PRISM SCHEMA DEFINITION

PROOFSPACE.ID

https://linktr.ee/proofspace

// RUSLAN SHEVCHENKO <RUSLAN@PROOFSPACE.ID>

Prism VC Schema/Cred Definition: F8 https://cardano.ideascale.com/c/idea/400403

Goal — receive early feedback from developers.
setup communication channels

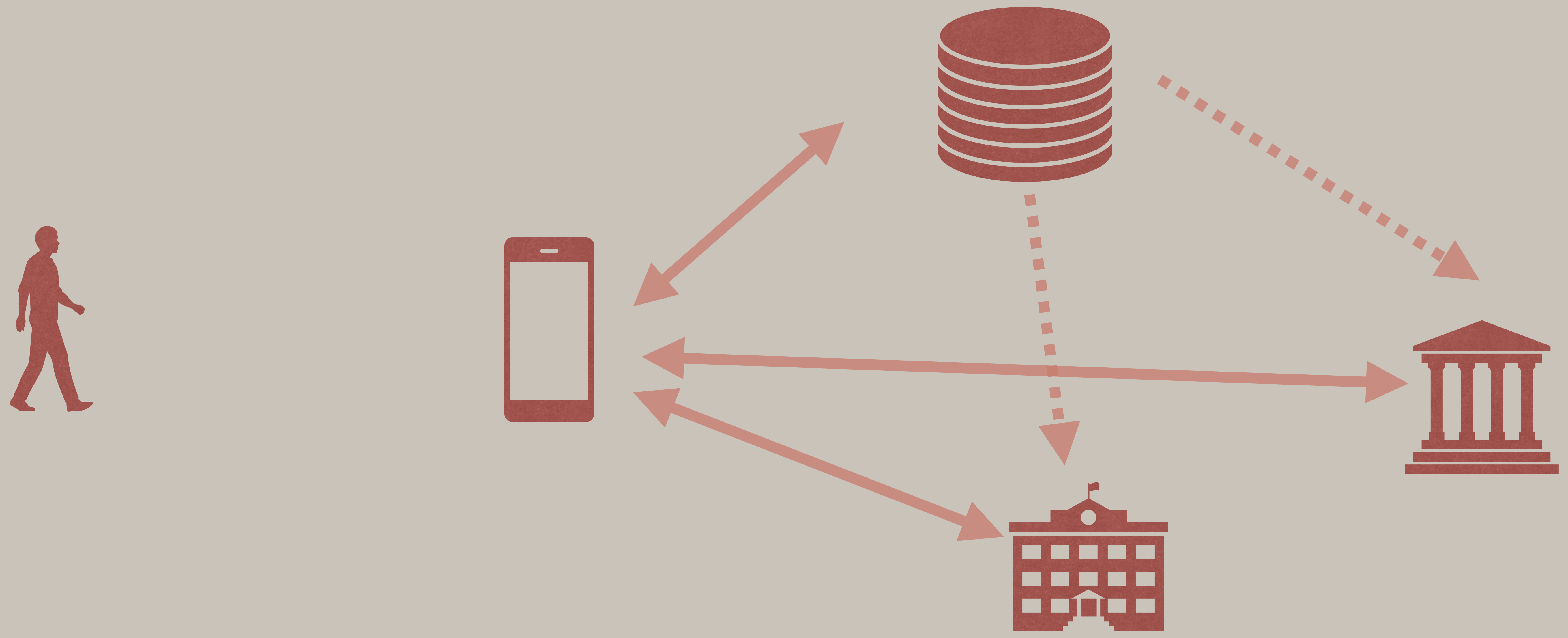
(Agenda = Frequently Asked Questions):

- What is a credential schema and why is it needed?
- Why can’t we use existing JSON-LD Standard?
- What are similar generic schema standards?
- What do other SSI project use for the similar purposes?
- What properties (dimensions) we want to describe?

