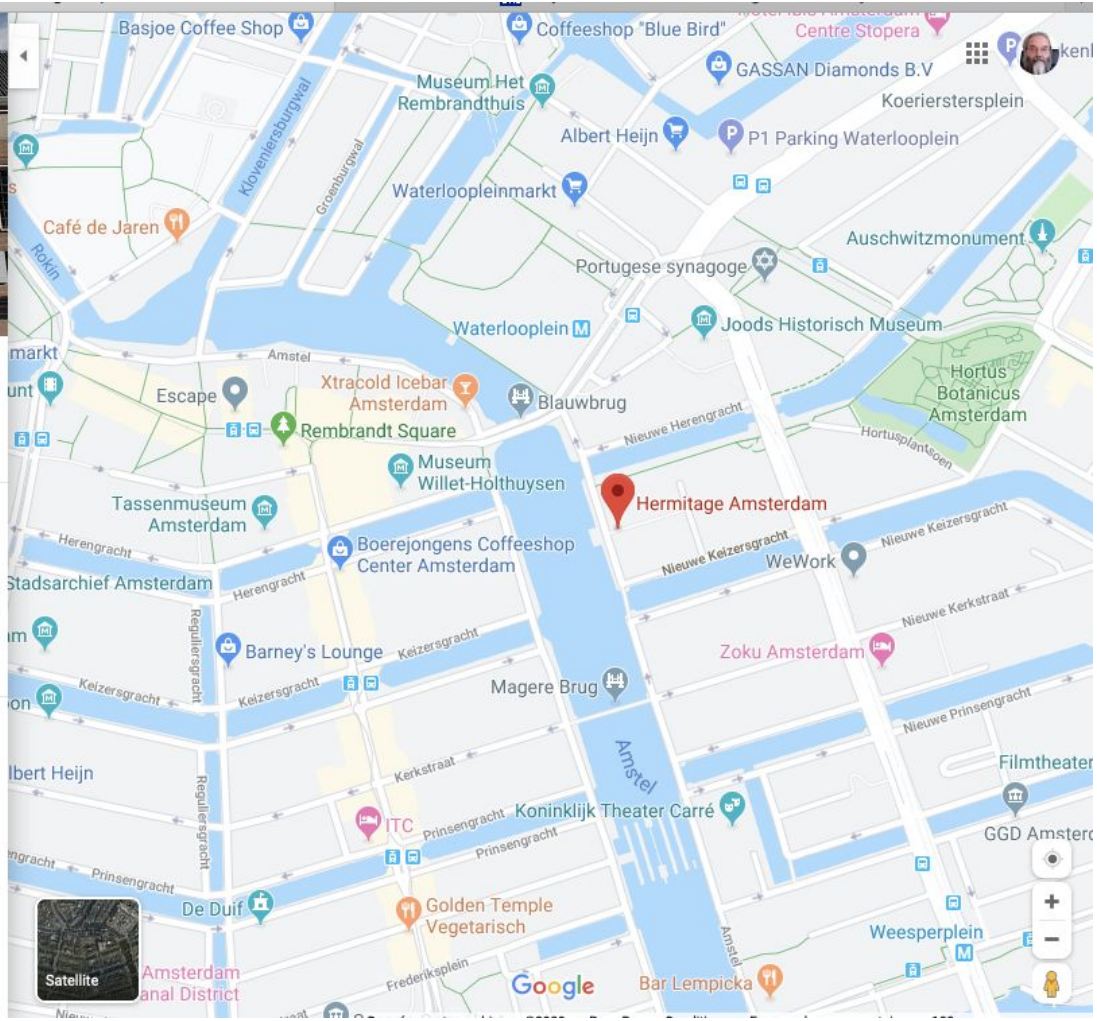
9W82+4X Amsterdam

hermitage.nl

020 530 8755

Établissement ouvert : 10:00–17:00

Ajouter un libellé



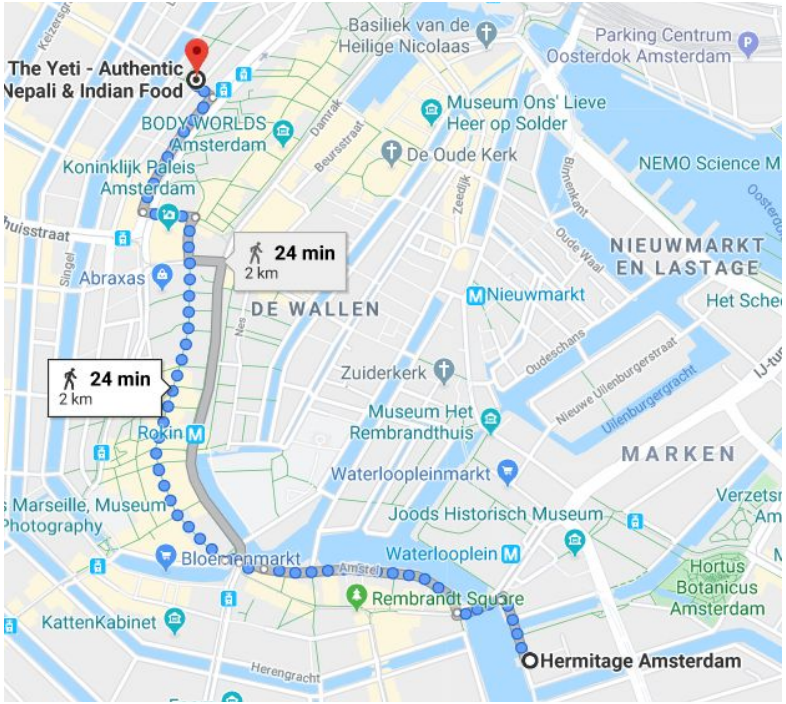
Dinner 30 Jan - 19:00

Restaurant: x

Location: Spuistraat 54D, 1012TV Amsterdam

Website: http://theyeti.nl/

Booking Status: Booking confirmed.



Decentralized Identifier WG Face-to-Face meeting

Day 3: January 31, 2020
Chairs: Brent Zundel, Dan Burnett
Location: Microsoft Schiphol

Today’s agenda

8:00	Breakfast	
8:30	Review and Agenda	Chairs
9:00	Matrix parameters & Query Parameters	Markus
10:30	Break	
10:45	Spec Structure	Drummond
11:30	Rubric	Joe Andrieu
12:30	Lunch	
13:30	Use Cases and Requirements	Joe and Phil
14:30	Break	
15:00	Topics brought up at F2F	Chairs
16:00	Overlap with DID Resolution	Chairs
16:45	Wrap up	Chairs

Dinner Tonight

Expected Count: 10-12

Dinner Proposals:

Matrix Parameters & Query Parameters (Markus, 90 min)


DID:EX:123;SERVICE=HUB/FILES/RESUME.PDF

Matrix parameters

- Actually called "DID parameters" in the spec

```
did-url  = did *( ";" param ) path-abempty [ "?" query ][ "#" fragment ]
did      = "did:" method-name ":" method-specific-id
param    = param-name [ "=" param-value ]
```

Origin:
Tim Berners-Lee
Matrix URIs - Ideas about Web Architecture
Date: December 19, 1996

How do we use DIDs?

- Discover public keys and service endpoints
- Some applications just need public keys
- Many applications just need public keys and a service endpoint

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:ex:123",
  "authentication": [{
     "id": "did:ex:123#keys-1",
     "type": "Ed25519VerificationKey2018",
