DID (Decentralized Identifier) Data Model and Generic Syntax 1.0

Editors: Drummond Reed, Les Chasen, Christopher Allen, Ryan Grant
Contributors: Manu Sporny, David Longley, Jason Law, Daniel Hardman, Markus Sabadello, Christian Lundkvist, Jonathan Endersby

STATUS: Implementer's Draft 01, 21 November 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. Specifically, a DDO contains a set of **key descriptions**—machine-readable descriptions of the identity owner's public keys—and a set of **service endpoints**—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 specific distributed ledger or network.

Table of Contents

1. Introduction
    1.1 Overview
    1.2. URIs, URLs, and URNs
    1.3. Motivations for DIDs
    1.4 The Role of Human-Friendly Identifiers
    1.4. Purpose of This Specification

2. Example DIDs and DDOs
    2.1. Example Owner-Managed DDO
    2.2. Example Guardian-Managed DDO

3. Terminology and Acronyms

4. Design Goals & Principles

5. DIDs (Decentralized Identifiers)
    5.1. The Generic DID Scheme
    5.2. Specific DID Method Schemes
    5.3 DID Paths
    5.4 DID Fragments
    5.5 DID Normalization
    5.6 DID Persistence

6. DDOs (DID Descriptor Objects)
    6.1. Context (Required)
    6.2. Primary DID (Required)
    6.3. Guardian (Required If No Proof of Ownership)
    6.4. Proof of Ownership (Required If No Guardian)
    6.5. Proof of Control (Optional and Method-Specific)
        6.5.1 "Or" Control
        6.5.2 "And" Control
        6.5.3 "M-of-N" Control
    6.6. Service Endpoint References (Optional)
    6.7. Created (Optional)
    6.8. Updated (Optional)
    6.9. Signature (Optional)

7. DID Operations
    7.1. Create
    7.2. Read/Verify
    7.3. Update

7.4. Delete/Revoke

8. DID Resolvers

9. Security Considerations
9.1. Requirements of DID Method Specifications
9.2 Binding of Identity
9.2.1 Proving Ownership of a DID and DDO
9.2.2 Proving Ownership of a Public Key
9.2.3 Identity Owner Authentication and Verifiable Claims
9.3 Authentication Service Endpoints
9.4 Non-Repudiation
9.5 Notification of DDO Changes
9.6 Key and Signature Expiration
9.7 Key Revocation and Recovery

10. Privacy Considerations
10.1 Requirements of DID Method Specifications
10.2 Keep Personally-Identifiable Information (PII) Off-Ledger
10.3 DID Correlation Risks and Pseudonymous DIDs
10.4 DDO Correlation Risks
10.5 Herd Privacy

11. Future Work
11.1 Upper Limits on DID character length
11.2 Equivalence
11.3 Timestamps
11.7 Time Locks and DDO Recovery
11.4 Smart Signatures
11.5 Verifiable Claims
11.6 Alternate Serializations and Graph Models

12. References

Appendix A: Proposed DID Method Specifications

Appendix B: The Generic DID Context for JSON-LD

Appendix C: Standard Key Descriptions
RSA Keys
EdDSA Keys

1. Introduction

1.1 Overview

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**. In this ecosystem, 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).

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 control of an identity owner (in the case of an owner-managed identity) or a **guardian** (in the case of a guardian-managed identity). A corresponding public key is published in the DDO using a **key description**. A DDO may also contain a set of **service endpoints** for interacting with the identity owner. 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.

To use a DID with a particular distributed ledger or network requires defining a **DID method** in a separate **DID method specification**. A DID method specifies the set of rules for how a DID is registered, resolved, updated, and revoked on that specific ledger or network.

This design eliminates dependence on centralized registries for identifiers as well as centralized certificate authorities for key management—the standard pattern in hierarchical PKI (public key infrastructure). With DID records on a distributed ledger, each identity owner may serve as its own root authority—an architecture referred to as DPKI (decentralized PKI).

Note that DID methods may also be developed for identities registered in federated identity management systems. For their part, federated identity systems may add support for DIDs. This creates an interoperability bridge between the worlds of centralized and decentralized identity.

1.2. URIs, URLs, and URNs

In September 2001, the W3C issued a note clarifying the terms **URI** (Uniform Resource Identifier), **URL** (Uniform Resource Locator), and **URN** (Uniform Resource Name). The key difference between these three categories of identifiers are:

1. **URI** is the term for any type of identifier used to identify a resource on the Web.
2. **URL** is the term for any type of URI that can be resolved or de-referenced to locate a representation of a resource on the Web (e.g., Web page, file, image, etc.)
3. **URN** is the term for a specific type of URI intended to persistently identify a resource, i.e., an identifier that will never change no matter how often the resource moves, changes names, changes owners, etc. URNs are intended to last forever.

1.3. Motivations for DIDs

The growing need for decentralized identity has produced three specific requirements for a new type of URI that fits within the URI/URL/URN architecture, albeit in a less than traditional way:

1. **A URI that is persistent like a URN yet can be resolved or de-referenced to locate a resource like a URL.** In essence, a DID is a URI that serves both functions.
2. **A URI that does not require a centralized authority to register, resolve, update, or revoke.** The overwhelming majority of URIs today are based on DNS names or IP addresses that depend on centralized authorities for registration and ultimate control. DIDs can be created and managed without any such authority.
3. **A URI whose ownership and associated metadata, including public keys, can be cryptographically verified.** Control of DIDs and DDOs leverages the same public/private key cryptography as distributed ledgers.

1.4 The Role of Human-Friendly Identifiers

DIDs achieve global uniqueness without the need for a central registration authority. This comes, however, at the cost of human memorability. The algorithms capable of generating globally unique identifiers automatically produce random strings of characters that have no human meaning. This leads to the axiom about identifiers known as Zooko's Triangle: "human-meaningful, decentralized, secure—pick any two".