- What is a credential schema and why is it needed?

```
{
  name: “Basic Passport Data“,
  author: “did:prism:fe6f01c776514efa82c82a73fa00c0c91368ff……",
  version: “1.0”,
  trustRegistry:  { type:  "permissionless" },
  properties: {
       Country: { type: “string”, enum: [ ….””],   }
       names: {  type: “object“, properties:{
            “First Name”: {  type:  string, maxLength: 50  },
            “Last Name” : {  type:  string, maxLength: 50  },
        }},
        namesInternational: {. ……. }
      “passportId”: {  type: “string”,  indexable: true  }
      “Credential Issue Date”:  { type: Date }
  }
  Uniqueness:  [“Country”,”passportId”]
}
```

- What is a credential schema and why is it needed?
    - Contains semantic description of data inside credential.
    - Tools can use schema to load, verify and process credential.
    - Stored in public blockchain.
    - Can be retrieved from credential



- Typical scenario:

  - request credential by schema;
  - receive and verify credential;
  - verify that credential is issued by issuer from trust registry;
  - Do some custom processing, based on fields.

-

- Why can’t we use existing JSON-LD Standard?

Verifiable credential data model.

- https://json-ld.org/
- https://www.w3.org/TR/json-ld11/

Schema = “context definition”

```
{
  "@context": {
    "Person": "http://www.w3.org/ns/person#Person",
    "alternativeName": {
      "@id": "http://purl.org/dc/terms/alternative",
      "@type": "http://www.w3.org/2001/XMLSchema#string"
    },
    "birthName": {
      "@id": "http://www.w3.org/ns/person#birthName",
      "@type": "http://www.w3.org/2001/XMLSchema#string"
    },
    "citizenship": {
      "@id": "http://www.w3.org/ns/person#citizenship",
      "@type": "http://purl.org/dc/terms/Jurisdiction"
    },
```

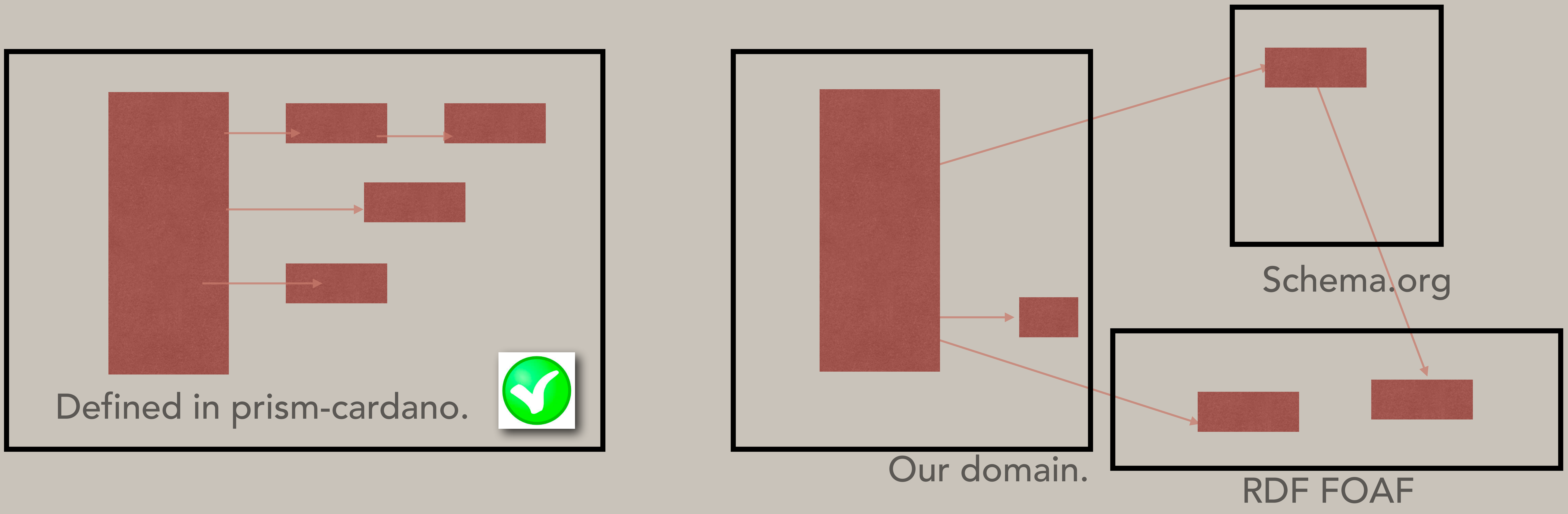
```
{
  "@context": "http://www.schema.org",
  "@type": "Person",
  "@id": "https://jay.holtslander.ca/#person",
  "name": "Jay Holtslander",
  "alternateName": "Jason Holtslander",
  "nationality": "Canadian",
  "birthPlace" : {
    "@type": "Place",
    "address": {
      "@type": "PostalAddress",
      "addressLocality": "Vancouver",
      "addressRegion": "BC",
      "addressCountry": "Canada"
    }
  },
  "affiliation": [
    {
      "@type": "Organization",
```

