DID-Linked Resources Specification

- Document Status
- Introduction
- Terminology
- Purpose
- Context
  - DID Resolution
  - DID URL Dereferencing
- DID-Linked Digital Resources
  - Motivations
  - Design Principles
- Verifiable Data Registry (VDR) and DID Method Requirements
- Resource Parameters
- Linked Resource Metadata
- DID Resolver Requirements
- Query Syntax for Resource DID URLs
- Error messages
- Registration with the W3C DID Specification Registries
- Contributors
- Acknowledgements
- Licensing

Document Status

This document is a Draft Deliverable of the Utility Foundry Working Group.

The current version is Working Draft 01.

Introduction

This is a specification for an extension to the W3C Decentralized Identifiers (DIDs) 1.0 specification to support the identification of **Resources linked to DIDs**, as listed in the W3C DID Specification Registries 1.0, using DID URLs. This specification is intended to supersede the DID URL Resource Parameter Specification.

Terminology

In this document, the keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL", when appearing in ALL CAPITALS, are to be interpreted as described in RFC 2119.

All other terms are linked to their definitions in the W3C Decentralized Identifiers (DIDs) 1.0 specification.

Purpose

The aim of this specification is to define how DID URLs SHOULD act as persistent identifiers for referencing and retrieving Resources (such as data schemas, interface definitions, governance documents, or policy definitions). Through using DID URLs which remain conformant with W3C Decentralized Identifiers (DIDs) 1.0 specification, existing DID Resolvers will be able to dereference these DID URLs to retrieve the identified resource using the DID URl query syntax in this specification.

Context

This specification builds on two existing concepts for processing a DID URL:

1. **DID Resolution**; and
2. **DID URL Dereferencing**.

DID Resolution

DID Resolution is the process of resolving the plain DID (defined by the ABNF from section 3.1 of the DID spec) to a DID document.

This flow is shown in *figure 1,* below:

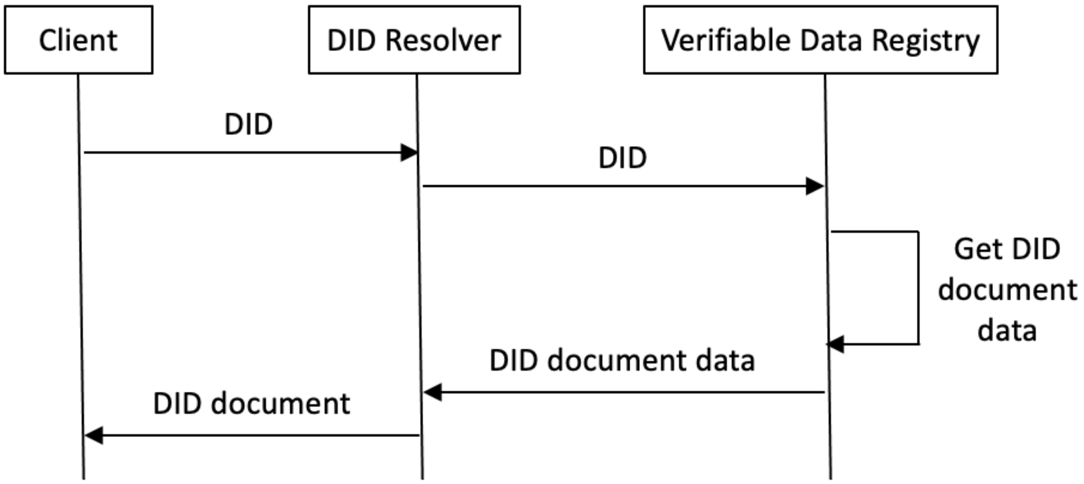

Figure 1: The normal DID resolution process

DID URL Dereferencing

DID URL Dereferencingis the process of resolving a DID to a DID document in order to determine how to **dereference** the remainder of the DID URL (path, query, fragment as defined by the ABNF in section 3.2 of the DID 1.0 specification.)