There are of course many use cases where it is desirable to discover a DID starting from a human-friendly identifier—a natural language name, a domain name, or a conventional address for a DID owner, such as a mobile telephone number, email address, Twitter handle, blog URL, etc. However, the problem of mapping human-friendly identifiers to DIDs (and doing so in a way that can be verified and trusted) is out-of-scope for this specification.

Solutions to this problem (and there are many) should be defined in separate specifications that reference this specification. It is strongly recommended that such specifications carefully consider: (a) the numerous security attacks based on deceiving users about the true human-friendly identifier for a target entity, and (b) the privacy consequences of using human-friendly identifiers that are inherently correlatable, especially if they are globally unique.

1.4. Purpose of This Specification

The first purpose of this specification is to define the generic DID scheme and a generic set of operations on DID records that can be implemented for any distributed ledger or network capable of accepting DID records. The second purpose of this specification to define the conformance requirements for a **DID method specification**—a separate specification that defines a specific DID scheme and specific set of DID record operations for a specific distributed ledger or network.

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 (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 difference is that a DID method specification, in addition to defining a specific DID scheme, must also specify the methods for reading, writing, and revoking DID records on the network for which it is written.

For a list of DID method specifications, see Appendix A.

2. Example DIDs and DDOs

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

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

2.1. Example Owner-Managed DDO

Following is an example of a DDO that describes the DID above. This example assumes that the **identity owner**—the entity that controls the private keys for this identity—is authoritative for the DDO. See section 2.2 for an example of a DDO created by a **guardian**. For the authoritative JSON-LD context definition, see Appendix B (the URL for the `@context` property below is just for illustration).

```
{
      "@context": "https://example.org/did/v1",
      "id": "did:sov:21tDAKCERh95uGgKbJNHYp",
      "owner": [{
            "id": "did:sov:21tDAKCERh95uGgKbJNHYp#key-1",
            "type": ["CryptographicKey", "EdDsaPublicKey"],
            "curve": "ed25519",
            "expires": "2017-02-08T16:02:20Z",
            "publicKeyBase64":
      "lji9qTtkCydxtez/bt1zdLxVMMbz4SzWvlqgOBmURoM="
           }, {
            "id": "did:sov:21tDAKCERh95uGgKbJNHYp#key-2",
            "type": ["CryptographicKey", "RsaPublicKey"],
            "expires": "2017-03-22T00:00:00Z",
            "publicKeyPem": "----BEGIN PUBLIC
      KEY-----\r\nMIIBOgIBAAJBAKkbSUT9/Q2uBfGRau6/XJyZhcF5abo7b37I5hr
      3EmwGykdzyk8GSyJK3TOrjyl0sdJsGbFmgQaRyV\r\n-----END PUBLIC
      KEY-----"
      }],
      "control": [{
            "type": "OrControl",
            "signer": [
                  "did:sov:21tDAKCERh95uGgKbJNHYp",
                  "did:sov:8uQhQMGzWxR8vw5P3UWH1j"
            ]
      }],
      "service": {
            "openid": "https://openid.example.com/456",
            "xdi": "https://xdi.example.com/123"
      },
      "created": "2002-10-10T17:00:00Z",
      "updated": "2016-10-17T02:41:00Z",
      "signature": {
            "type": "RsaSignature2016",
            "created": "2016-02-08T16:02:20Z",
            "creator": "did:sov:8uQhQMGzWxR8vw5P3UWH1j#key/1",
            "signatureValue":
"IOmA4R7TfhkYTYW87z640O3GYFldw0yqie9Wl1kZ5OBYNAKOwG5uOsPRK8/2C4STOWF+
83cMcbZ3CBMq2/gi25s="
      }
}
```

2.2. Example Guardian-Managed DDO

Following is a second example of a DDO that describes the DID above. In this case the DDO describes a **dependent**—an entity who is not currently in a position to control the private keys for this identity. This DDO was created by a **guardian**—a separate identity owner with its own DID that serves as a trustee for the dependent. Note that while this DDO asserts a set of service endpoints, it does not yet contain a set of key descriptions because the dependent does not yet have its own set of private keys.


```
{
      "@context": "https://example.org/did/v1",
      "id": "did:sov:21tDAKCERh95uGgKbJNHYp",
      "guardian": "did:sov:8uQhQMGzWxR8vw5P3UWH1j"
      "control": [ "did:sov:8uQhQMGzWxR8vw5P3UWH1j" ],
      "service": {
            "openid": "https://openid.example.com/456",
            "xdi": "https://xdi.example.com/123"
      },
      "type": "http://schema.org/Person",
      "created": "2002-10-10T17:00:00Z",
      "updated": "2016-10-17T02:41:00Z",
      "signature": {
            "type": "RsaSignature2016",
            "created": "2016-02-08T16:02:20Z",
            "creator": "did:sov:8uQhQMGzWxR8vw5P3UWH1j#key-1",
            "signatureValue":
"IOmA4R7TfhkYTYW87z640O3GYFldw0yqie9Wl1kZ5OBYNAKOwG5uOsPRK8/2C4STOWF+
83cMcbZ3CBMq2/gi25s="
      }
}
```

3. Terminology and Acronyms

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.

All other terms used in this specification are defined in this glossary.

Blockchain. A specific type of **distributed ledger technology** (DLT) that stores ledger entries in blocks of transactions that are grouped together and hashed into a cryptographic chain. Because this type of DLT was introduced by Bitcoin, the term “blockchain” is sometimes used to refer specifically to the Bitcoin ledger.

