DID Specification

Editors: Drummond Reed, Les Chasen
Contributors: Christopher Allen, Manu Sporny, David Longley, Jason Law, Daniel Hardman, Markus Sabadello

STATUS: Working Draft 04, 24 October 2016

ACKNOWLEDGMENTS: Work on this specification has been funded in part by the United States Department of Homeland Security's Science and Technology Directorate under contract HSHQDC-16-R00012-H-SB2016-1-002. The content of this specification does not necessarily reflect the position or the policy of the U.S. Government and no official endorsement should be inferred.

Work on this specification has also been supported by the Rebooting the Web of Trust group facilitated by Christopher Allen, Brian Weller, Kiara Robles, and Shannon Appelcline.

*Note: terms in **bold** are defined in the Terminology section.*

ABSTRACT

DIDs (decentralized identifiers) are a new type of identifier intended for verifiable digital identity that is “self-sovereign”, i.e, fully under the control of the **identity owner** and not dependent on a centralized registry, identity provider, or certificate authority. DIDs resolve to **DDOs** (DID descriptor objects)—simple JSON documents that contain all the metadata needed to prove ownership and control of a DID as well as share the cryptographic keys and resource pointers necessary to initiate trusted interactions with the identity owner. Each DID uses a specific **DID method**, defined in a separate **DID method specification**, to define how the DID is registered, resolved, updated, and revoked on a distributed ledger or other form of decentralized network.

Table of Contents

1. Introduction

Conventional identity management systems are based on centralized authorities such as corporate directory services, certificate authorities, or domain name registries. From the standpoint of cryptographic trust verification, each of these centralized authorities serves as its own root of trust. To make identity management work across these systems requires implementing federated identity management.

The emergence of **distributed ledger technology** (DLT), sometimes referred to as **blockchain** technology, provides the opportunity to implement fully **decentralized identity management** (DIDM). In DIDM, all participants with identities (called **identity owners**) share a common root of trust in the form of a globally distributed ledger (or a decentralized P2P network that provides similar capabilities).

In a DIDM architecture, each identity owner can be identified on a ledger with a key-value pair. The index key is a **DID** (**decentralized identifier**) and the value is its associated **DDO** (**DID description object**). Together these form a **DID record**. Each DID record is cryptographically secured by private keys under the identity owner's control. Following the dictums of Privacy by Design, each identity owner may have as many DID records as necessary to respect the identity owner's desired separation of identities, personas, and contexts.

This architecture not only eliminates dependence on centralized registries for identifiers, but also on centralized certificate authorities for key management as is typical of hierarchical PKI (public key infrastructure). Instead each identity owner serves as its own root authority via its own DID record(s) on the shared ledger—an architecture called DPKI (decentralized PKI).

2. Example DID and DDO

This example of a DID uses the Sovrin DID method defined in Appendix A:

```
did:sov:21tDAKCERh95uGgKbJNHYp
```

Following is an example of a DDO that describes this DID. Note that the other DIDs included in this example represent other DID methods that have been proposed for Bitcoin and Ethereum.

```
{
      "@context": "https://example.org/did/v1",
      "id": "did:sov:21tDAKCERh95uGgKbJNHYp",
      "equiv-id": [
            "did:sov:33ad7beb1abc4a26b89246",
            "did:btc1:794856-624",
            "did:uport:0xa9be82e93628abaac5ab557a9b3b02f711c0151c"
      ],
      "owner": [{
            "id": "did:sov:33ad7beb1abc4a26b89246#key/1",
            "type": "Ed25519",
            "expires": "2017-02-08T16:02:20Z",
            "key":
"IOmA4R7TfhkYTYW87z640O3GYFldw0yqie9Wl1kZ5OBYNAKOwG5uOsPRK8/2C4STOWF+
83cMcbZ3CBMq2/gi25s="
      }, {
            "id": "did:sov:33ad7beb1abc4a26b89246#key/2",
            "type": "rsa256",
            "expires": "2017-03-22T00:00:00Z",
            "key":
"MIIBOgIBAAJBAKkbSUT9/Q2uBfGRau6/XJyZhcF5abo7b37I5hr3EmwGykdzyk8GSyJK
3TOrjyl0sdJsGbFmgQaRyV"
      }],
      "control": [
            "self",
            "did:sov:bsAdB81oHKaCmLTsgajtp9AoAHE9ei4",
            "did:sov:21tDAKCERh95uGgKbJNHYpE8WEogrsf"
      ],
      "service": {
            "openid": "https://openid.example.com/456",
            "xdi": "https://xdi.example.com/123"
      },
      "type": "http://schema.org/Person",
      "creator": "did:sov:21tDAKCERh95uGgKbJNHYpE8WEogrsf",
      "created": "2002-10-10T17:00:00Z",
      "updated": "2016-10-17T02:41:00Z",
      "signature": {
            "type": "LinkedDataSignature2015",
            "created": "2016-02-08T16:02:20Z",
            "creator":
"did:sov:21tDAKCERh95uGgKbJNHYpE8WEogrsf#keys/1",
            "signatureValue":
"IOmA4R7TfhkYTYW87z640O3GYFldw0yqie9Wl1kZ5OBYNAKOwG5uOsPRK8/2C4STOWF+
83cMcbZ3CBMq2/gi25s="
      }
}
```