The path, query or fragment within the DID URL provides a DID Resolver additional specific information regarding the exact object within the DID Document that should be returned.

Normal dereferencing can be conceived in two steps:

1. A DID is resolved to a DID Document;
2. A resource within the DID Document is identified, based on the portion of the DID URL that follows the DID (path, query, fragment as defined by the ABNF in section 3.2 of the DID 1.0 specification.).

An example is processing a DID fragment to return a specific public key from the DID Document. Or alternatively, processing a DID query to return a specific service endpoint, specified in the service section of the DID Document. (Note: the process for how a DID Document is dereferenced is left to each DID Method implementation.)

This flow is shown in *figure 2*, below:

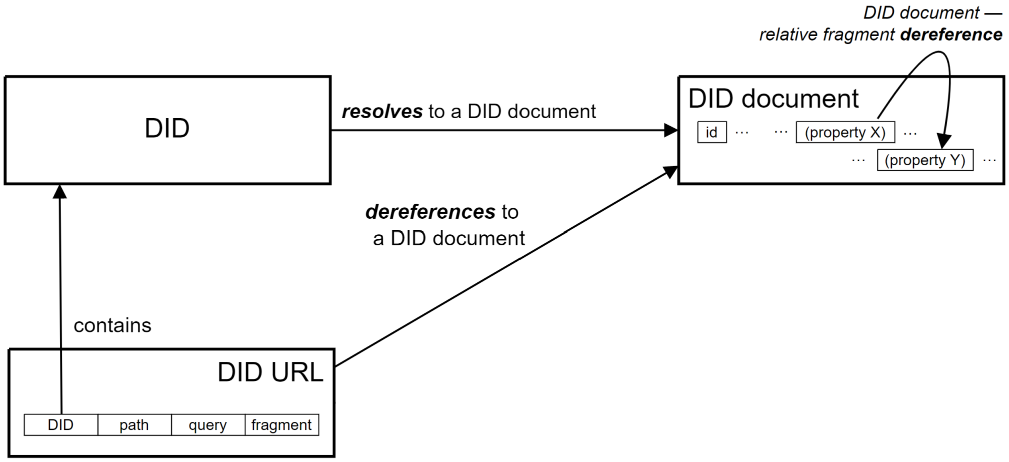

Figure 2: The normal DID URL Dereferencing process

DID-Linked Digital Resources

Motivations

Although the DID Core specification defines an interoperable standard for DID Documents and associated core properties, it currently does not have a standardised way to specify properties of Resources and how to fetch them. A *Resource* is defined as any file or digital asset that is linked to a DID (such as schemas, credential definitions, trusted issuer lists, logos, image assets for credentials etc). Specifying

Design Principles

This specification is intended to improve the way Digital Resources are stored and retrieved through the use of DID URLs, in the following ways:

1. Highly available and easily retrievable
    a. Digital Resources MUST be **accessible using a DID URL** which allows them to be retrieved easily from a distributed ledger using **existing DID Resolvers.**
    b. Di MAY use a Verifiable Data Registry to store and index resources removes the problem identified by centralized systems creating single points of failure, which is a common problem in the way schemas, trust registries and revocation statuses are often stored.
2. Controllable and self-attestable
    a. Resources **MUST be tied to DID Documents** and **control over resources SHOULD be exerted via the same verification method keys** as those written into an associated **DID Document**.
    b. This allows persons to **authenticate with a DID Document to update or prove control of a resource**, which addresses the problem of tamper-proofing identified around centralized cloud providers (which resources are currently commonly stored on).
3. Built to be consumed by client applications
    a. Resources MUST **specify a name, and resource type and compile into a media type**, which provides **specific information to any client application** about what **data format and syntax** the data are expected to be retrieved in.
    b. This allows client applications to **apply business rules** to what types of resources are expected, and which should be accepted, making resources far easier to be consumed by third-party software. This differs from existing Hyperledger Indy resources, which require the client applications to be familiar with Indy in order to process and consume Indy resources.
4. Indexable
    a. Resources SHOULD be versioned with **unique identifiers**, allowing previous versions to be archived within a collection, and **retrieved through querying a unique ID or a version time**.
    b. This mitigates the problem identified of **link rot** when using centralized storage systems since each version is archived immutably.