Decentralized identifier (DID). A globally unique identifier that does not require a centralized registration authority because it is registered with **distributed ledger technology** or other form of decentralized network. The generic format of a DID is defined in this specification. A specific **DID scheme** is defined in a **DID method specification**.

Decentralized identity management (DIDM). Identity management based on decentralized identifiers. DIDM extends identifier creation authority beyond the traditional roots of trust required by X.500 directory services, the Domain Name System, national ID systems, etc.

Decentralized PKI (DPKI). Public key infrastructure based on decentralized identifiers and identity records (e.g., **DDOs**) containing verifiable public key descriptions.

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

Dependent. A person, organization, or thing whose **DID record** is registered and maintained by a **guardian** because the dependent is not in a position to control the private keys. A dependent becomes an **identity owner** when the dependent takes control of the private keys.

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 fragment. The portion of a **DID reference** that follows the first hash sign character (“#”). A DID fragment uses the same syntax of a **URI** fragment. See section 5.5. Note that a **DID fragment** MUST immediately follow a DID. If a DID reference includes a **DID path** followed by a fragment, that fragment is NOT a **DID fragment**.

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

DID method conformance profile. A specification (or section of a specification) that defines the rules an implementer must follow in order to conform with another DID method specification. A conformance profile narrows the set of options that may be used in order to meet a specific implementation goal.

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

DID path. The portion of a **DID reference** that follows the first forward slash character. A DID path uses the identical syntax of a **URI** path. See section 5.4. Note that if a DID path is followed by a fragment, that fragment is NOT a **DID fragment**.

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

DID reference. A **DID** plus an optional **DID path** or **DID fragment**.

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

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**.

Guardian. An identity owner who creates a **DID record** for a **dependent** who does not yet have the capacity to control the private keys. The dependent must rely on the guardian to safeguard the private keys until the dependent can assume control and become an **identity owner**.

Identity owner. The natural person, party, organization, or thing whose identity is represented by a **DID record** and who directly controls the private keys to control that record. (Note: this specification avoids the term "user" since an identity owner is not always an individual person.)

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

JSON-LD (JSON Linked Data). A method of encoding Linked Data using JSON. The W3C JSON-LD 1.0 specification enables object properties in a JSON document to be linked to concepts in an **RDF** ontology.

Key description. A JSON object contained inside a **DDO** that contains all the metadata necessary to use a public key or verification key. A list of standard key descriptions is included in Appendix C.

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

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

RDF (Resource Description Framework). A semantic graph model defined by the W3C RDF Working Group.

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 official name for all Web identifiers as defined by IETF RFC 3986. See section 1.2.

URL (Uniform Resource Locator). Any **URI** that can be resolved or de-referenced to locate a specific resource on the Web. See section 1.2.

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. In general URNs cannot directly be resolved or de-referenced to locate the resource they identify. See section 1.2.

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

4. Design Goals & Principles

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

#	Principle	Description
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.
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.
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.
4	Security	DID architecture should provide sufficient security for relying parties to rely on DID records to their required level of assurance.
5	Cryptography	DID architecture should enable an identity owner to provide cryptographic proof of ownership and proof of access control rights.
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.
7	Interoperability	DID architecture should use interoperable standards so DIDs and DID record infrastructure can make use of existing tools and software libraries designed for interoperability.
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.
9	Simplicity	To meet these design goals, DID architecture should be (to paraphrase Albert Einstein) “as simple as possible but no simpler”.
10	Extensibility	When possible, DID architecture should enable extensibility provided it does not hinder interoperability, portability or simplicity.

5. DIDs (Decentralized Identifiers)

The foundation of DID architecture is the concept of the decentralized identifier. This concept is not new; UUIDs (Universally Unique IDentifiers) were 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 namespace specified in IETF RFC 4122.

A DID is similar to a UUID except: (a) it can be resolved or dereferenced to a standard resource describing the identity owner (a DDO—see section 6), and (b) the DDO may contain public key descriptions that enable cryptographic verification of DID ownership.

5.1. The Generic DID Scheme

The generic DID scheme is a URI scheme conformant with RFC 3986. It consists of a DID followed by an optional path and/or fragment. The term **DID** refers only to the identifier conforming to the `did` rule in the ABNF below—when used alone, it does not include a path or fragment. A DID that may optionally include a path and/or fragment is called a **DID reference**.

Following is the ABNF definition using the syntax in RFC 5234 (which defines ALPHA as upper or lowercase A-Z).

```
did-reference       = did [ "/" did-path ] [ "#" did-fragment ]
did                 = "did:" method ":" specific-idstring
method              = 1*methodchar
methodchar          = %x61-7A / DIGIT
specific-idstring   = idstring *( ":" idstring )
idstring            = 1*idchar
idchar              = ALPHA / DIGIT / "." / "-"
```

See sections 5.3 and 5.4 for the ABNF rules defining DID paths and fragments.

5.2. Specific DID Method Schemes

A DID method specification MUST define exactly one specific DID scheme identified by exactly one method name (the `method` rule in section 5.1). 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.

A list of known DID method names is included in Appendix A.

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

The DID method specification for the specific DID scheme MUST specify how to generate the `specific-idstring` component of a DID. The `specific-idstring` value MUST be able to be generated without the use of a centralized registry service. The `specific-idstring` value SHOULD be globally unique by itself. The fully qualified DID as defined by the `did` rule in section 5.1 MUST be globally unique.

If needed, a specific DID scheme MAY define multiple specific `specific-idstring` formats. It is RECOMMENDED that a specific DID scheme define as few `specific-idstring` formats as possible.

5.3 DID Paths

A generic DID path (the `did-path` rule in section 5.1) is identical to a URI path and MUST conform to the ABNF of the `path-rootless` ABNF rule in RFC 3986. A DID path SHOULD be used to address resources available via a DID service endpoint. See section 6.6.