[TODO - Complete all examples above and generate JSON-LD context file.]

3. Purpose of this Specification

The purpose of this specification is to define the two logical components of DID records—DIDs and DDOs—in a manner capable of being implemented on any DLT or decentralized network capable of accepting DID records. It is out of scope for this specification to define the precise method by which DID records shall be implemented on any particular DLT or decentralized network—that is the job of a separate **DID method specification**.

Conceptually, the relationship of this specification and a DID method specification is similar to the relationship of the IETF generic URI specification (RFC 3986) and a specific URI scheme specification (such as the http: and https: schemes specified in RFC 7230). It is also similar to the relationship of the IETF generic URN specification (RFC 2141) and a specific URN namespace definition (such as the UUID URN namespace defined in RFC 4122). The key difference is that a DID method specification, in addition to specifying a specific DID scheme, must also specify the methods for reading, writing, and revoking DID records on the DLT or decentralized network for which it is written.

This specification defines the requirements of a conformant DID method specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

4. Terminology and Acronyms

Alias. A hash of a conventional globally unique identifier based on traditional registry services, such as a mobile telephone number, an email address, or a URI (Uniform Resource Identifier). Aliases can be a type of **NCID** used for **DID** discovery. See Appendix B.

Blockchain. A specific type of **distributed ledger technology** (DLT) that stores ledger entries in blocks for efficiency. Because this type of DLT was introduced by Bitcoin, the term “blockchain” is sometimes used to refer specifically to the Bitcoin ledger.

CID. Acronym for **cryptographic identifier**.

Cryptographic identifier (CID). A **DID** with specific cryptographic properties as defined by a **DID method specification**. A CID has an associated private key or signing key that can be used to verify ownership of the CID.

Cryptonym. Another name for a **CID**.

Decentralized identifier (DID). A globally unique identifier that does not require a centralized registration authority. The generic format of a DID is defined in this specification. A specific **DID scheme** is defined in a **DID method specification**. A DID may be either a **cryptographic identifier** (CID) or a **non-cryptographic identifier** (NCID).

Decentralized identity management (DIDM). Identity management based on decentralized identifiers that do not require centralized authorities such as those required by X.500 directory services, the Domain Name System, national ID systems, etc.

DDO. Acronym for **DID descriptor object**.

DID. Acronym for **decentralized identifier**.

DID descriptor object (DDO). A JSON data structure containing metadata describing an identity owner, including the cryptographic key material required for the identity owner to prove ownership and control of the **DID record**. A DDO may also contain other attributes or claims describing the identity owner.

DID method. A definition of how a specific **DID scheme** can be implemented on specific DLT or other decentralized network, including the precise method(s) by which DIDs and DDOs can be read, written, and revoked.

DID method specification. The specification for a specific **DID scheme** and **DID method** that is conformant with the requirements of this specification.

DID record. The combination of a **DID** and a **DDO** that forms the "root identity record" for an identity. From the standpoint of claims-based identity, a DID record is the "genesis claim" for an identity.

DID scheme. The formal syntax of a DID identifier. The generic DID scheme is defined in this specification. A specific DID scheme that works with a specific **DID method** is defined in a **DID method specification**.

DIDM. Acronym for **decentralized identity management**.

Distributed ledger technology (DLT). A distributed database in which the various nodes use a consensus protocol to maintain a shared ledger in which each transaction is cryptographically signed and chained to the previous transaction. See also **blockchain**.

DLT. Acronym for **distributed ledger technology**.

DPKI. Acronym for decentralized **PKI**.

Identity owner. The person, organization, or thing whose identity is represented by a **DID record**. (Note: this specification avoids the term "user" since an identity owner is not always an individual person.) Note that the identity owner may not be the director controller of the DID record; for example a parent may control the DID record(s) for a young child, but the child is the identity owner.

