Workday Credentials Schema Specification; 1.0

- Authors:
  - Bjorn Hamel bjorn.hamel@workday.com
  - Gabe Cohen gabe.cohen@workday.com
  - Joe Genereux joe.genereux@workday.com
  - Jonathan Reynolds jonathan.reynolds@workday.com
  - Rory Martin rory.martin@workday.com

- Last updated: 2019-09-27

Status

- Status: **PROPOSAL**
- Status Date: 2019-09-27

Abstract

The W3C Verifiable Credentials Data Model specifies the models used for Verifiable Credentials and Verifiable Presentations, and explains the relationships between three parties: *issuer*, *holder*, and *verifier*. A critical piece of infrastructure out of the scope of those specifications is the **Credential Schema**. This specification provides a mechanism to express a Credential Schema and the protocols for evolving the schema over time.

Contents

Standards & Compliance

The W3C Verifiable Credentials Data Model specifies the models used for Verifiable Credentials and Verifiable Presentations, and while the Workday Credentials Schema Spec attempts to align closely to the W3C model, it should **NOT** be assumed this specification is fully compliant at the time of this writing. Ultimately, we would like to be fully compliant with the W3C spec; however, due to the fact that the W3C specification is currently under active development and is subject to change, we have diverged from that model with several "forks" that are fully outlined in this and other Workday Credentialing specifications. As such, while the W3C specification is our target for compliance, at this time, *this document* should be treated as the source of truth for interoperating with the Workday Credentialing ecosystem for Credential Schemas.

Formatting

The W3C Verifiable Credentials Data Model provides its examples in the JSON Linked Data (JSON-LD) interchange format. The specification allows for other formats, such as standard JSON with JSON Schema but provides only limited examples. In the credentialSchema section, *JSON-SCHEMA-2018* validation is explicitly called out. This specification does not use JSON-LD. If it becomes evident that it would be useful to include JSON-LD or another format that decision would be made in a revisal draft at a later date.

The Workday VC data model relies heavily upon standard JSON with validation provided by JSON Schema Draft 7. The data model embeds JSON Schema documents inside a larger document that contains useful metadata about a given credential schema.

Specification

1.0 Introduction

Workday Credentialing provides a mechanism for the use of verifiable credentials in a highly scalable, secure, and verifiable way. A large part of the integrity of a verifiable credential is how to structure the credential so that all three parties (issuer, holder, verifier) may have a singular mechanism of trust into what they are using. At Workday we call that document a *Credential Schema*.

This specification provides a standardized way of creating Credential Schemas to be used in the Workday Platform, how to version them, and how to read them.

1.1 What is a Credential Schema?

The **Credential Schema** is a document that is used to guarantee the structure, and by extension the semantics, of the set of claims comprising a Verifiable Credential. A shared Credential Schema allows all parties to reference data in a known way.

A schema can be viewed from four perspectives: the author, issuer, verifier and holder.

Author: An author creates a schema as a blueprint for a verifiable credential, specifying the shape and format of the data in such a credential.

Issuer: Issuers utilize schemas to provide structure and meaning to the data they issue as verifiable credentials. By using schemas, issuers contribute to a credentialing ecosystem, and promote the use and adoption of data standards.

Verifier: A verifier requesting data needs to do so with knowledge of the available credentials shapes. Credential Schemas allow a Verifier to ask for data knowing that an issuer has issued in an understood way and that a holder's wallet can find data matching that requested.

Holder: Holders, or those who are the subject of credential issuance, can make sense of the data they own -- values -- by viewing it against a schema -- keys. When data is requested from them by referencing a Credential Schema, this known structure allows the holder's wallet to return the data specifically requested by the verifier.

1.2 Schema Guarantees

By adhering to the specification, the following guarantees can be made about a schema:

- A schema is *versionable* and new versions can be created to evolve it over time
- A schema is publicly available for any issuer to use and any verifier, or other platform member to *read*
- A schema always guarantees the structure of a credential. The described structure can be used by the Verifier to understand what data the Holder holds. There is no requirement for the Verifier to validate sent data against the schema since this sent data may only be partial, for example in event of a proof request only requiring a single field from a credential with multiple fields defined in the schema.