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

5.4 DID Fragments

A generic DID fragment (the `did-fragment` rule in section 5.1) is identical to a URI fragment and MUST conform to the ABNF of the `fragment` ABNF rule in RFC 3986. A DID fragment MUST be used only as a method-independent pointer into the DDO to identify a unique key description or other DDO component. To resolve this pointer, the complete DID reference including the DID fragment MUST be used as the value of the `id` key for the target JSON object.

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

5.5 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 the value of the `specific-idstring` rule in section 5.1 MUST be defined by the governing DID method specification.

5.6 DID Persistence

A DID MUST be persistent and immutable, i.e., bound to an identity owner once and never changed (forever). Ideally a DID would be a completely abstract decentralized identifier (like a UUID) that could be bound to multiple underlying distributed ledgers or 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 developers.

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.

NOTE: Although not included in this version, future versions of this specification may support a DDO `equivID` property to establish verifiable equivalence relations between DID records representing the same identity owner on multiple ledgers or networks. Such equivalence relations can produce the practical equivalent of a single persistent abstract DID. See Future Work (section 11).

6. DDOs (DID Descriptor Objects)

If a DID is the index key in a key-value pair, then the DDO is the value to which the index key points. The combination of a DID and its associated DDO forms the root **identity record** for a decentralized identity.

A DDO MUST be 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.

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

6.1. 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 required for DDOs.)
3. The key for this property MUST be `@context`.
4. The value of this key MUST be the URL for the generic DID context as specified in Appendix B.

Example (using a example URL):

```
{
      "@context": "https://example.org/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 implement the method. Method-specific contexts MUST NOT override the terms defined in the generic DID context listed in Appendix B.

6.2. Primary DID (Required)

The primary DID is the primary index key for the DDO, i.e., it is DID described by DDO. The rules for a primary DID are:

1. A DDO MUST have exactly one primary DID.
2. The key for this property MUST be `id`.
3. The value of this key MUST be a valid DID.
4. When this DDO is registered with the target distributed ledger or network, the registered DID MUST match this primary DID value.

Example:

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

6.3. Guardian (Required If No Proof of Ownership)

A **guardian** is an identity owner who creates and maintains an identity record for a **dependent** who is not in a position to hold or control the necessary cryptographic keys (e.g., a parent creating an identity record for a child). In this case, there are no owner keys to represent the ultimate identity owner. So the DDO needs to assert the identity of the guardian.

The rules for a guardian are:

1. A DDO that includes an `owner` (section 6.4) MAY have a guardian.
2. A DDO that does not include an `owner` MUST have a guardian.
3. The key for this property MUST be `guardian`.
4. The value of this key MUST be a valid DID.

Example:

```
{
      "guardian": "did:sov:8uQhQMGzWxR8vw5P3UWH1j"
}
```

6.4. Proof of Ownership (Required If No Guardian)

Proof of Ownership is the mechanism by which an identity owner can cryptographically prove ownership of a DID and DDO by virtue of publishing a set of public key or verification key descriptions. See section 9.2. 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 as discussed in section 6.5) without enabling them to prove ownership (and thus be able to impersonate the identity owner).

The rules for Proof of Ownership are:

1. A DDO that includes an `guardian` (section 6.3) MAY have exactly one owner.
2. A DDO that does not include an `guardian` MUST have exactly one owner.
3. The key for this property MUST be `owner`.
4. The value of this key MUST be an array where each member of the array is a key description of a valid public key or verification key. A list of standard key descriptions is included in Appendix C. A new key description MAY also be defined by a DID method specification.
5. If this array contains more than one key description, each key description MUST include a property with the key `id` whose value is a DID reference consisting of the primary DID and a DID fragment. This DID reference uniquely identifies this key description.

Example:

```
{
      "owner": [{
            "id": "did:sov:21tDAKCERh95uGgKbJNHYp#key/1",
            "type": ["CryptographicKey", "EdDsaSAPublicKey"],
            "curve": "ed25519",
            "expires": "2017-02-08T16:02:20Z",
            "publicKeyBase64":
"IOmA4R7TfhkYTYW87z640O3GYFldw0yqie9Wl1kZ5OBYNAKOwG5uOsPRK8/2C4STOWF+
83cMcbZ3CBMq2/gi25s="
```