Why we can't just get existing JSON-LD Context:

- Open definitions.

  Schemas can refers to another schemas in external sources.

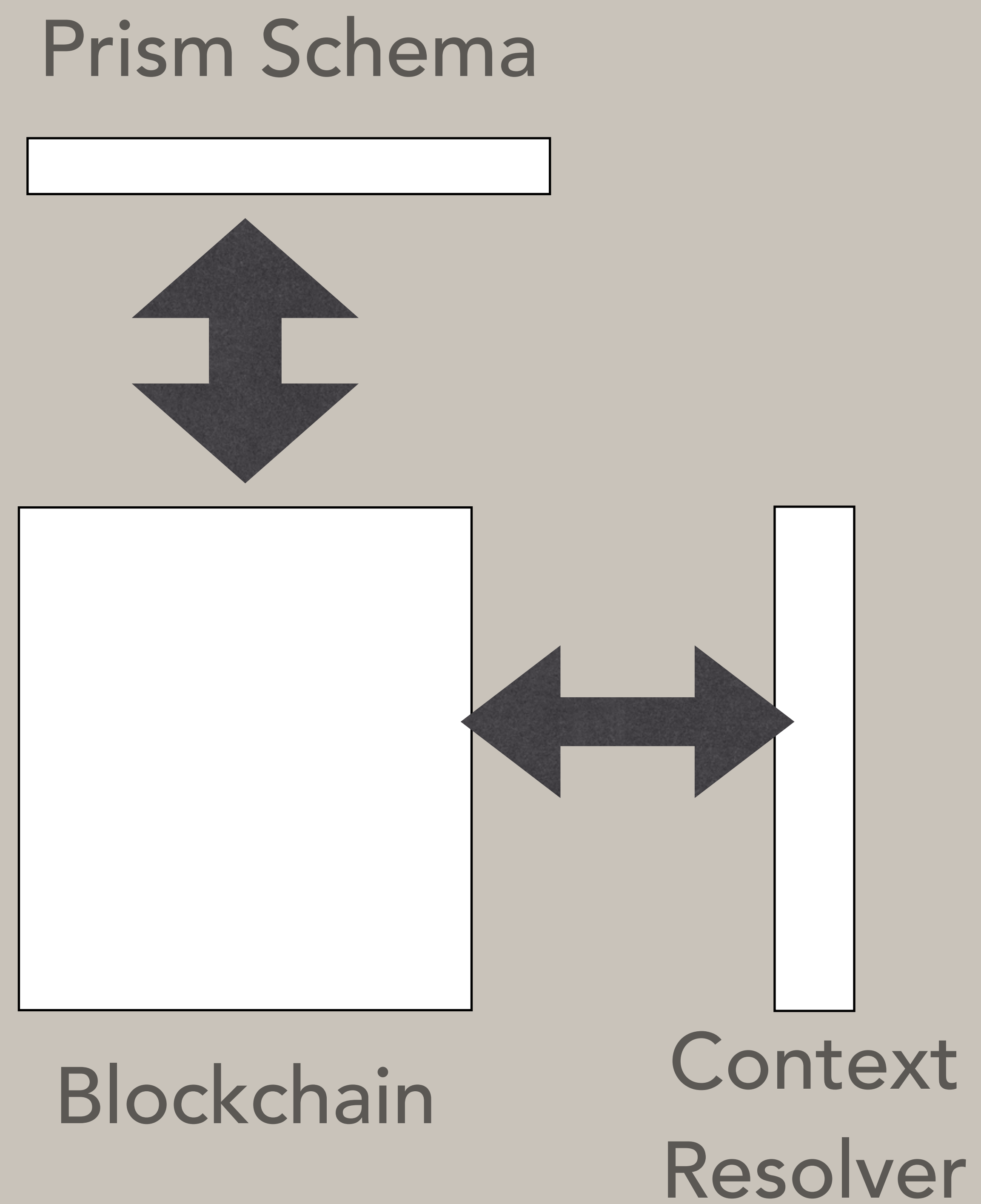
  We want only references to previously defined schemas in our sources.