Identity record. Another name for a **DID record**.

JSON-LD (JSON Linked Data). A method of encoding Linked Data using JSON. JSON-LD enables object properties in a JSON document to be linked to concepts in an **RDF** ontology.

Ledger. In the context of this specification, a shared database of transactions maintained via **distributed ledger technology**.

NCID. Acronym for **non-cryptographic identifier**.

Non-cryptographic identifier (NCID). A **DID** that does not have any cryptographic properties. An NCID does not have an associated private key or signing key. A **UUID** is an example of an NCID.

PKI. Acronym for **public key infrastructure**.

RDF. Acronym for Resource Description Framework—a semantic graph model defined by the W3C RDF Working Group.

Relying party (RP). The entity accepting a digital identity asserted by an identity owner.

RP. Acronym for **relying party**.

Service endpoint. A network address at which a service operates on behalf of an identity owner. Examples of specific DIDM services include discovery services, authentication services, authorization services, interaction services, etc. A **DIDM** service endpoint may also be provided by a generalized data interchange protocol such as **XDI**.

UUID. Universally Unique Identifier as specified by RFC 4122.

URI (Uniform Resource Identifier). The standard identifier in Web architecture defined by IETF RFC 3986.

URN (Uniform Resource Name). A URI intended to serve as a persistent, location-independent identifier of a resource, i.e, an identifier that will always represent the same resource and never change (ideally forever). URN syntax is defined by IETF RFC 2141.

XDI. Acronym for Extensible Data Interchange (also XRI Data Interchange)—a semantic graph format and semantic data interchange protocol defined by the OASIS XDI Technical Committee.

5. Design Goals & Principles

This section summarizes the design goals and principles of DID architecture.

5.1. Decentralization

DID architecture should eliminate the requirement for centralized authorities or single points of failure in identity management, including the registration of globally unique identifiers, public verification keys, service endpoints, and other metadata.

5.2. Self-Sovereignty

DID architecture should give identity owners the power to directly own and control their own digital identities without the need to rely on external authorities.

5.3. Privacy

DID architecture should enable identity owners to control the privacy of their digital identities, including selective disclosure of attributes or other identity data.

5.4. Security

DID architecture should provide sufficient security for relying parties to rely on DID records to their required level of assurance.

5.5. Cryptography

DID architecture should enable an identity owner to provide cryptographic proof of ownership and proof of access control rights.

5.6. Discoverability

DID architecture should make it possible for entities to discover DID records for other entities to learn more about or interact with those entities.

5.7. Interoperability

DID architecture should use interoperable standards so DIDs and DID records can be used with the maximum number of existing systems.

5.8. Portability

DID architecture should be system-independent and enable identity owners to use their digital identities with any system that supports DIDs and DIDM.

5.9. Simplicity

To meet these design goals, DID architecture should be (to paraphrase Albert Einstein) “as simple as possible but no simpler”.

5.10. Extensibility

When possible, DID architecture should enable extensibility provided it does not hinder interoperability, portability or simplicity.

6. DIDs (Decentralized Identifiers)

The foundation of DID architecture is the concept of the decentralized identifier. This concept is not new; the structure of UUIDs (Universally Unique IDentifiers) was first developed in the 1980s and later became a standard feature of the Open Software Foundation’s Distributed Computing Environment. UUIDs achieve global uniqueness without a centralized registry service by using an algorithm that generates 128-bit values with sufficient entropy that the chance of collision are infinitesimally small. UUIDs are formally a URN (Uniform Resource Name) namespace specified in IETF RFC 4122.

However the UUID URN specification does not provide:

1. A standard method for resolving the UUID to discover more about or interact with the resource it identifies.
2. A standard mechanism for proving control and ownership of a UUID, or

This is the motivation for DIDs: a new type of decentralized URN that can have one or both of these additional properties. The abstract syntax of a DID is specified by the generic DID scheme defined in section 6.1. The concrete syntax of a specific DID scheme is defined in a DID method specification. Note that a specific DID scheme MAY define either or both types of DIDs:

1. **CIDs** (cryptographic identifiers) are encodings, hashes, or other derivations of public keys or verification keys that are globally unique and have specific cryptographic properties.
2. **NCIDs** (non-cryptographic identifiers) are globally unique identifiers that do not have any cryptographic properties (a UUID is an example of an NCID).

A DID is intended to serve as an index key (not to be confused with a cryptographic key) to a corresponding value—the DDO. See section 7.