```
      }, {
            "id": "did:sov:21tDAKCERh95uGgKbJNHYp#key/2",
            "type": ["CryptographicKey", "RsaPublicKey"],
            "expires": "2017-03-22T00:00:00Z",
            "publicKeyBase64":
"MIIBOgIBAAJBAKkbSUT9/Q2uBfGRau6/XJyZhcF5abo7b37I5hr3EmwGykdzyk8GSyJK
3TOrjyl0sdJsGbFmgQaRyV"
      }]
}
```

Note that caching and expiration of the keys in DDO key descriptions is entirely the responsibility of DID resolvers and other clients. See section 9.6.

6.5. Proof of Control (Optional and Method-Specific)

Proof of Control is the mechanism by which an identity owner may give itself or other entities permission to update the DDO—for example to assist with key recovery. Note that Proof of Control is separate from Proof of Ownership as explained in section 6.4. This is particularly important for key recovery in the case of key loss, when the identity owner no longer has access to the keys described using the `owner` property (section 6.4), or key compromise, where the owner's trusted third parties need to override malicious activity by the attacker. See section 9.

Because the access control logic in a Proof of Control block must be implemented by the target distributed ledger or network, a DID method specification MUST include its own Proof of Control rules and processing logic. It is RECOMMENDED that all DID method specifications support the generic Proof of Control rules specified in this section. A DID method specification MAY add its own method-specific Proof of Control rules.

The generic Proof of Control rules are:

1. A DDO MAY have exactly one property representing Proof of Control.
2. The key for this property MUST be `control`.
3. The value of this key MUST be an array.
4. The values of this array MUST be defined in a DID method specification.
5. If a Proof of Control rule accepts a DID without a fragment as a value, then an update signature verified with any key in the `owner` of the DDO dereferenced from that DID MUST be considered valid for Proof of Control.
6. If a Proof of Control rule accepts a DID with a fragment as a value, then:
    a. The fragment MUST identify a key description in the DDO dereferenced from that DID.
    b. An update signature verified with the key in target key description MUST be considered valid for Proof of Control.

6.5.1 "Or" Control

To assert that any single member of a group of other DID owners has permission to update the DDO, the control block array MAY contain a single JSON object with two properties:

1. A property `type` whose value is the string `OrControl`.
2. A property `signers` whose value is an array of DIDs with or without fragments.

If at least one update signature from a DID in this array is verified, it MUST be considered valid for Proof of Control.

6.5.2 "And" Control

To assert that only all members of a group of other DID owners must act together to update the DDO, the control block array MAY contain a single JSON object with two properties:

1. A property `type` whose value is the string `AndControl`.
2. A property `signers` whose value is an array of DIDs with or without fragments.

If update signatures from ALL DIDs in this array are verified, it MUST be considered valid for Proof of Control.

6.5.3 "M-of-N" Control

To assert that a minimum number of members of a group of other DID owners must act together to update the DDO, the control block array MAY contain a single JSON object with three properties:

1. A property `type` whose value is the string `MofNControl`.
2. A property `minimumSignatures` whose value is an integer representing the minimum threshold of signatures required to act together.
3. A property `signers` whose value is an array of DIDs with or without fragments.

If the number of verified update signatures from DIDs in this array equals or exceeds the value of `minimumSignature s`, it MUST be considered valid for Proof of Control.

Following is an example of a Proof of Control property implementing these rules:

```
{
      "control": [{
            "type": "OrControl",
            "signer": [
                  "did:sov:21tDAKCERh95uGgKbJNHYp",
                  "did:sov:8uQhQMGzWxR8vw5P3UWH1j"
            ]
      }, {
```

```
            "type": "AndControl",
            "signer": [
                "did:sov:7P7xfv5AeTSSWcuq6hPptQ",
                "did:sov:X2v8rvzoCxayhYV5mhESQ1"
            ]
      }, {
            "type": "MofNControl",
            "minimumSignatures": 2,
            "signer": [
                "did:sov:4XirzuHiNnTrwfjCMtBEJ6",
                "did:sov:iCGjJEChRbAdfZbGqZAYT7",
                "did:sov:WUoAyXB7mhfcVESjypm5ty"
            ]
      }]
}
```

6.6. Service Endpoint References (Optional)

In addition to 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 decentralized identity management services for further discovery, authentication, authorization, or interaction.

The rules for service endpoints are:

1. A DDO MAY have exactly one property representing service endpoints.
2. The key for this property MUST be `service`.
3. The value of this key MUST be a nested JSON object.
4. In this nested JSON object, each property MUST have a key representing the name of a service established in either the generic JSON-LD context (see Appendix B) or a method-specific context definition.
5. The value of this key MUST be a valid URI conforming to RFC 3986 and normalized according to the rules in section 6 of RFC 3986 and to any normalization rules in its applicable URI scheme specification.

Example:

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

See sections 9.1 and 9.3 for further security considerations regarding authentication service endpoints.

6.7. 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 property representing a creation timestamp. It is RECOMMENDED to include this property.
2. The key for this property MUST be `created`.
3. 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.
4. This datetime value MUST be normalized to UTC 00:00 as indicated by the trailing “Z”.
5. Method specifications that rely on DLTs SHOULD require time values that are after the known "median time past" (defined in Bitcoin BIP 113), when the DLT supports such a notion.

Example:

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

6.8. 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 property representing an updated timestamp. It is RECOMMENDED to include this property.
2. The key for this property MUST be `updated`.
3. The value of this key MUST follow the formatting rules (3, 4, 5) from section 6.7.

Example:

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

6.9. Signature (Optional)

A signature on a DDO is cryptographic proof of the integrity of the DDO according to either:

1. The identity owner as defined in section 6.4, or if not present:
2. The guardian as defined in section 6.3.

This signature is NOT proof of the binding between a DID and a DDO. See section 9.2.

The rules for a signature are:

1. A DDO MAY have exactly one property representing a signature.
2. The key for this property MUST be `signature`.
3. The value of this key MUST be a valid JSON-LD signature as defined by [Linked Data Signatures](#).

Example:


```
{
      "signature": {
            "type": "LinkedDataSignature2015",
            "created": "2016-02-08T16:02:20Z",
            "creator": "did:sov:8uQhQMGzWxR8vw5P3UWH1ja#keys-1",
            "signatureValue":
"QNB13Y7Q9oLlDLL6AHyL31OE5fLji9DwJSA8qnv81oRaKonij8m+Jv4XdiEYvJ97iRlz
KU/92/0LafSL5JftEgl960DLcbqMFxOtbAmFOIMa7eDcrgTL5ytXeYCYKLjHQG3s8a3UK
DKRuEK54qK1G5hGKGoLgAVa6xgcDLjW7M19PEJV/c3HpGA7Eez6VFMoTt4yESjZvOXC97
xN3KpshOx2HT/btgUbo0XjA1Oi0QHdgrLcUsQGt6w23RjeSToalrsA1G69OFeN2OiQrz9
Jb4561hvKLSyWObwRmS6n5Vgr5xkvUm6MONRq0Vg33kXevoVM64KTBkISul61tzjn4w==
"
      }
}
```

7. DID Operations

To enable the full functionality of DIDs and DDOs on a particular distributed ledger or 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

7.1. 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.

7.2. 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 response.

7.3. 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.

7.4. 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 revocation.

8. DID Resolvers

A DID resolver is a software component with an API designed to accept requests for DID lookups and execute the corresponding DID method to retrieve the authoritative DDO. To be conformant with this specification, a DID resolver:

1. SHOULD validate that a DID is valid according to its DID method specification, otherwise it should produce an error.
2. MUST conform to the requirements of the applicable DID method specification when performing DID resolution operations.
3. SHOULD offer the service of verifying the integrity of the DDO if it is signed.
4. MAY offer the service of returning requested properties of the DDO.

9. Security Considerations

NOTE TO IMPLEMENTERS: During the Implementer's Draft stage, this section focuses on security topics that should be important in early implementations. The editors are also seeking feedback on threats and threat mitigations that should be reflected in this section or elsewhere in the spec.

As the root identity records for decentralized identities, DIDs and DDOs are a vital component of decentralized identity management. They are also the foundational building blocks of DPKI (decentralized public key infrastructure) as an augmentation to conventional X.509 certificates.

As such, DIDs are designed to operate under the general Internet threat model used by many IETF standards. We assume uncompromised endpoints, but allow messages to be read or corrupted on the network. Protecting against an attack when a system is compromised requires external key-signing hardware. See also section 6.5 regarding key revocation and recovery.

For their part, the DLTs hosting DIDs and DDOs have special security properties for preventing active attacks. Their design uses public/private key cryptography to allow operation on passively monitored networks without risking compromise of private keys. This is what makes DID architecture and decentralized identity possible.

9.1. Requirements of DID Method Specifications

1. DID method specifications MUST include their own Security Considerations sections.
2. This section MUST consider all the requirements mentioned in section 5 of RFC 3552 (page 27) for the DID operations defined in the specification. In particular:

   At least the following forms of attack MUST be considered: eavesdropping, replay, message insertion, deletion, modification, and man-in-the-middle. Potential denial of

service attacks MUST be identified as well. If the protocol incorporates cryptographic protection mechanisms, it should be clearly indicated which portions of the data are protected and what the protections are (i.e., integrity only, confidentiality, and/or endpoint authentication, etc.). Some indication should also be given to what sorts of attacks the cryptographic protection is susceptible. Data which should be held secret (keying material, random seeds, etc.) should be clearly labeled. If the technology involves authentication, particularly user-host authentication, the security of the authentication method MUST be clearly specified.

3. This section MUST also discuss, per Section 5 of RFC 3552, residual risks (such as from compromise in a related protocol, incorrect implementation, or cipher) after threat mitigation has been deployed.
4. This section MUST provide integrity protection and update authentication for all operations required by Section 7 of this specification (DID Operations).
5. Where DID methods make use of peer-to-peer computing resources (such as with all known DLTs), the expected burdens of those resources SHOULD be discussed in relation to denial of service.
6. Method-specific endpoint authentication MUST be discussed. Where DID methods make use of DLTs with varying network topology, sometimes offered as “light node” or “thin client” implementations to reduce required computing resources, the security assumptions of the topology available to implementations of the DID method MUST be discussed.
7. DID methods MUST discuss the policy mechanism by which DIDs are proven to be uniquely assigned. A DID fits the functional definition of a URN as defined in RFC 2141—a persistent identifier that is assigned once to a resource and never reassigned. In a security context this is particularly important since a DID may be used to identify a specific party subject to a specific set of authorization rights.
8. DID methods that introduce new authentication service endpoint types (section 6.6) SHOULD consider the security requirements of the supported authentication protocol.

9.2 Binding of Identity

9.2.1 Proving Ownership of a DID and DDO

By itself, a verified signature on self-signed DDO does not prove ownership of a DID. It only proves the following:

1. The DDO has not been tampered with since it was registered.
2. The owner of the DDO controlled the private key used for the signature at the time the signature was generated.

Proving ownership of a DID, i.e., the binding between the DID and the DDO that describes it, requires a two step process:

1. Resolving the DID to a DDO according to its DID method specification.
2. Verifying that the `id` property of the resulting DDO matches the DID that was resolved.

It should be noted that this process proves ownership of a DID and DDO *regardless of whether the DDO is signed*.

9.2.2 Proving Ownership of a Public Key

There are two methods for proving ownership of the private key corresponding to a public key description in the DDO: *static* and *dynamic*.

The static method is to sign the DDO with the private key. This proves ownership of the private key at a time no later than the DDO was registered.

If the DDO is not signed, ownership of a public key described in the DDO may still be proven dynamically as follows:

1. Send a challenge message containing a public key description from the DDO and a nonce to an appropriate service endpoint described in the DDO.
2. Verify the signature of the response message against the public key description.

9.2.3 Identity Owner Authentication and Verifiable Claims

A DID and DDO do not inherently carry any PII (personally-identifiable information). The process of binding a DID to the real-world owner of an identity using claims about the owner is of out is of scope for this specification. However this topic is the focus of the verifiable claims standardization work at the W3C (where the term “DID” originated).

9.3 Authentication Service Endpoints

If a DDO publishes a service endpoint intended for authentication or authorization of an identity owner (section 6.6), it is the responsibility of the service endpoint provider, identity owner, and/or relying party to comply with the requirements of the authentication protocol(s) supported at that service endpoint.

9.4 Non-Repudiation

Non-repudiation of DIDs and DDO updates is supported under the assumption that: (1) the identity owner is monitoring for unauthorized updates (see section 9.5) and (2) the identity owner has had adequate opportunity to revoke malicious updates according to the DID method's access control mechanism (section 6.5). This capability is further supported if timestamps are included (sections 6.7 and 6.8) and the target DLT system supports timestamps.

9.5 Notification of DDO Changes

One mitigation against unauthorized changes to a DDO is monitoring and actively notifying the identity owner when there are changes. This is analogous to helping prevent account takeover on conventional username/password accounts by sending password reset notifications to the email addresses on file.

In the case of a DID, where there is no intermediary registrar or account provider to generate the notification, the following approaches are RECOMMENDED:

1. **Subscriptions.** If the ledger or network on which the DID is registered directly supports change notifications, this service can be offered to DID owners. Notifications may be sent directly to the relevant service endpoints listed in an existing DID.
2. **Self-monitoring.** An identity owner may employ its own local or online agent to periodically monitor for changes to a DDO.
3. **Third-party monitoring.** An identity owner may rely on a third party monitoring service, however this introduces another vector of attack.

9.6 Key and Signature Expiration

In decentralized identity architecture, there are no centralized authorities to enforce key or signature expiration policies. Therefore DID resolvers and other client applications SHOULD validate that keys have not expired. Since some use cases may have legitimate reasons why already-expired keys can be extended, a key expiration SHOULD NOT prevent any further use of the key, and implementations SHOULD attempt to update its status upon encountering it in a signature.

9.7 Key Revocation and Recovery