     "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "service": [{
     "id": "did:ex:123",
     "type": "AgentService",
     "serviceEndpoint": "https://agency.com/myagent"
  }]
}
```

But I could have many "services"

- Services are like URLs that my DID links to.
- Not all services necessarily require a special protocol or DID-based auth/encryption.
- I can have many services/links in my DID document.

But I could have many "services"

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:ex:123",
  "service": [{
    "id": "did:ex:123#homepage",
    "serviceEndpoint": "https://alice.me/home/"
  }, {
    "id": "did:ex:123#work",
    "serviceEndpoint": "https://acmecorp.com/employees/alice-7332"
  }, {
    "id": "did:ex:123#linkedin",
    "type": "SocialNetworkService",
    "serviceEndpoint": "https://www.linkedin.com/in/alice-b26187c4/"
  }, {
    "id": "did:ex:123#message",
    "type": "ActivityPubService",
    "serviceEndpoint": "https://chaos.social/@alice01"
  }
}
```

Web Linking (RFC 8288)

- A model for the relationships between resources on the Web ("links") and the type of those relationships ("link relation types").

```
<link rel="stylesheet" href="/media/example.css">
```

- Used by IndieWeb community for service/link discovery

```
<link rel="micropub" href="https://alice.me/pub">
```

```
<link rel="profile" href="https://profiles.com/alice">
```

- Or using the HTTP "Link" header:

```
Link: <https://profiles.com/alice>; rel="profile"
```

PURLs

- From Wikipedia:
  “A Persistent URL is an address on the World Wide Web that causes a redirection to another Web resource.“
  “If a Web resource changes location (and hence URL), a PURL pointing to it can be updated.”