DID URLs SHOULD be able to dereference directly to a resource, rather than first dereferencing to a DID Document.

This is an exception to the normal 2-step resolution/dereferencing process, enabling **the DID itself to directly identify a digital resource that can be returned directly by the VDR of the associated DID method**. This behaviour may be desirable:

- When the DID serves as a persistent identifier of a machine-readable digital resource that the client wishes to consume directly, such as a data schema, interface definition, or policy definition.
- When the DID serves as a persistent identifier of a human-readable document that needs a long-lived, cryptographically verifiable identifier such as a legal document (e.g., title, deed, will, regulatory filing), a governance framework, or a non-fungible token (NFT) or any other type of digital asset.

This flow is shown in *figure 3*, below:

blocked URL

Figure 3: The relationship of DIDs, DID URLs, DID documents, and Resources

In this case, the client MAY wish to use a DID URL to request that a DID resolver return the identified digital resource in a single step.

Three important notes about this process:

1. **The DID document is not retrieved by the resolver as part of the dereferencing process**. Rather the resolver makes a call to the VDR with the DID URL including a `resource` parameter conformant with this specification. The VDR then follows the specification of the associated DID method to retrieve the identified digital resource and return that resource to the resolver directly.
2. **The resource is still associated with a DID Document.** A DID URL that includes the `resource` parameter does still have an associated DID document like any other DID. However if the DID URL includes the `resource` parameter, the associated DID document is **not involved** in the combined resolution/dereferencing step to fetch the resource specified. If the DID alone is resolved (without the `resource` parameter), it will still return the associated DID document.
    a. If the DID document contains only an `id` property (whose value must be the DID) and no verification methods, then the DID document cannot be updated and the identified digital resource is a static resource that cannot be versioned.
    b. If the DID document contains one or more verification methods, the same verification methods may be used for authenticating/versioning/updating the identified digital resource and/or the DID document.
3. **The DID Document refers to the associated resource via linked resource metadata.** Through associating the resource with a DID Document, the DID Document may generate associated metadata about the resource, defined further below.

Verifiable Data Registry (VDR) and DID Method Requirements

A Verifiable Data Registry (VDR) and DID Method conforming to this specification to include support for `resources` identifiable via DID URLs **MUST define**:

1. A method to write a resource to the VDR which is referenceable via an immutable and uniquely identifiable using DID URLs.
2. A method to write a resource to the VDR which either directly includes or is mapped to the Resource Parameters.
3. A method to write a resource to the VDR which is linked-to and referenced within an associated DID Document, whilst remaining conformant with W3C Decentralized Identifiers (DIDs) 1.0 specification.
4. A method to version resources, with each current and old version easily accessible in the future.
5. A method to map the individual fields of the resource metadata to the Query Syntax Resource Parameters.
6. A method to dereference directly to a resource using a DID URL with the Query Syntax Resource Parameters.

A Verifiable Data Registry (VDR) and DID Method conforming to this specification to include support for `resources` identifiable via DID URLs **SHOULD define**:

1. A method to update/manage the resource using the verification method relationships and keys within the associated DID Document.
2. A method to authenticate resource transactions using the verification method relationships and keys within the associated DID Document.

A Verifiable Data Registry (VDR) and DID Method conforming to this specification to include support for `resources` identifiable via DID URLs **MAY define**:

1. A method to organize resources associated with a DID Document into a collection of resources.
2. A method to mark resources as deprecated or superseded by new versions.
3. A method to dereference directly to a resource using DID URL paths '/'.

Resource Parameters

The following list defines which specific parameters a resource MUST contain to conform with this specification, and which parameters are OPTIONAL.