Section 7 specifies the DID operations that must be supported by a DID method specification, including revocation of a DDO by replacing it with an updated DDO. In general, checking for key revocation on DLT-based methods is expected to be handled in a manner similar to checking the balance of a cryptocurrency account on a distributed ledger: if the balance is empty, the entire DID is revoked.

DID method specifications SHOULD enable support for a quorum of trusted parties to enable key recovery. Some of the facilities to do so are suggested in section 6.5, Proof of Control. Note that not all DID method specifications will recognize control from DIDs registered using other DID methods and they MAY restrict third-party control to DIDs that use the same method.

Access control and key recovery in a DID method specification MAY also include a time lock feature to protect against key compromise by maintaining a second track of control for recovery. Further specification of this type of control is a matter for future work (see section 11.7).

10. Privacy Considerations

It is critically important to apply the principles of Privacy by Design to all aspects of decentralized identity architecture, because DIDs and DDOs are—by design—administered directly by their owners. There is no registrar, hosting company, or other intermediate service provider to recommend or apply additional privacy safeguards.

The authors of this specification have applied all seven Privacy by Design principles throughout its development. For example, privacy in this specification is preventative not remedial, and privacy is an embedded default. Furthermore, decentralized identity architecture by itself embodies principle #7, “Respect for user privacy—keep it user-centric.”

This section lists additional privacy considerations that implementers, guardians, and identity owners should bear in mind.

10.1 Requirements of DID Method Specifications

1. DID method specifications MUST include their own Privacy Considerations sections, if only to point to the general privacy considerations in this section.
2. The DID method privacy section MUST discuss any subsection of section 5 of RFC 6973 that could apply in a method-specific manner. The subsections to consider are: *surveillance, stored data compromise, unsolicited traffic, misattribution, correlation, identification, secondary use, disclosure, exclusion*.

10.2 Keep Personally-Identifiable Information (PII) Off-Ledger

If a DID method specification is written for a public ledger or network where all DIDs and DDOs will be publicly available, it is STRONGLY RECOMMENDED that DDOs contain no PII. All PII should be kept off-ledger behind service endpoints under the control of the identity owner. With this privacy architecture, PII may be exchanged on a private, peer-to-peer basis using communications channels identified and secured by key descriptions in DID records. This also enables identity owners and relying parties to implement the GDPR right to be forgotten, as no PII will be written to an immutable ledger.

10.3 DID Correlation Risks and Pseudonymous DIDs

Like any type of globally unique identifier, DIDs may be used for correlation. Identity owners can mitigate this privacy risk by using *pairwise unique DIDs*, i.e., by sharing a different private DID

for every relationship. In effect, each DID acts as a pseudonym. A pseudonymous DID need only be shared with more than one party when the identity owner explicitly authorizes correlation between those parties.

If pseudonymous DIDs are the default, then the only need for a public DID—a DID published openly or shared with a large number of parties—is when the identity owner explicitly desires public identification.

10.4 DDO Correlation Risks

The anti-correlation protections of pseudonymous DIDs are easily defeated if the data in the corresponding DDOs can be correlated. For example, using same public key descriptions or bespoke service endpoints in multiple DDOs can provide as much correlation information as using the same DID. Therefore the DDO for a pseudonymous DID SHOULD also use pairwise-unique public keys and pairwise-unique service endpoints.

10.5 Herd Privacy

When an entity is indistinguishable from others in the herd, privacy is available. When the act of engaging privately with another party is by itself a recognizable flag, privacy is greatly diminished.

DIDs and DID methods SHOULD work to improve herd privacy, particularly for those who legitimately need it most. Choose technologies and human interfaces that default to preserving anonymity and pseudonymity. In order to reduce digital fingerprints, share common settings across client implementations, keep negotiated options to a minimum on wire protocols, use encrypted transport layers, and pad messages to standard lengths.

11. Future Work

11.1 Upper Limits on DID Character Length

The current specification does not take a position on maximum length of a DID. The maximum interoperable URL length is currently about 2K characters. QR codes can handle about 4K characters. Clients using DIDs will be responsible for storing many DIDs, and some methods would be able to externalize some of their costs onto clients by relying on more complicated signature schemes or by adding state into DIDs intended for temporary use. A future version of this specification should set reasonable limits on DID character length to minimize externalities.

11.2 Equivalence

Including an equivalence property, such as `equivID`, in DDOs whose value is an array of DIDs would allow identity owners to assert two or more DIDs that represent the same identity owner. This capability has numerous uses, including supporting migration between ledgers and providing forward compatibility of existing DIDs to future DLTs. In theory, equivalent DIDs should have the same identity rights, allowing verifiable claims made against one DID to apply to equivalent DIDs.

Equivalence was not included in the current specification due to the complexity of verifying equivalence across different DLTs and different DID methods, and also of aggregating properties of equivalent DDOs. However equivalence should be supported in a future version of this specification.

11.3 Timestamps

Verifiable timestamps have significant utility for identity records. This is a good fit for DLTs, since most offer some type of timestamp mechanism. Despite some transactional cost, they are the most censorship-resistant transaction ordering systems in the world, so they are nearly ideal for DDO timestamping. In some cases a DLT's immediate timing is approximate, however their sense of "median time past" (see Bitcoin BIP 113) can be precisely defined.

A generic DDO timestamping mechanism could would work across all DLTs and might operate via a mechanism including either individual transactions or transaction batches. The generic mechanism was was deemed out of scope for this version, although it may be included in a future version of this specification.

11.7 Time Locks and DDO Recovery

Section 9.7 mentions one possible clever use of time locks to recover control of a DID after a key compromise. The technique relies on an ability to override the most recent update to a DDO with Proof of Control applied by an earlier version of the DDO in order to defeat the attacker. This protection depends on adding a time lock (see Bitcoin BIP 65) to protect part of the transaction chain, enabling a Proof of Control block to be used to recover control. We plan to add support for time locks in a future version of this specification.

11.4 Smart Signatures

Not all DLTs can support the Proof of Control logic in section 6.5. Therefore, in this version of the specification, all Proof of Control logic must be delegated to DID method specifications. A potential future solution is a Smart Signature specification that specifies the code any conformant DLT may implement to process signature control logic.

11.5 Verifiable Claims

Although DIDs and DDOs form a foundation for decentralized identity, they are only the first step in describing an identity owner. The rest of the descriptive power comes through collecting and selectively using verifiable claims. Future versions of the specification will describe in more detail how DIDs and DDO can be integrated with—and help enable—the verifiable claims ecosystem.

11.6 Alternate Serializations and Graph Models

This version of the specification relies on JSON-LD and the RDF graph model for expressing a DDO. 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.

12. References

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

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

[LINKED-DATA-SIGNATURES] Draft Community Group Report https://w3c-dvcg.github.io/ld-signatures/

[RFC 3552] Guidelines for Writing RFC Text on Security Considerations. IETF RFC 3552. https://tools.ietf.org/html/rfc3552

[RFC 6973] Privacy Considerations for Internet Protocols. IETF RFC 6973. https://tools.ietf.org/html/rfc6973

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

[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

[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

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

[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: Proposed DID Method Specifications

This table summarizes the DID method specifications currently in development. The links will be updated as subsequent Implementer's Drafts are produced.

Method Name	DLT or Network	Authors	Link
did:sov:	Sovrin	Sovrin Foundation	https://docs.google.com/document/d/1X7dWpVvskGRpk05yyPEMd1uqaJ9FnOzoeWMdwzdIFyU/edit#
did:btc1:	Bitcoin	Christopher Allen	
did:uport	Ethereum	uPort	
did:cnsnt:	Ethereum	Consent	

Appendix B: The Generic DID Context for JSON-LD

This JSON-LD document is the generic context for all DDOs. See section 6.1 for the rules for using this context.

For this implementer's draft, the URL for this context is:

https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-fall2016/blob/master/final-documents/did-context-v1-draft-01.txt