1.3 Where Can I Find a Credential Schema?

Credential Schemas are created and made publicly available as immutable objects on the Workday Credentialing distributed ledger.

1.4 A Note on Versioning

Schemas are versioned in two ways, via `modelVersion` and `version` properties, both of which are expanded upon later in this document. Versioning provides benefits for schema authors, issuers, and verifiers to make sense of their data.

Authors and Issuers care about versioning to track advancements and changes over time both for formatting changes (e.g. supporting JSON Schema Draft 7 as opposed to Draft 6) as well as fields as a schema converges to its most currently usable form (e.g. adding a new required field). Holders care about versioning to know where and how their credential can be used. Similarly, Verifiers care about versioning to know which data, or model versions they should accept in their systems.

2.0 Credential Schema Definition

This section provides the json-schema definition for Credential Schema along with an example of a Credential Schema for an Email Verified Credential.

JSON Schema

▶ Show/Hide JSON Schema

Example


```
{
  "type": "https://credentials.workday.com/docs/credential-schema.json",
  "modelVersion": "1.0",
  "id": "did:work:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326fbe21db;versior
  "name": "EmailCredentialSchema",
  "author": "did:work:MDP8AsFhHzhwUvGNuYkX7T",
  "authored": "2018-01-01T00:00:00+00:00",
  "schema": {
    "$schema": "http://json-schema.org/draft-07/schema#",
    "description": "Email",
    "type": "object",
    "properties": {
      "emailAddress": {
        "type": "string",
        "format": "email"
      }
    },
    "required": ["emailAddress"],
    "additionalProperties": false
  },
  "proof": {
      "created": "2019-09-27T06:26:11Z",
      "creator": "did:work:MDP8AsFhHzhwUvGNuYkX7T#key-1",
      "nonce": "0efba23d-2987-4441-998e-23a9d9af79f0",
      "signatureValue": "2A7ZF9f9TWMdtgn57Y6dP6RQGs52xg2QdjUESZUuf4J9BUnwwWFNL8vFshQAEQF
      "type": "Ed25519VerificationKey2018"
  }
}
```


2.1 Type

It is important in software systems for machines to be able to understand the context of what a document is. In credential schemas this is declared in the **type** field. This field resolves to a JSON schema with details about the **schema metadata** that applies to the given schema.

2.2 Model Version

After a machine has parsed the type property it should know that the document it is reading is a credential schema. The next field is the version, which simply denotes what version of the **schema metadata** this is.

2.3 ID

A globally unique identifier to locate the schema on the Workday distributed ledger. Each credential schema has its own unique identifier, and each *version* of a credential schema is required to have its own unique identifier.

This identifier is a method-specific DID parameter name based upon the author of the schema. For example, if the author had a did like `did:work:abcdefghi` a possible schema ID the author created would have an identifier such as: `did:work:abcdefghi;schema=17de181feb67447da4e78259d92d0240;version=1.0`

2.4 Schema Version

Schema versioning is defined as **MODEL.REVISION** where **MODEL** is a breaking change and **REVISION** is non-breaking. The version is contained within the schema identifier.

With schemas we are concerned with a new schema and backwards compatibility of *existing data* on an older schema.

MODEL Updating this number tells the end user that this version breaks the schema for ANY interaction with an older schema. For verification if a holder presents a credential built from a schema of version 1.0 and the platform is only looking for > 2.0, it is *not* able to parse ANY information.

REVISION Updating this number tells the end user that this version may prevent interactions with parts of the schema. For verification if a holder presents a credential built from a schema of version 1.0 and the platform is looking for > 1.5, there are likely to be SOME fields that are incompatible with the expected credential.