Resource Parameter	Requirement	Description
`"resourceUri"`	YES	A string or a map that conforms to the rules of [*RFC3986*] for URIs which SHOULD directly lead to a location where the resource can be accessed from. For example: `did:example:46e2af9a-2ea0-4815-999d-730a6778227c/resources/0f964a80-5d18-4867-83e3-b47f5a756f02,` or, `https://gateway.ipfs.io/ipfs/bafybeihetj2ng3d74k7t754atv2s5dk76pcqtvxls6dntef3xa6rax25xe`
`"resourceCollectionId"`	OPTIONAL	A string that conforms to a method specific unique identifier format. For example: `46e2af9a-2ea0-4815-999d-730a6778227c`
`"resourceId"`	OPTIONAL	A string that conforms to a method specific unique identifier format. For example: `0f964a80-5d18-4867-83e3-b47f5a756f02`
`"resourceName"`	YES	A string that uniquely names and identifies a resource. This property, along with the resourceType below, can be used to track version changes within a resource. For example: `degreeLaw`
`"resourceType"`	YES	A string that identifies the type of resource. This property, along with the resourceName above, can be used to track version changes within a resource. Not to be confused with media type. (TBC to add to DID Spec Registries) For example: `JSONSchema2020`
`"resourceVersionId"`	OPTIONAL	A string that uniquely identifies the version of the resource provided by the resource creator as a tag. For example: `1.3.1`
`"mediaType"`	YES	A string that identifies the IANA-registered Media Type for a resource. For example: `application/json`
`"created"`	YES	A JSON String serialized as an XML Datetime normalized to UTC 00:00:00 and without sub-second decimal precision. For example: `2020-12-20T19:17:47Z`
`"checksum"`	OPTIONAL	A string that provides a checksum (e.g. SHA256, MD5) for the resource to facilitate data integrity. For example: `7b2022636f6e74656e74223a202274657374206461746122207d0ae3b0c44298`
`"previousVersionId"`	OPTIONAL	The value of the property *MUST* be an string. This is the previous version of a resource with the *same* `resourceName` and `resourceType.` The value must be 'null' if there is no previous version. For example: `67618cfa-7a1d-4be3-b9b2-3a9ea52af305`
`"nextVersionId"`	OPTIONAL	The value of the property *MUST* be an string. The value must be 'null' if there is no next version. For example: `null`

Linked Resource Metadata

The Resource Parameters for a resource SHOULD be referenced within an associated DID Document `didDocumentMetadata` property, for example:

Linked Resource Metadata

```
    "didDocumentMetadata": {
  "created": "2022-07-19T08:29:07Z",
  "versionId": "57543FA1D9C56033BABBFA3A438E0A149E01BBB89E6D666ACE1243455AA6F2BC",
  "linkedResourceMetadata": [
    {
      "resourceURI": "did:cheqd:mainnet:46e2af9a-2ea0-4815-999d-730a6778227c/resources/0f964a80-5d18-4867-83e3-
b47f5a756f02",
      "resourceCollectionId": "46e2af9a-2ea0-4815-999d-730a6778227c",
      "resourceId": "0f964a80-5d18-4867-83e3-b47f5a756f02",
      "resourceName": "DegreeLaw",
      "resourceType": "CL-Schema",
      "mediaType": "application/json",
      "created": "2022-07-19T08:40:00Z",
      "checksum": "7b2022636f6e74656e74223a202274657374206461746122207d0ae3b0c44298",
      "previousVersionId": null, // null if no previous version, otherwise, resourceId of previous version
      "nextVersionId": null, // null if no new version, otherwise, resourceId of new version
    }
  ]
}
```