6.1. The Generic DID Scheme

The generic DID scheme is a URI (Uniform Resource Identifier) scheme conformant with RFC 3986. Due to the limited character set, it also an IRI (Internationalized Resource Identifier) scheme conformant with RFC 3987. Following is an ABNF definition using the ABNF syntax defined in RFC 5234.

```
did                = "did:" method ":" specific-idstring
                     ["/" path ] [ "#" fragment ]
specific-idstring  = idstring *( ":" idstring )
method             = 1*idchar
idstring           = 1*idchar
idchar             = ALPHA / DIGIT / "." / "-"
```

See sections 6.3 and 6.4 for the balance of the ABNF rules defining DID paths and DID fragments.

6.2. Specific DID Schemes

A DID method specification MUST define exactly one specific DID scheme identified by exactly one method name (the `method` rule in section 4.1). Since DIDs are intended for decentralized identity infrastructure, it is NOT RECOMMENDED to establish a registry of unique DID method names. Rather the uniqueness of DID method names should be established via human consensus, i.e., a specific DID scheme MUST use a method name that is unique among all DID method names known to the specification authors at the time of publication.

Since the method name is part of the DID, it SHOULD be as short as practical. A method name of four characters or less is RECOMMENDED. The method name MAY reflect the name of the DLT or decentralized network to which the DID method specification applies.

If needed, a specific DID scheme MAY define multiple specific ID formats (the `method-idstring` rule in section 4.1). It is RECOMMENDED that a specific DID scheme define as few specific ID formats as possible.

6.3 DID Paths

A DID—the decentralized identifier that uniquely represents an identity owner—is the entire identifier defined in the ABNF in section 6.1 WITHOUT the path component or fragment identifier. If a DID is followed by a path component, the path MUST NOT be considered part of the DID. The DID path (the `path` rule in section 6.1) MUST be used only to address resources identified by a DID service endpoint. See section 7.7.

[TODO: Add ABNF for DID paths, which should be identical to the path component of RFC 3986.]

Note that these rules are identical to the path rules in RFC 3986, i.e., a DID path is identical to a URI path.

A specific DID scheme MAY specify ABNF rules for DID paths that are more restrictive than the rules in this section.

6.4 DID Fragments

As stated in section 6.3, a DID is the identifier defined in the ABNF in section 4.1 WITHOUT the path component or fragment identifier. If a DID is followed directly by a fragment identifier, the fragment identifier MUST NOT be considered part of the DID. The fragment identifier (the `fragment` rule in section 6.1) MUST be used only as a pointer into the DDO to identify a unique cryptographic key or other component of the DDO.

Following are the ABNF rules for a DID fragment identifier.

```
fragment      = *( pchar / "/" / "?" )
pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"
unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
pct-encoded   = "%" HEXDIG HEXDIG
sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
                / "*" / "+" / "," / ";" / "="
```

Note that these rules are identical to the fragment rules in RFC 3986, i.e., a DID fragment is identical to a URI fragment.

A specific DID scheme MAY specify ABNF rules for DID fragments that are more restrictive than the rules in this section.

6.5 DID Normalization

For the broadest interoperability, DID normalization should be as simple and universal as possible. Therefore:

1. The "did" scheme name MUST be lowercase.
2. The method name MUST be lowercase.
3. Case sensitivity and normalization of a specific idstring (the `specific-idstring` rule in section 6.1) MUST be defined by a DID method specification.

6.6 DID Persistence

Ideally a DID would be a completely abstract decentralized identifier (like a UUID) that could be bound to multiple underlying DLTs or decentralized networks over time, thus maintaining its persistence independent of any particular ledger or network. However registering the same identifier on multiple ledgers or networks introduces extremely hard identity ownership and start-of-authority (SOA) problems. It also greatly increases implementation complexity for clients.

To avoid these issues, it is RECOMMENDED that DID method specifications only produce DIDs and DID methods bound to strong, stable ledgers or networks capable of making the highest level of commitment to persistence of the DID and DID method over time.

It is also RECOMMENDED to establish verifiable equivalence statements between DID records representing the same identity owner on multiple ledgers or networks. Such equivalence statements can produce the practical equivalent of a single persistent abstract DID. DID equivalence statements are addressed in section 5.2.

7. DDOs (DID Descriptor Objects)

If a DID is the index key in a key-value pair, the DDO is the value to which the index key points. A fundamental goal of this specification is to define DIDs and DDOs such that a DID will always returns a valid DDO regardless of the underlying DLT or decentralized network for which the DID method is defined. This enables a foundational layer of root identity objects to interoperate across all DLTs and decentralized networks that support DIDs.