REVISION

The addition or removal of an**optional** field is what would typically constitute a**REVISION**. Removing or adding an optional field does not break historical data in a schema and in the claims exchange protocol fields that are returned negative in the optional field can be ignored as they are optional by default.

```
{
 "type": "https://credentials.workday.com/docs/credential-schema.json",
 "modelVersion": "1.0",
 "id": "did:work:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326fbe21db;version=
 "name": "EmailCredentialSchema",
 "author": "did:work:MDP8AsFhHzhwUvGNuYkX7T",
 "authored": "2018-01-01T00:00:00+00:00",
 "schema": {
   "$schema": "http://json-schema.org/draft-07/schema#",
   "description": "Email",
   "type": "object",
   "properties": {
      "emailAddress": {
      "type": "string",
      "format": "email"
    }
   },
   "required": ["emailAddress"],
   "additionalProperties": false
 }
}
```


In this example we once again reference the email schema, but this time we add an optional field *backupEmailAddress*. Notice how this would not break the claims exchange because the field is*optional*.

```
{
 "type": "https://credentials.workday.com/docs/credential-schema.json",
 "modelVersion": "1.0",
 "id": "did:work:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326fbe21db;version=
 "name": "EmailCredentialSchema",
 "author": "did:work:abc123",
 "authored": "2018-01-01T00:00:00+00:00",
 "schema": {
   "$schema": "http://json-schema.org/draft-07/schema#",
   "description": "Email",
   "type": "object",
   "properties": {
    "emailAddress": {
      "type": "string",
      "format": "email"
    },
    "backupEmailAddress": {
      "type": "string",
      "format": "email"
    }
   },
   "required": ["emailAddress"],
   "additionalProperties": false
 },
```


MODEL

When a schema breaks historical data we call it a model change. This is the major differentiating point between other schema versioning protocols which allow for some breaking changes because as we learned in the problem section, even breaking one area can lead to very difficult issues for verifiers through the credential exchange protocol. The most common case of a **MODEL** change is the addition or subtraction of a required field. It is also important to note that for the change of a key name on a required field constitutes a MODEL change. Why? Because technically that introduces a breaking change and adds a required field.

An example of this rule is when the `additionalProperties` field's value changes. Changing `additionalProperties` from `false` to `true` **OR** from `true` to `false` constitutes a breaking change, necessitating a **MODEL** increment.

```
{
 "type": "https://credentials.workday.com/docs/credential-schema.json",
 "modelVersion": "1.0",
 "id": "did:work:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326fbe21db;version=
 "name": "EmailCredentialSchema",
 "author": "did:work:MDP8AsFhHzhwUvGNuYkX7T",
 "authored": "2018-01-01T00:00:00+00:00",
 "schema": {
   "$schema": "http://json-schema.org/draft-07/schema#",
   "description": "Email",
   "type": "object",
   "properties": {
    "emailAddress": {
      "type": "string",
      "format": "email"
    },
    "backupEmailAddress": {
      "type": "string",
      "format": "email"
    }
   },
   "required": ["emailAddress"],
   "additionalProperties": false
 },
```


This time our credentialing requirements for email have changed and email address is no longer enough information on a credential, and we need to attach a name for verification as well to our schema. This is a *required* field, so we know it is a**MODEL** change.

```
{
 "type": "https://credentials.workday.com/docs/credential-schema.json",
 "modelVersion": "1.0",
 "id": "did:work:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326fbe21db;version=
 "name": "EmailCredentialSchema",
 "author": "did:work:MDP8AsFhHzhwUvGNuYkX7T",
 "authored": "2018-01-01T00:00:00+00:00",
 "schema": {
   "$schema": "http://json-schema.org/draft-07/schema#",
   "description": "Email",
   "type": "object",
   "properties": {
    "emailAddress": {
      "type": "string",
      "format": "email"
    },
    "firstName": {
      "type": "string"
    },
    "backupEmailAddress": {
      "type": "string",
      "format": "email"
    }
   },
   "required": ["emailAddress", "firstName"],
   "additionalProperties": false
 },
```

2.5 Name

A human-readable name for the schema.