```
{
      "@context": ["https://w3id.org/identity/v1", {
            "ctrl": "https://w3id.org/control#",
            "ddo": "https://w3id.org/ddo#",
            "control": {
                  "@id": "ddo:control",
                  "@type": "@id"
            },
            "curve": "sec:curve",
            "guardian": "ddo:guardian",
            "minimumSignatures": {
                  "@id": "ctrl:minimumSignatures",
                  "@type": "xsd:integer"
            },
            "openid": "ddo:openid",
            "publicKeyBase64": "sec:publicKeyBase64",
            "service": "ddo:service",
            "signer": "sec:signer",
            "updated": {
                  "@id": "ddo:updated",
                  "@type": "xsd:dateTime"
            },
            "xdi": "ddo:xdi",
            "EdDsaPublicKey": "sec:EdDsaPublicKey",
            "RsaPublicKey": "sec:RsaPublicKey",
            "OrControl": "ctrl:OrControl",
            "AndControl": "ctrl:AndControl",
            "MofNControl": "ctrl:MofNControl"
      }]
}
```

Appendix C: Standard Key Descriptions

As described in section 6. key description is a standard way to describe a public key or verification key in JSON-LD. This appendix contains a list of key descriptions recommended for use in DDOs.

RSA Keys


```
{
      "owner": [{
            "id": "did:sov:21tDAKCERh95uGgKbJNHYp#key/2",
            "type": ["CryptographicKey", "RsaPublicKey"],
            "expires": "2017-03-22T00:00:00Z",
            "publicKeyBase64":
"MIIBOgIBAAJBAKkbSUT9/Q2uBfGRau6/XJyZhcF5abo7b37I5hr3EmwGykdzyk8GSyJK
3TOrjyl0sdJsGbFmgQaRyV"
      }]
}
```


EdDSA Keys


```
{
      "owner": [{
            "id": "did:sov:21tDAKCERh95uGgKbJNHYp#key/1",
            "type": ["CryptographicKey", "EdDsaSAPublicKey"],
            "curve": "ed25519",
            "expires": "2017-02-08T16:02:20Z",
            "publicKeyBase64":
"IOmA4R7TfhkYTYW87z640O3GYFldw0yqie9Wl1kZ5OBYNAKOwG5uOsPRK8/2C4STOWF+
83cMcbZ3CBMq2/gi25s="
      }]
}
```