- Why can’t we use existing JSON-LD Context?

- Open definitions.
  Schemas can refers to another schemas in external sources.
  We want only references to previously defined schemas in our sources.
- Set of primitive types is not fixed.
  Convention — all primitive types are defined in (schema.org, xmlns.org, etc)
  Big vocabularies.
- Base model: JSON-LD defined on RDF which historically linked with XML ….
  Complexity inherited from history

- But Verifiable Credential Data Model use JSON-LD ?
  - If we want to support it, then we should map our model to json-ld @context and implement context resolver.



JSON-LD. @Context

- What are similar generic schema standards ?

  - json_schema: https://json-schema.org/
    - describe and verify any json.
    - based more on verification than semantics.

  (We will use json_schema for defining of our definitions,
  Full json_schema is quite big)

JSON SCHEMA

Exists standard for verifiable credentials:
https://w3c-ccg.github.io/vc-json-schemas/v1/index.html

```
{
  "type": "https://w3c-ccg.github.io/vc-json-schemas/schema/1.0/schema.json",
  "modelVersion": "1.0",
  "id": "did:ethr:rsk:0x8a32da624dd9fad8bf4f32d9456f374b60d9ad28;id=1eb2af6b-0dee-6090-cb55-0ed093f9b026;version=1.0",
  "name": "EmailCredentialSchema",
  "author": "did:ethr:rsk:0x8a32da624dd9fad8bf4f32d9456f374b60d9ad28",
  "authored": "2020-11-20T03:22:00-03:00",
  "schema": {
    "$schema": "http://json-schema.org/draft-07/schema#",
    "description": "Email",
    "type": "object",
    "properties": {
      "emailAddress": {
        "type": "string"
      }
    },
    "required": ["emailAddress"],
    "additionalProperties": true
  }
}
```

- metainformation.

- Set of required properties is a schema attribute ....

JSON SCHEMA

Exists standard for verifiable credentials:
https://w3c-ccg.github.io/vc-json-schemas/v1/index.html

— Set of supported datatypes is different in json-ld contexts and json_schemas.

— Exists many tools, which allow to automatically display UI forms,

— Can be too verbose for inline definition [?] (from other side - actually used).
Current approach: generate from smaller definition.
(Want to hear options).

- ## What are similar generic schema standards ?

JSON Type Definitions:
RFC 8927: https://datatracker.ietf.org/doc/rfc8927/

— Looks like stripped-down version of json_schema.

— Main purpose: generate type definitions: https://github.com/jsontypedef

— Too minimal for our purpose [we need extra vocabulary for describing vc]

Can include this as subset [Discussion topic].

- What are similar generic schema standards ?

- IDL-s: (Interface Definition Language).

fragmented (exists many flavors)

- Created for binary serialisation.
- Also usually define interfaces.
- Nice to have as option
  - (human-readable schema.)
- For tooling json is better.