  `http://purl.org/some/path` → `http://example.com/another/path`
  `http://purl.org/some/path` → `http://selfhosted.me:8080/`

  (“PURLs have been criticized for their need to resolve a URL, thus tying a PURL to a network location.”)

PURLs

*“The PURL service includes a concept known as **partial redirection**.“*

`http://purl.org/some/path/and/some/more/data`

→ `http://example.com/another/path/and/some/more/data`

→ `http://selfhosted.me:8080/and/some/more/data`

*“The concept of partial redirection **allows hierarchies of Web-based resources to be addressed** via PURLs without each resource requiring its own PURL.”*

DID-based PURLs

`did:ex:123`

→ `http://example.com/another/path`

→ `http://selfhosted.me:8080`

DID-based PURLs

`did:ex:123/and/some/more/data`

→ `http://example.com/another/path/and/some/more/data`

→ `http://selfhosted.me:8080/and/some/more/data`

DID-based PURLs

`did:ex:123/and/some/more/data`

→ `http://example.com/another/path/and/some/more/data`

- BUT: DIDs don't just redirect to another URL.
  They have a list of services/links (Web Linking !).

- So we need a way to indicate the service as part of the DID URL.

`did:ex:123;profile/and/some/more/data`

→ `http://example.com/another/path/and/some/more/data`

DID-based PURLs

did:ex:123;service=files/myresume/doc?version=latest#intro

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:ex:123",
  "publicKey": [{
      "id": "did:ex:123#keys-1",
      "type": "RsaVerificationKey2018",
      "publicKeyPem": "-----BEGIN PUB...01"
  }],
  "service": [{
      "id": "did:ex:123#files",
      "serviceEndpoint":
      "https://filestore.org/user123/"
  }]
}
```

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:ex:123",
  "publicKey": [{
      "id": "did:ex:123#keys-1",
      "type": "RsaVerificationKey2018",
      "publicKeyPem": "-----BEGIN PUB...01"
  }],
  "service": [{
      "id": "did:ex:123#files",
      "serviceEndpoint":
      "https://selfhosted.me:8080/"
  }]
}
```

→ **https://filestore.org/user123/myresume/doc?version=latest#intro**

→ **https://selfhosted.me:8080/myresume/doc?version=latest#intro**

DID-based PURLs (without "brittle" path/query/fragment)

did:ex:123;service=socialnetwork

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:ex:123",
  "authentication": [{
      "id": "did:ex:123#keys-1",
      "type": "RsaVerificationKey2018",
      "publicKeyPem": "-----BEGIN PUB.\r\n"
  }],
  "service": [{
      "id": "did:ex:123#socialnetwork",
      "serviceEndpoint":
"https://socialnetwork.com/user/123/profile"
  }]
}
```

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:ex:123",
  "authentication": [{
      "id": "did:ex:123#keys-1",
      "type": "RsaVerificationKey2018",
      "publicKeyPem": "-----BEGIN PUB...--\r\n"
  }],
  "service": [{
      "id": "did:ex:123#socialnetwork",
      "serviceEndpoint":
      "https://newnetwork.com/account022/info"
  }]
}
```

→ **https://socialnetwork.com/user/123/profile**

→ **https://newnetwork.com/account022/info**

DID-based PURLs

`did:ex:123;service=files/myresume/doc?version=latest#intro`

- Combines:
  - Web Linking
  - PURLs
- Plus the other features of DIDs, i.e. decentralization, cryptographic verifiability!
- A self-sovereign indirection layer for Web addresses.

DID parameters

- Purpose is to pass parameters into the DID resolution process:

```
did:ex:123;service=files/myresume/doc?version=latest#intro
```

- Path, query, fragment:
  Purpose is to identify the resource that is being dereferenced under the URL's authority.

- DID Core spec should not define concrete paths, queries, fragments.

URNs

- For additional inspiration, see RFC 8141.
- URNs have "r-components", "q-components", "f-components".