To support this interoperability, a DDO is specified as a single JSON object conforming to RFC 7159. For purposes of this version of the DID specification, the format of this JSON object is specified in JSON-LD, a format for mapping JSON data into the RDF semantic graph model as defined by the W3C JSON-LD 1.0 specification. Future versions of this specification MAY specify other semantic graph formats for a DDO such as JXD (JSON XDI Data), a serialization format for the XDI graph model as defined by the OASIS XDI Core 1.0 specification.

The following sections define the members of this JSON object, including whether these members are required or optional.

7.1. Context (Required)

JSON objects in JSON-LD format must include a JSON-LD context statement. The rules for this statement are:

1. A DDO MUST have exactly one top-level context statement.
2. This statement MUST be the first line in the JSON object. (This is not strictly necessary under JSON-LD but is required for DDOs.)
3. The key for this member MUST be `@context`.
4. The value of this key MUST be the URL specified in the example below.

Example:

```
{
      "@context": "[URL-for-JSON-LD-DDO-context-map]"
}
```

[TODO: Generate canonical URL for JSON-LD context map for DDOs. Also, consult with Manu on generic vs. DID method specific context maps.]

7.2. Primary DID (Required)

This primary DID is the primary index key for the DDO. The rules for a primary DID are:

1. A DDO MUST have exactly one member representing the primary DID.
2. The key for this member MUST be `id`.
3. The value of this key MUST be a valid DID according to a specific DID scheme as defined in section 6.

Example:

```
{
      "id": "did:sov:21tDAKCERh95uGgKbJNHYp"
}
```

7.3. Equivalent DIDs (Optional)

An equivalent DID is an additional index key for the DDO. Under the RDF OWL graph model, this is a DID that has an owl:sameAs relationship with the primary DID. Under the XDI graph model, it is an XDI address that has an XDI $is identity equivalence relationship with the primary DID.

The rules for equivalent identifiers are:

1. A DDO MUST have zero or one member representing an array of equivalent DIDs.
2. The key for this member MUST be `equiv-id`.
3. The value of this key MUST be an array.
4. The values in this array MUST be valid DIDs according as defined in section 6.

Example:

```
{
      "equiv-id": [
            "did:sov:33ad7beb1abc4a26b89246",
            "did:btc1:79485-624",
            "did:uport:0xa9be82e93628abaac5ab557a9b3b02f711c0151c"
      ]
}
```

[TODO: Equivalence is still a topic of active discussion among the editors and contributors. It may be removed from the V1 specification. At a minimum we need to add guidance about the requirements for proving equivalence.]

7.4. Proof of Ownership (Optional)

Proof of Ownership is the mechanism by which an identity owner can cryptographically prove ownership of a DID. This is indicated by publication of at least one public key or verification key. Note that Proof of Ownership is separate from Proof of Control because an identity owner may wish to enable other entities to update the DDO (for example, to assist with key recovery) without enabling them to prove ownership (and thereby be able to impersonate the identity owner).

The rules for Proof of Ownership are:

1. A DDO MUST have exactly one member for providing Proof of Ownership.
2. The key for this member MUST be `owner`.
3. The value of this key MUST be an array where each member of the array is a JSON object describing a valid public key or verification key as defined by the DID method specification for the primary DID.
4. If this array contains more than one member, the JSON object representing each member MUST include a member with the key `id` whose value is the primary DID concatenated with valid DID fragment identifier.

[OPEN ISSUE: Clarification of the key description format in JSON-LD.]

Example:

```
{
      "owner": [{
            "id": "did:sov:33ad7beb1abc4a26b89246#key/1",
            "type": "sovrin.1",
            "expires": "2017-02-08T16:02:20Z",
            "key":
"IOmA4R7TfhkYTYW87z640O3GYFldw0yqie9Wl1kZ5OBYNAKOwG5uOsPRK8/2C4STOWF+
83cMcbZ3CBMq2/gi25s="
      }, {
            "id": "did:sov:33ad7beb1abc4a26b89246#key/2",
            "type": "rsa256",
            "expires": "2017-03-22T00:00:00Z",
            "key":
"MIIBOgIBAAJBAKkbSUT9/Q2uBfGRau6/XJyZhcF5abo7b37I5hr3EmwGykdzyk8GSyJK
3TOrjyl0sdJsGbFmgQaRyV"
      }]
}
```

7.5. Proof of Control (Optional)

Proof of Control is the mechanism by which an identity owner can give itself or other entities permission to update the DDO (for example to assist with key recovery) without being able to prove ownership. Unlike Proof of Ownership, Proof Of Control may be indicated by publishing one or more DIDs.