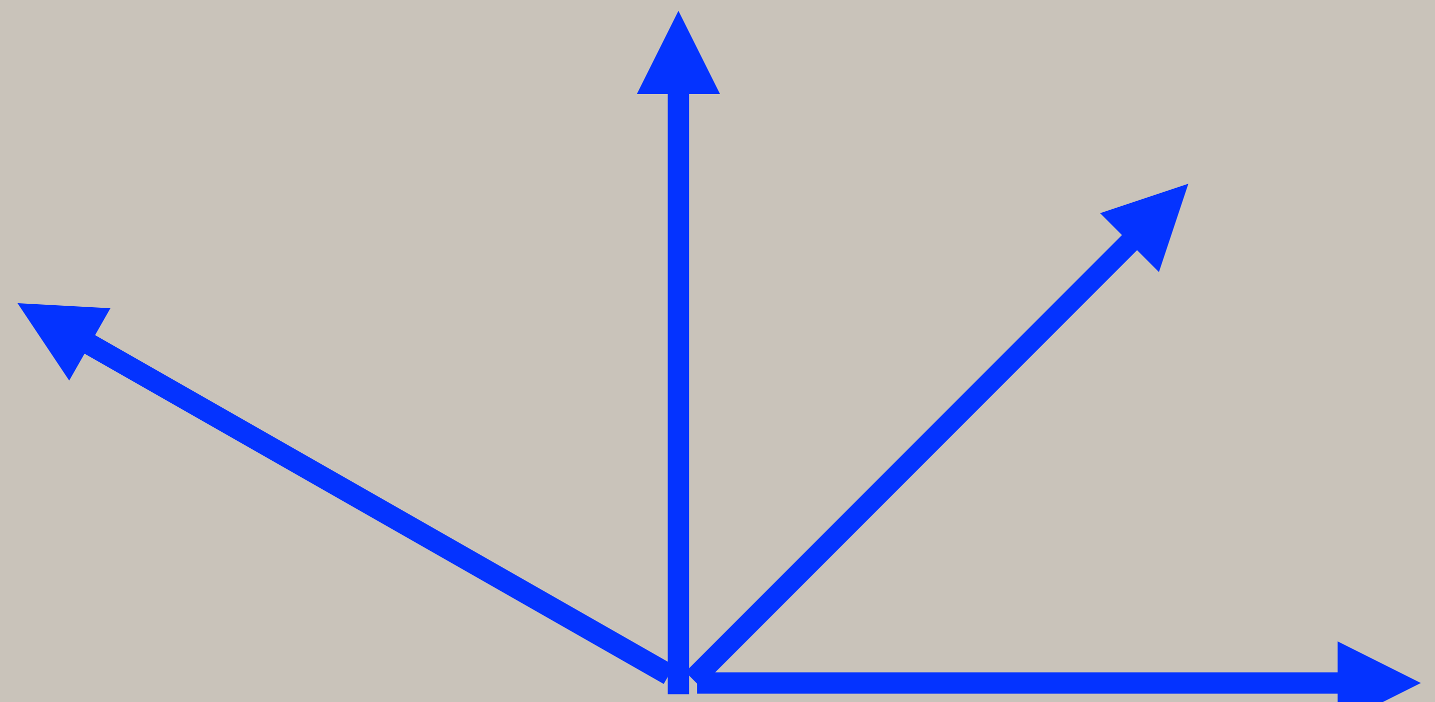
```
syntax "proto3"

message Person {
  string firstName = 1;
  string lastName = 2;
  int32 age = 3;
}
```

What other SSI project use for the similar purpose ?