Example:

```
urn:example:foo-bar-baz-qux?+CCResolve:cc=uk?=lat=39.56&lon=-104.85
```

More matrix parameters: version-id

`did:ex:123;version-id=4#keys-1`

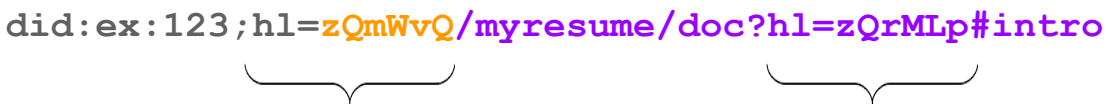
version of the DID document

`did:ex:123;version-id=4;service=files/myresume/doc?version=latest#intro`

version of the DID document

service-specific "version" parameter

More matrix parameters: hl



More DID parameters

- service
- version-id
- version-time
- hl
- initial-values
- transform-keys
- id
- fork

- There are also DID resolver input options.

Not supported by standard URI parsers?

- DID URLs (incl. matrix parameters) work with "standard" URI parsers !

- Java example (no error):

```
System.out.println(
    URI.create("did:ex:123;service=files/myresume/doc?version=latest#intro"));
```

- DID method name and method-specific identifier are also "non-standard".
- Key/value query parameters are also "non-standard".

Other concerns?

- Too "complex"?
- Let's not "resurrect" them?
- Use "special query parameters" instead?

- Proposal by Mike to remove them: #159
- Question by Ivan if they are normative: #137

Discussion

Use Cases?

DID-Based Data Hierarchy Portability - A User want to be able to change the root of a service.

Bootstrapping Communication - Enabling the establishment of communication with the DID subject or entities authorized on-behalf of the DID subject

Discovery Mechanism - For discovering information about the DID subject, (e.g useful in establishing trust in DIDs)

IoT Data Streaming - Give a datastream a DID and make all its data objects in the stream hash-addressable

Versions for Verifiers - A verifier would like to access a particular named or timed-based version of a set of cryptographic keys

Mapping the Crypto - pointing to particular elements of particular DID docs

Transforming Keys -

Data Hierarchy Portability without matrix parameters?

- With matrix parameter:

```
did:ex:123;service=files/myresume/doc?version=latest#intro
```

- With query parameter:

```
did:ex:123/myresume/doc?_did_service=files&version=latest#intro
```

- Result:

```
https://selfhosted.me:8080/myresume/doc?version=latest#intro
```

Morning break (10:30 - 10:45)

Spec Structure (Drummond, 45 min)

Goal of this session

- Based on our conclusions from Proposal #1 yesterday, discuss/decide on revised structure of DID Core spec
- Decisions are not final, but consensus from this group will accelerate the process not just for the Editors but all of us
- The proposed goal is to implement the restructure quickly so we can start doing PRs targeted to specific sections

Current Structure

1. Introduction
2. Terminology
3. Data Model
4. Decentralized Identifiers (ABNF)
5. DID Documents (Data Model)
6. DID Document Syntax
    a. JSON
    b. JSON-LD
7. DID Methods
8. DID Resolvers
9. Security Considerations
10. Privacy Considerations
11. Future Issues
12. Registries
13. Real World Example

New Structure

1. Introduction & Motivations
2. Terminology
3. DID Syntax (ABNF)
4. Abstract Data Model
5. Extensibility Mechanisms
6. Representation Requirements
7. Representations
    a. JSON
    b. JSON-LD
    c. CBOR
8. DID Methods
9. DID Resolvers
10. Security Considerations
11. Privacy Considerations
12. Future Issues
13. Registry Governance
14. Real World Example

Current Structure

1. Introduction
2. Terminology
3. Data Model
4. Decentralized Identifiers (ABNF)
5. DID Documents (Data Model)
6. DID Document Syntax
    a. JSON
    b. JSON-LD
7. DID Methods
8. DID Resolvers
9. Security Considerations
10. Privacy Considerations
11. Future Issues
12. Registries
13. Real World Example

New Structure

1. Introduction & Motivations
2. Terminology
3. Overall Architecture
4. DID Syntax (ABNF)
5. Extensibility
6. DID Documents
    a. Abstract Data Model
    b. Representation Requirements
    c. Representations
        i. JSON
        ii. JSON-LD
        iii. CBOR
7. DID Methods
8. DID Resolvers
9. Security Considerations
10. Privacy Considerations
11. Future Issues
12. Registry Governance
13. Real World Example