To give identity owners maximum flexibility for key recovery, Proof of Control SHOULD support both multiple keys and multisignatures.

The rules for Proof of Control using multiple keys are:

1. A DDO MUST have exactly member representing Proof of Control.
2. The key for this member MUST be `control.`
3. The value of this key MUST be an array.
4. The values in this array MUST be either:
    a. The string “self”.
    b. A valid DID.
    c. A public key or verification key as defined by the DID method specification for the primary DID.
    d. Another specific proof of control mechanism defined by the DID method specification for the primary DID.
5. If one of the values of this array is “self”, then any key in the `owner` member specified in section 7.4 MUST be considered valid for Proof of Control.

[OPEN ISSUE: Clarification of other proof-of-control methods that should be universal across all DID method specifications. Note that this may include Smart Signatures when it is ready.]

Example:

```
{
	"control": [
		"self",
		"did:sov:bsAdB81oHKaCmLTsgajtp9AoAHE9ei4",
		"did:sov:21tDAKCERh95uGgKbJNHYpE8WEogrsf"
	]
}
```

7.6. Service Endpoint References (Optional)

After publication of cryptographic key material, the other primary purpose of DID records is to enable discovery of service endpoints for the identity owner. A service endpoint may represent any type of service the identity owner wishes to advertise, including DIDM services for further discovery, authentication, authorization, or interaction.

The rules for service endpoints are:

1. A DDO MUST have zero or one member representing service endpoints.
2. The key for this member MUST be “service”.
3. The value of this key MUST be a nested JSON object.
4. In this nested JSON object, each member MUST have a key representing the name of a service established in the JSON-LD context map.
5. The value of this key MUST be a valid IRI conforming to RFC 3987 that represents the service endpoint.

Example:

```
{
	"service": {
		"openid": "https://openid.example.com/456",
		"xdi": "https://xdi.example.com/123"
	}
}
```

[OPEN ISSUE: Discuss with Manu Sporny whether each service needs additional description and thus this should be an array of JSON objects.]

7.7. Identity Type (Optional)

An identity type is a semantic assertion about the type of resource an identity owner represents (i.e., a person, an organization, a type of thing).

The rules for an identity type are:

1. A DDO MUST have zero or one member representing an identity type.
2. The key for this member MUST be `type`.
3. The value of this key MUST be either:
    a. A valid IRI conforming to RFC 3987 that represents identity type.
    b. A valid XDI address conforming to XDI Core 1.0 that represents the identity type.

Example:

```
{
      "type": "http://schema.org/Person"
}
```

7.8. Creator (Optional Unless No Proof of Ownership)

In certain cases, one identity owner (called the creator) may provision an identity record for another identity owner who is not in a position to hold or control the necessary cryptographic keys (such as a parent creating an identity record for a young child). In this case, there are no verkeys to represent the ultimate identity owner. So the DDO needs to express the identity of the creator.

The rules for a creator are:

1. A DDO MUST have zero or one member representing a creator.
2. The key for this member MUST be `creator`.
3. The value of this key MUST be a valid DID as defined in section 6.

Example:

```
{
      "creator": "did:sov:21tDAKCERh95uGgKbJNHYpE8WEogrsf"
}
```

7.9. Created (Optional)

Standard metadata for identity records includes a timestamp of the original creation. The rules for including a creation timestamp are:

1. A DDO MUST have zero or one member representing a creation timestamp.

2. It is RECOMMENDED that each DID include this member.
3. The key for this member MUST be `created`.
4. The value of this key MUST be a valid XML datetime value as defined in section 3.3.7 of [W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes](https://www.w3.org/TR/xmlschema11-2/).
5. This datetime value MUST be normalized to UTC 00:00 as indicated by the trailing “Z”.

Example:

```
{
      "created": "2002-10-10T17:00:00Z"
}
```

[OPEN ISSUE: What do we do about approximate timestamps, e.g., a bitcoin record being written between the start and end of a 10-minute block?]

7.10. Updated (Optional)

Standard metadata for identity records includes a timestamp of the most recent change. The rules for including a updated timestamp are:

1. A DDO MUST have zero or one member representing an updated timestamp.
2. It is RECOMMENDED that each DID include this member.
3. The key for this member MUST be `updated`.
4. The value of this key MUST be a valid XML datetime value as defined in section 3.3.7 of [W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes](https://www.w3.org/TR/xmlschema11-2/).
5. This datetime value MUST be normalized to UTC 00:00 as indicated by the trailing “Z”.