DID Resolver Requirements

1. If a DID URL includes any of the `resource` Query Syntax Parameters (below), a conforming DID resolver MUST return the digital resource identified by the DID URL from the VDR, provided such resource is available.
    a. If the DID resolver is unable to return the identified resource, the resolver MUST return an error (see Error Messages)
2. If a DID URL includes the `resourceMetadata` parameter with a value of `true`, a conforming DID resolver MUST return the requested associated metadata of the digital resource identified from the VDR, provided such resource is available.
    a. If the DID resolver is unable to return the identified resource in the requested media type, the resolver MUST return an error (see Error Messages)
3. If a DID URL includes the `resourceMetadata` parameter with a value of `false`, a conforming DID resolver SHOULD ignore the parameter.
4. If a DID URL includes a custom, implementation-specific path, specified by the associated DID method, a conforming DID resolver MUST return the digital resource identified by the DID URL from the VDR, provided such resource is available.
5. If the DID alone is resolved **without** the `resource` parameter, it MUST return the authoritative DID document as defined in W3C Decentralized Identifiers (DIDs) 1.0.

Query Syntax for Resource DID URLs

To enable combined resolution/dereferencing behavior, this specification defines multiple DID URL parameters to fetch `resource` or associated metadata . If a DID method specification supports these parameters, and if a DID URL using that method includes the parameter with a valid value, then when a resolver calls the associated VDR using that DID URL, the VDR returns the identified digital resource, **not** the DID document.

IMPORTANT: DID URL queries should be fully qualified so that they **uniquely identify a single resource, or single resource version unless expressly specified**.

Common and standardized `resource` parameters:

Parameter	Type	Example
`"resourceId"`	String	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceId=0f964a80-5d18-4867-83e3-b47f5a756f02
`"resourceName"`	String	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw
`"resourceType"`	String	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020
`"resourceVersionId"`	String	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceVersionId=1.3.1
`"versionTime"`	XML Datetime	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020&versionTime=2015-03-11T05:30:02Z
`"versionId"`	String	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?versionId=0f964a80-5d18-4867-83e3-b47f5a756f02

`"versionTime"`	XML Datetime	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020&versionTime=2018-07-19T08:40:00Z
`"linkedResource"`	Boolean	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?linkedResource=true // *note that this would only be a valid query if there is ONLY ONE resource associated with the DID and DID Document.*
`"resourceMetadata"`	Boolean	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020&versionTime=2018-07-19T08:40:00Z&resourceMetadata=true or, did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceMetadata=true // *note that this would only be a valid query if there is ONLY ONE resource associated with the DID and DID Document.*
`"latestResourceVersion"`	Boolean	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020&latestResourceVersion=true
`"allResourceVersions"`	Boolean	did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020&allResourceVersions=true

Error messages

There MAY be instances where the resolver has not been provided enough specific information to process the request and fetch a single resource.

In these cases, the error messages should follow the Error section in the DID Resolution specification here.

Example	Explanation	Error
did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw	There MAY be insufficient information for the VDR to process the request (for example, if there are multiple resource versions of the name degreeLaw, or different resource types of the name degreeLaw).	`"notFound"`.
did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020&versionTime=2018-07-19T08:40:00Z	The identified resource does not exist at the versionTime.	`"notFound"`.
did:example:46e2af9a-2ea0-4815-999d-730a6778227c?resourceName=degreeLaw&resourceType=JSONSchema2020&versionId=2.1.4	The identified version does not exist.	`"notFound"`.

Registration with the W3C DID Specification Registries

The Query Syntax for Resource DID URLs defined by this specification will be registered with the W3C DID Specification Registries 1.0 at the following URL:

https://www.w3.org/TR/did-spec-registries/#resource

Contributors

To comply with the intellectual property rights protections in the charter of the ToIP Foundation (as required by all Joint Development Foundation projects hosted the Linux Foundation), all contributors to this Pre-Draft Deliverable **MUST be current members of the ToIP Foundation**. The following contributors each certify that they meet this requirement:

- Alex Tweeddale

Acknowledgements

The authors wish to thank the editors and contributors to the W3C Decentralized Identifiers (DIDs) 1.0 specification.

Licensing

This is a publicly available specification published by the ToIP Foundation under the following licenses:

- Copyright: Creative Commons Attribution 4.0.
- Patent: Joint Development Foundation W3C Mode (based on the W3C Patent Policy).