Current Structure

1. Introduction
2. Terminology
3. Data Model
4. Decentralized Identifiers (ABNF)
5. DID Documents (Data Model)
6. DID Document Syntax
    a. JSON
    b. JSON-LD
7. DID Methods
8. DID Resolvers
9. Security Considerations
10. Privacy Considerations
11. Future Issues
12. Registries
13. Real World Example

New Structure

1. Introduction
2. Terminology
3. Overall Architecture
4. Identifier
    a. Syntax
    b. .....
5. Data Model
    a. Definition
    b. Extensibility
    c. Representation Requirements
    d. Representations
        i. JSON
        ii. JSON-LD
        iii. CBOR
6. Methods
    a. Extensibility
7. Resolvers
8. Security Considerations
9. Privacy Considerations

Representation Requirements

1. MIME Type
2. Encoding Rules
3. Extension Rules
4. Roundtrip Lossless Conversion Rules
5. Fragment Processing Rules

Rubric (Joe Andrieu, 60 min)

Rubric for Decentralization of DID Methods

JOE ANDRIEU

DID WG FACE TO FACE AMSTERDAM 2020

JOE@LEGREQ.COM

Agenda

- Why a rubric
- Our approach
- Work to Date
- Lessons Learned
- Next Steps

Why a rubric

- Defining “decentralized” intractable
  - How decentralized MUST a method be?
- And yet… there are commonalities
- A Rubric offers a way forward
  - Method of evaluation (from education)
  - Multi-dimensional
  - Tailored to specific goals

What is a rubric?

- A set of criteria
  - specific questions
  - specific possible responses
  - explanation of how what each response means
- Can be consistently applied
  - by different evaluators
  - on different students/projects/products

Our approach

- Limited to “decentralization”
- Capture the motivations of DID community
- Not exhaustive
  - pick what matters
- NOT an authority for evaluations
- Make it easy for others to evaluate & compare

Work to Date

- Several IIW sessions
- Google docs set of questions
- Included in DID WG charter
- RWOT9 Paper (started in Sept)
  - Six co-authors

  Joe Andrieu joe@legreq.com , Shannon Appelcline shannona@skotos.net, Amy Guy amy@rhiaro.co.uk, Joachim Lohkamp joachim@jolocom.com, Drummond Reed drummond.reed@evernym.com, Markus Sabadello markus@danubetech.com, Oliver Terbu oliver.terbu@consensys.net, and Kai Wagner kai@jolocom.com
  - Almost ready for handing off to Chief Editor

Current Draft

- https://docs.google.com/document/d/1rYdWiwawWmLOWtHRvT0GzYcdewW_OS9M2mAkENLFdtY/edit?usp=sharing

Lessons Learned - Subjectivity

- Beauty is in the eye of the Evaluator
- Know **your** use cases
- Pick the most relevant criteria to **you**
- Record
  - Evaluator, Date, Use Cases
- Document **your** reasoning for each response

Lessons Learned – Categories Matter

- Governance
  - Rulemaking, Operations, Enforcement
- More than Governance
  - Alternatives
  - Adoption

Lessons Learned – Architecture

- Many criteria need to be applied across different architectural layers
- The layers vary
- Consolidating by layer GREATLY simplified the reports

Lessons Learned – Examples

- Examples are vital
- Can't include everyone
- Proposed constraints
  - ONLY 3 examples for each criteria
  - Examples should highlight differences
- Need help from Method implementers

Current Draft

- https://docs.google.com/document/d/1rYdWiwawWmLOWtHRvT0GzYcdewW_OS9M2mAkENLFdtY/edit?usp=sharing

Next Steps

- Get feedback from WG
  - Concerns
  - Consensus
  - Editors & Contributors
- Finalize RWOT Draft & Publish
- Convert to ReSpec and **FPWD**
- Weekly meetings with core team
  - Flesh out Enforcement
  - Method-specific calls for consideration

2nd Public Working Draft ~July 2020

Lunch (12:30 - 13:30)

Use cases and Requirements (Joe and Phil, 60 min)

Use Cases & Requirements

JOE ANDRIEU (AND CO-EDITOR PHIL ARCHER)

DID WG FACE TO FACE AMSTERDAM 2020

JOE@LEGREQ.COM

Agenda

- Why are we here?
- Work to Date
- How to Help

Why Use Cases & Requirements?

