This is a proposed collaborative paper to be completed at RWOT 2022, Den Haag, Netherlands, 26-30 September.

Did Resources for SSI - address ALL requirements via DIDs

By: Mirko Mollik TrustCerts
Date: Jul 15, 2022 Version: 0.1

Abstract

When validating credentials the minimal requirement is a public key to validate the signature. This problem is already solved with the W3C did core spec. But when looking at verifiable credentials other resources like schemas or revocation lists are needed. Hyperledger Indy was designed to solve this addressing problem, but is doing it on its own way. In a perfect DID world the resources to validate a credential can be quired from multiple VDRs, as we know it from the WWW when requesting HTML,CSS,JS or audio files to display a webpage.

Current problem

When talking about self sovereign identities we are talking about digital identities and verifiable claims. The first part can already be solved with decentralized identifiers with the help of published crypto materials like public keys or service endpoints to get more information. This is already covered by the W3C spec and there are dozens of methods that connect to different verifiable date registries.

Most of these methods are using a distributed system to request the did methods. But other resources like JSON-schemas are queried via an URL and point to a single point of failure. The same problem goes with the Revocation-List approach where the list can be deployed on a web server.

Solution

The DID Core spec has only one requirement for the document: it needs a unique id. But all other properties are optional and more properties can be added.

Instead of defining ONE did method that fits for all the approach should be to define some kind of modules how to implement different resources inside a did method. The id for a schema could be defined like:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
```

```
    ...
  ]
  "id": "did:myMethod:schema:38dsNNd83nwdwdh",
  values: {
    "type": "object",
    "properties": {
      "greeting": {
        "type": "string
       },
    }
  }
}
```

Where `myMethod` could be a any did method that wants to implement this resource.

Benefits building on the did core spec

- SSI resources are based on a standard
- allows versioning of DID documents (schemas could be immutable, but revocation lists are designed to be updated)
- Hashlinks allow integrity checks
- Controllers are managing responsibilities

Open Question

- is this approach aligned with other data format like json-ld?
- is it possible to generalize it so it does not get in conflict with the open approach of the core?
- Will this approach make it more complex since multiple VDRs are offering the same schema, but with another URI?
- are there resources that can not be put inside a json structure?