Example:

```
{
      "updated": "2016-10-17T02:41:00Z"
}
```

[OPEN ISSUE: Same as above, i.e., what do we do about approximate timestamps?]

7.11. Signature (Required)

The signature is cryptographic proof of the validity of the DDO according to either:

1. The verkey of the identity owner as defined in section 5.4, or if not present:
2. The verkey of the DID creator as defined in section 5.9.

The rules for a signature are:

1. A DDO MUST have zero or one member representing a signature.
2. The key for this member MUST be `signature`.
3. The value of this key MUST be a valid JSON-LD signature as defined by [OPEN ISSUE: need current reference]

[OPEN ISSUE: Need to specify any additional rules for applying an JSON-LD signature.]

Example:

```
{
      "signature": {
            "type": "LinkedDataSignature2015",
            "created": "2016-02-08T16:02:20Z",
            "creator":
"did:76d0cdb7-9c75-4be5-8e5a-e2d7a35ce907/keys/1",
            "signatureValue":
"QNB13Y7Q9oLlDLL6AHyL31OE5fLji9DwJSA8qnv81oRaKonij8m+Jv4XdiEYvJ97iRlz
KU/92/0LafSL5JftEgl960DLcbqMFxOtbAmFOIMa7eDcrgTL5ytXeYCYKLjHQG3s8a3UK
DKRuEK54qK1G5hGKGoLgAVa6xgcDLjW7M19PEJV/c3HpGA7Eez6VFMoTt4yESjZvOXC97
xN3KpshOx2HT/btgUbo0XjA1Oi0QHdgrLcUsQGt6w23RjeSToalrsA1G69OFeN2OiQrz9
Jb4561hvKLSyWObwRmS6n5Vgr5xkvUm6MONRq0Vg33kXevoVM64KTBkISul61tzjn4w==
"
      }
}
```

8. DID Operations

To enable the full functionality of DIDs and DDOs on a particular DLT or decentralized network (called the *target system*), a DID method specification MUST specify how each of the following CRUD operations is performed by a client. Each operation MUST be specified to the level of detail necessary to build and test interoperable client implementations with the target system.

Note that, due to the specified contents of DDOs, these operations can effectively be used to perform all the operations required of a CKMS (cryptographic key management system), e.g.:

- Key registration
- Key replacement
- Key rotation
- Key recovery
- Key expiration

8.1. Create

The DID method specification MUST specify how a client creates a DID record—the combination of a DID and its associated DDO—on the target system, including all cryptographic operations necessary to establish proof of ownership.

8.2. Read/Verify

The DID method specification MUST specify how a client uses a DID to request a DDO from the target system, including how the client can verify the authenticity of the DDO.

8.3. Update

The DID method specification MUST specify how a client can update a DID record on the target system, including all cryptographic operations necessary to establish proof of control.

8.4. Delete/Revoke

Although a core feature of distributed ledgers is immutability, the DID method specification MUST specify how a client can revoke a DID record on the target system, including all cryptographic operations necessary to establish proof of control.

9. DID Resolvers

[TODO: Specify any normative requirements for DID resolvers.]

10. Security Considerations

[TODO]

11. Privacy Considerations

[TODO]

12. References

[RFC-KEYWORDS] Key words for use in RFCs to Indicate Requirement Levels. IETF RFC 2119. https://www.ietf.org/rfc/rfc2119.txt

[ABNF] Augmented BNF for Syntax Specifications: ABNF. IETF RFC 5234. https://tools.ietf.org/html/rfc5234

[SBIR-TOPIC] Applicability of Blockchain Technology to Privacy Respecting Identity Management. U.S Department of Homeland Security Small Business Innovation Research Grant. https://www.sbir.gov/sbirsearch/detail/867797

[SBIR-SUBMISSION] Respect Network submission for DHS SBIR topic on the Applicability of Blockchain Technology to Privacy Respecting Identity Management. Respect-Network-HSHQDC-16-R00012-H-SB2016-1-002

[URI] Uniform Resource Identifiers. IETF RFC 3986. https://www.ietf.org/rfc/rfc3986.txt

[URN] URN (Uniform Resource Name) Syntax. IETF RFC 2141. https://tools.ietf.org/rfc/rfc2141.txt

[IRI] Internationalized Resource Identifiers. IETF RFC 3987. https://www.ietf.org/rfc/rfc3987.txt