- Indy: simple json with plain set of fields.
- Aries Rich Schema Objecte: JSON-LD context
  https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0250-rich-schemas/
- KILT CTYPE: Subset of json_schema.
  https://docs.kilt.io/docs/concepts/credentials/ctypes
- RSK.IO: json_schema. https://github.com/rsksmart/vc-json-schemas
- trinsic.id: json_schema. https://docs.trinsic.id/docs/issue-credentials
- EBSI: json_schema.
  https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/
- Serto.Id: generated both json-ld context and json_schema: https://schemas.serto.id/
- Some Eth-based projects IDL: EIP-1812 https://eips.ethereum.org/EIPS/eip-1812

What we need in cardano-specific credentials schema:

- Axes. (Dimensions)
  - Public/Private, Transfer/Reveal restrictions, Verification Policy,
  - Structure for Verticals, Indexes, etc.
- Limitations.
  - Generation of json-ld, json_schema, mapping to security signatures, etc …



Verification:

To be aware of limitations of common verification schemas, if ones will be available

- Simple: we have signature of a document and publish signature and merkle tree in blockchain.
  - What is currently implemented in PRISM.

- ZKP (zero knowledge predicate): we can ask a question about credential and receive proof in answe
  - Implementation schemas:
    - Camenish-Luchanska [Indy, Aries]

      Map selected attributes into array
    - Bulletproof   (early) POC implementation - (https://github.com/MarcKloter/zkStrata)
    - ZK-SNARK (microsoft) (early)

      https://github.com/decentralized-identity/snark-credentials/blob/master/whitepaper.pdf

- Selective disclose: we can ask a value of some subset of credentials data.
  - Implementation schemas:
    - BLS signatures. [W3C]   https://arxiv.org/abs/2006.05201

      Mark subsets which signatures we want to aggregate

Structure:

MetaInfo:

- Name
- Version
- Trust Registry
- Author
- Domain Uniqueness

Objects:

Map<String, Schema>

Array<Schema>

Properties - known names.

// Identifiers [name] or human-readable [type] .
// Locale-aware ?
// Are we want to support nested objects mapping ?
// How we will support proofs limitations ?

Primitives:

- String,
  - ValueId
- Number:
  - Integer
  - Decimal
  - Float
- Boolean
- Enum
- Date, Timestamp
  - Issuing date
  - Expire Date
- Duration
- ? - images, binary data

MetaInfo:

- Id — Unique uri (not transaction id)
- Name
- Version
- Author
- Trust Registry
  - Permission less (no trust registry)
  - One Issuer
  - Token-Curated
- Domain Uniqueness
  - set of fields, which should be unique in domain.

```
   "name":
   "version": { "type":  "string" },
   "description": { "type": "string" },
   "id": {  "type": “string" },
   "author": { "$ref":  "#/$defs/did" },
   "trustRegistry": {
      "$ref": "#/$defs/trustRegistry"
   },
  "uniquiness": { "type": "array", "items": { "type":"string" }  },
},
```

Objects:

Map<String, Field>

Array<Field>

Field:

```
title
fieldName
description
contextUri
optional
disclosable
comparable
indexable
unique
```

For human. (By default - property name)

For code generation. (By default - property name)

For json_ld. (By default - standard mapping)

For BLS+ signatures.

For ZKP schemas.

For 'search button' in tool.

For integrity control

Primitives:

- String,
  - ValueId
- Number:
  - Integer
  - Decimal
  - Float
- Boolean
- Enum
- Date, Timestamp.
  - Issuing date
  - Expire Date,
- ? - images, binary data

Multi-line
Keyboard-hint

Special attribute

JSON SCHEMA

no Null

Additional

Special attribute

Current Approach:

Light json-based metaschema.

GitHub: https://github.com/zakaio/atala-prism-schema

Let's collaborate on GitHub project discussion forum and/or Astros #SSIAlliance slack chat

Current activities:

- Convert some examples from root-id interoperability catalog
- json-ld and json_schema mapping
- Endpoint for publishing
  - Schema resolver

Github: https://github.com/zakaio/atala-prism-schema

Let's collaborate on GitHub project discussion forum and/or Astros #SSIAlliance slack chat

Questions / Comments ?