2.6 Author

DID of the identity which authored the credential schema.

2.7 Authored

RFC-3339 date on which the schema was created.

2.8 Schema

This is where the Credential Schema data fields are defined

```
{
 "$schema": "http://json-schema.org/draft-07/schema#",
 "description": "Email",
 "type": "object",
 "properties": {
   "emailAddress": {
       "type": "string",
       "format": "email"
   }
 },
 "required": ["emailAddress"],
 "additionalProperties": false
}
```

Extensibility

By introducing a `modelVersion` field we allow the credential schema to become extensible. Properties such as `derivedFrom` could be used to reference a schema that a new schema is built on top of. Similarly, platform-utility features such as searchability could be provided by adding a `tags` array that contains categorization and classification information for a given schema.

These are just a few examples that illustrate the flexibility of the proposed model. It can be extended to support a wide variety of use-cases and make the burden on issuance and verification simpler by facilitating the development of higher-level tooling.

Drawbacks

Within the Workday Credentialing Ecosystem, relying heavily upon JSON Schema makes the data shape for credentials consistent, and could result in an ecosystem with many similar schemas with slight changes (naming, capitalization). Without proper oversight or authoritative schemas to limit duplication or misuse utilization of JSON Schema could result in a poor user experience. At the platform level tooling can be provided to minimize confusion and promote reuse.

Within the broader Credentialing Ecosystem, interoperability could be more difficult if the wider community adopts a standard such as JSON-LD and does not promote or support the usage of JSON Schema based schemas or credentials. This issue can mainly be side-stepped with the metadata we include -- the *Credential Schema* –– since our model is flexible to change. A new**modelVersion** could be introduced that supports JSON-LD and removes support for JSON Schema. A drawback here is the requirement that all schemas have this piece of metadata, which itself is versioned and evolvable.

A flip side to drawbacks of the usage of JSON Schema is that there is a plethora of documentation, libraries, and usage of JSON Schema across programming languages and the web.

Alternatives

JSON-LD schemas is the most prominent alternative.

Prior Art

- The Verifiable Credential Specification is valuable for providing initial context.
- Hyperledger Indy uses schemas in a similar way to the description in this document.

Security & Privacy Considerations

Privacy & security considerations mainly revolve around Personally Identifiable Information (PII) leaks in schemas. Any field which a user could enter data is a potential area for personally identifiable information. When implementing systems that support the storage and querying of schemas relevant data privacy laws and regulations must be taken into account.

Interoperability Considerations

The primary concern of this specification is to facilitate an ecosystem in which Verifiable Credentials can be issued and utilized. In order to be interoperable, additional schema types may need to be supported. Given the investment into a robust versioning strategy of our **Credential Schema Metadata** interoperability with the current design is less of a concern.

A goal of publishing this document is to promote others to adopt our schema philosophy. It also opens the door for providing feedback and collaborative contribution to developing primitives that would result in a successful verifiable ecosystem.

Glossary

Schema Metadata: Top level information about a credential schema. An example for an email credential schema is provided below.

```
{
  "type": "https://credentials.workday.com/docs/credential-schema.json",
  "modelVersion": "1.0",
  "id": "did:work:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326fbe21db;versior
  "name": "Email",
  "author": "did:work:MDP8AsFhHzhwUvGNuYkX7T",
  "authored": "2018-01-01T00:00:00+00:00"
}
```

Credential Schema: The data template for a credential. An example of an email credential schema is provided below.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "description": "Email",
  "type": "object",
  "properties": {
    "emailAddress": {
      "type": "string",
      "format": "email"
    }
  },
  "required": ["emailAddress"],
  "additionalProperties": false
}
```

References

- Verifiable Credential Specification
- DID Specification
- JSON Schema
- JSON-LD
- RFC-3339
- Hyperledger Indy Schemas