[UUID] A Universally Unique IDentifier (UUID) URN Namespace. IETF RFC 4122. https://www.ietf.org/rfc/rfc4122.txt

[JSON] The JavaScript Object Notation (JSON) Data Interchange Format https://tools.ietf.org/html/rfc7159

[JSON-LD] JSON-LD 1.0. http://www.w3.org/TR/json-ld/

[VCTF] W3C Verifiable Claims Task Force. http://opencreds.org/specs/source/claims-data-model/#expressing-entity-credentials-in-json

[XDI-CORE] OASIS XDI Core 1.0 Specification Working Draft 01 http://docs.oasis-open.org/xdi/xdi-core/v1.0/csd01/xdi-core-v1.0-csd01.xml

[XML-DATETIME] W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. W3C Recommendation. https://www.w3.org/TR/xmlschema11-2/

Appendix A: The Sovrin DID Method

[NOTE: This appendix is provided in this Working Draft for illustration purposes. The Sovrin DID method specification will be moved to a separate specification like all other DID method specifications.]

Sovrin is a public permissioned ledger for self-sovereign identity governed by the global non-profit Sovrin Foundation. This appendix is the beginning of the DID method specification for the Sovrin DID method.

A.1. The Sovrin DID Scheme

The Sovrin DID scheme is defined by the following ABNF:

```
sovrin-did     = "did:sov:" idstring
idstring       = 22*22char
char           = ALPHA / DIGIT
```

This means all Sovrin DIDs are exactly 22 characters representing a base58Check encoding of the first 16 bytes of a 256 bit Ed25519 verification key (the public portion of the key pair).

A.2. Verification Keys

It is impossible to tell from a Sovrin DID alone whether it is a CID or NCID. Rather the associated DDO MUST be checked to determine one of three options for the associated verification key (verkey).

1. **Empty.** In this case, the DDO does not contain a verkey, and the DID is an NCID. In this case, the creator of the Sovrin identity record (called a trust anchor) controls the identifier, and no independent proof-of-existence is possible until the DDO is updated to contain either an Abbreviated or Full verkey.
2. **Abbreviated.** In this case, the DDO contains a verkey starting with a tilde '~' followed by 22 characters. The tilde indicates that the DID itself represents the first 16 bytes of the verkey, and the string following the tilde represent the second 16 bytes of the verkey, both using base58Check encoding.
3. **Full.** In this case, the DDO contains a full 44 character verkey, representing a base58Check encoding of all 32 bytes of a Ed25519 verification key.

Note that an abbreviated verkey has one security benefit when it is first created: there is a provable binding between the DID and the DDO at the time of registration. However this could also be accomplished by having the Sovrin client signing a nonce, or by some other cryptographic handshake defined by a DID method specification.

In any case, a Sovrin client MUST NOT ever rely on an abbreviated key by itself. The client MUST always check with the Sovrin ledger to ensure that the client has the most current DDO associated with a Sovrin DID.

A.3. Sovrin DID Operations

A.3.1. Create

[TODO]

A.3.2. Read/Verify

[TODO]

A.3.3. Update

[TODO]

A.3.4. Delete/Revoke

[TODO]

Appendix B: Aliases—Mapping Human-Friendly Identifiers to DIDs

DIDs achieve global uniqueness without the need for a central registration authority. However, as Zooko's Triangle illustrates, this comes at the cost of human memorability. In some use cases, it is desirable to be able to discover the DID record for an identity owner from a conventional address for the owner, such as a mobile telephone number, an email address, a Twitter handle, a URI, etc.

An **alias** is a DID created from a hash of a conventional address. If a specific DID scheme supports aliases, the DID method specification should specify: a) the target identifier scheme, b) the normalization algorithm, and c) the hashing algorithm required to produce the alias. Also, because the conventional address is registered with a centralized registration authority, an identity owner should prove ownership of an alias prior to registration. The governance rules for this requirement should be specified in the DID method specification.

Aliases represent a tradeoff between convenience and privacy. When a DID record is indexed by one or more aliases, it enables discovery by parties who know the conventional address. Because it is a hash, it does not reveal the conventional address directly, however this is relatively weak privacy protection because an alias can be easily discovered via dictionary attacks. Therefore if a DID method specification supports aliases:

1. An alias should only be used to index a DID record with the permission of the identity owner.
2. Lookups on an alias should be subject to access controls that meet the identity owner's privacy requirements.
3. Alternately, an additional password or passkey should be combined with the conventional address prior to hashing to allow the identity owner to control who can generate the alias and thus use it to discover the associated DID.