- Focus our work
  - Keep us from rat-holing on irrelevant discussions
  - Avoid spending time designing features no one needs
- Convince ourselves we are covering what we need
  - Make sure we aren't missing anything
- Communicate to others the value of our work
  - So developers can understand where we are coming from
  - So non-techies can understand what DIDs can do for them

More Why are we here?

- Capture the consensus of the working group
  - NOT to constrain the consensus
  - Not waterfall
  - Will adapt as we, as a group, learn more
- Verify that the Spec is finished

Work to Date

- Updated CCG draft
- Minimize solution language
- Enable terminology
- Five Focal Use Cases
- Tables of Features and Benefits
- Current Collecting Brief Use Cases

Focal Use Cases

1. Corporate Identifiers
2. Life-long Recipient-Managed Credentials
3. Prescriptions
4. Digital Executor
5. Single-Sign On

Brief Use Cases

1. Online Shopper
2. Vehicle Assemblies
3. Encrypted Data Vault
4. Accessing service endpoints
5. Verifiable Credentials

Current Gaps

- A. Use Case Domain Map (aka User Needs)
- B. Brief Use Cases (the paragraph for each use case in the domain map)
- ~~C. Terminology~~
- D. User Roles

How to Help: Brief Use Cases

- Write a one paragraph description of a single use case
  - One value-creating interaction
  - Not a category
  - Not a list
- Be specific
  - "Real" Person – Name, role/background, accessible, clear motivation
  - Real task – A specific thing they do, perhaps triggered by a specific event
- Describe what they do
  - Not why
  - Not how – avoid solution language
  - "Just the facts, Ma'am"
- Be distinct – cover a feature and situation not already covered

Possible Domains & Use Cases

- IoT
  - Localized Identifiers
- Crypto
  - Signing VPs
  - Negotiating ephemeral secrets
- Authorization
  - Independent Rotation
  - Banking Delegation
  - Chain of Authority
- Extensibility
  - New crypto
  - New service types
  - New methods
- Service Endpoints
- Matrix Parameters
- ~~Partial~~ Proxy Resolution

FabF3bruary

Publiek_TheOutlook

Afternoon break (14:30 - 14:45)

F2F Topics (Chairs, 75 min)

Metadata ..... continued

DID Doc/metadata/etc. Properties

- For each property
  - Where does/did it come from?
  - What do you use it for?

Properties 1

1. UNNAMED #1: A timestamp at which a DID was resolved
    a. What it is used for: cache invalidation
    b. Where does it come from: the resolution process
    c. Ganesh
2. UNNAMED #2: A self-attested timestamp at which the DID controller created it
    a. Used for: Contributes to validation of temporal flow
    b. Where does it come from: DID controller (software agent)
    c. Ganesh and Joe
3. Document serialization format (representation)
    a. Used for: parser/code dispatch—programmatically recognizing the type of representation
    b. Where from: output of resolution
    c. Justin

Properties 2

1. Capability invocation
    a. For: key material used to invoke an object capability, i.e., the public key associated with the private key that an entity uses to invoke an object capability. For example, if a car key had a fingerprint reader before it could be used.
    b. From: DID Controller (some entity)
    c. Manu
2. Capability delegation
    a. For: key material used to delegate a capability
    b. From: DID Controller (some entity)
    c. Manu

CLARIFICATION TO THE ABOVE: These are both assertions by the DID controller about which verification methods are authorized to invoke or delegate a capability on behalf of the DID subject

Properties 3

1. Controller Version Identifier of a DID document
    a. For: A version identifier for relying parties to know that a DID document changed. It may be different that a date provided by the resolution process.
    b. From: DID Controller (software agent, but not a blockchain)
    c. Christopher Allen
2. Redacted Property Value or Flag
    a. For: Allows a controlled to redact information in a DID document
    b. From: DID Controller (person or agent)
    c. Christopher Allen
3. Anti-Sybil / Reputation Information
    a. For: Allows the verifier to evaluate trust based on various anti-sybil mechanisms
    b. From: the Verified Data Registry
    c. Christopher Allen

Properties 4

1. Retired Keys
    a. For: Key validation, i.e., for checking proofs from older keys that should not be considered invalid, but are not the current keys
    b. From: DID controller (any)
    c. Joe
2. DID Created (Timestamp)
    a. For: The timestamp at which the Verifiable Data Registry finished the Create Operation. Finished is defined by when the DID can be resolved with its new changes applied.
    b. From: Verifiable Data Registry
    c. Markus and Ganesh
3. DID Updated (Timestamp)
    a. For: The timestamp at which the Verifiable Data Registry finished the Update Operation. Finished is defined by when the DID can be resolved with its new changes applied.
    b. From: Verifiable Data Registry
    c. Markus and Ganesh

Properties 5

1. DID Deactivated (Timestamp)
    a. For: The timestamp at which the Verifiable Data Registry finished the Deactivate Operation. Finished is defined by when the DID can be resolved with its new changes applied.
    b. From: Verifiable data registry
    c. Markus and Ganesh
2. Version Identifier of DID Document
    a. For: Cache invalidation, contribute to verification
    b. From: verifiable data registry
    c. Markus
3. DID Document Cached Flag
    a. For: Making trust decisions about the DID document
    b. From: Resolver
    c. Markus

Properties 6

1. Resolver Method Spec Version Identifier
    a. For: Security—knowing which version has been implemented
    b. From: Resolver
    c. Oliver
2. Resolver Method Code Version Identifier
    a. For: Security—knowing which version has been implemented
    b. From: Resolver
    c. Oliver
3. Resolver Engine Code Version Identifier
    a. For: Security—knowing which version has been implemented
    b. From: Resolver
    c. Oliver

Properties 7

1. Verifiable Proof of Control Authority (Full Proof)
    a. For: cryptographically establishing control authority over the DID. It means the root control over the DID, from which all other forms of control derive. It can be attenuated from there. It's a proof that proves the set of root keys that have control authority for a DID. It is proof of ultimate control authority. This also gives you any keys that have been retired.
    b. From: DID Controller (it is a cryptographic proof, could come from an agent)
    c. Sam Smith
2. Verifiable Proof of Control Authority (Hash and a Reference)
    a. For: Same as above
    b. For: Same as above
    c. Sam Smith

Properties 8

1. Requested Content Type
    a. For: Requester expressing to a DID resolver for the representation type of a DID document requested
    b. From: DID client or requester
    c. Tobias
2. Assertion (incomplete name)
    a. For: a DID controller to express verification methods that are authorized to assert things on behalf of the DID subject
    b. From: DID controller
    c. Tobias

Properties 9

1. Controller
    a. For: For expressing the entity that is in control of the DID. This is an indirection to another entity, which is what makes it different from Verifiable Proof of Control Authority.
    b. From: DID controller
    c. Manu
2. Key Agreement Key
    a. For: establishing a key for the purposes of establishing another key for encrypted communication
    b. From: DID controller
    c. Manu

Properties 10

1. DID URL method VDR identifier
    a. For: identifying VDR
    b. From: DID controller
    c. Joe
2. Target Verifiable Data Registry Identifier
    a. For: Identifying the VDR intended for the registration
    b. From: DID Controller
    c. Christopher Allen
3. Actual Verifiable Data Registry Identifier
    a. For: Verifying the VDR
    b. From: Resolver
    c. Christopher Allen

Additional Proposed Properties

- Add your properties here now (all of these will be self-asserted)
  - Arbitrary claims about the DID subject (phone number, country, etc.)
    - For: Making trust decisions about the DID subject from the DID document alone
    - From: DID controller
    - Posted by: Markus Sabadello
  - Optional multisig flag on keys “multisig list”, “singlesig’ “multisig aggregated”
    - For: as we move to a multisig world, some signatures types (Schnoor) the multisig can look like single sig (and thus are not-accountable/aggregate). In same cases the controller may want to inform the verifier it explicitly is multisig.
    - From: DID controller
    - Posted by: Christopher Allen

Additional Proposed Properties

- Please use Google Doc

Add:

https://docs.google.com/document/d/1WoHIA5MzC-kKdyS3XVp5qT-ZiNUbpqXH59g3Q9Fnk04/edit?usp=sharing

Overlap with DID Resolution (Chairs, 30 min)

(Wrap up, 15 min)

The Ending Stuff

Dinner

Meeting length, structure, activities

Regarding requests for April face-to-face (& IIW discussion?)

Next week - face to face recovery, so no meeting

Our hosts are awesome

And so are we all

Not the end . . .