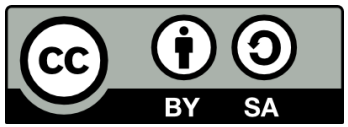
Decentralised IDentifiers and Friends

Vurucu ve Akillica Alt Başlık

Abdulhamit Kumru

Blokzincir Laboratuvarı

2020



License Attribution-ShareAlike 4.0 International

Nelerden bahsedecegiz

- DID Core
  - DID-common-java
- DID Auth
- DID Communication
  - Peer DID ?
  - V1 (Aries), V2 (DIF DIDcomm messaging) ?
- Proposed Applications
  - DID SIOP *identity.foundation/did-siop/*
- Similar Applicable Scenarios
  - DID SAML (IdP)
  - DID CAS (IdP)
  - DID PAM

Daha Soyut Bir giris

!!! gereginden fazla degindigim yerler olabilir !!! giris slayti
!!! ikna edici bir giris hazirla
!!! did in argumanlarini daha belirgin yap
!!! aktif gelistirilen did methodlarindan bahset !!! json ld yi iyi anla, sunumda bahset

DID Core

!!! bu spec hakkinda genel bilgiler

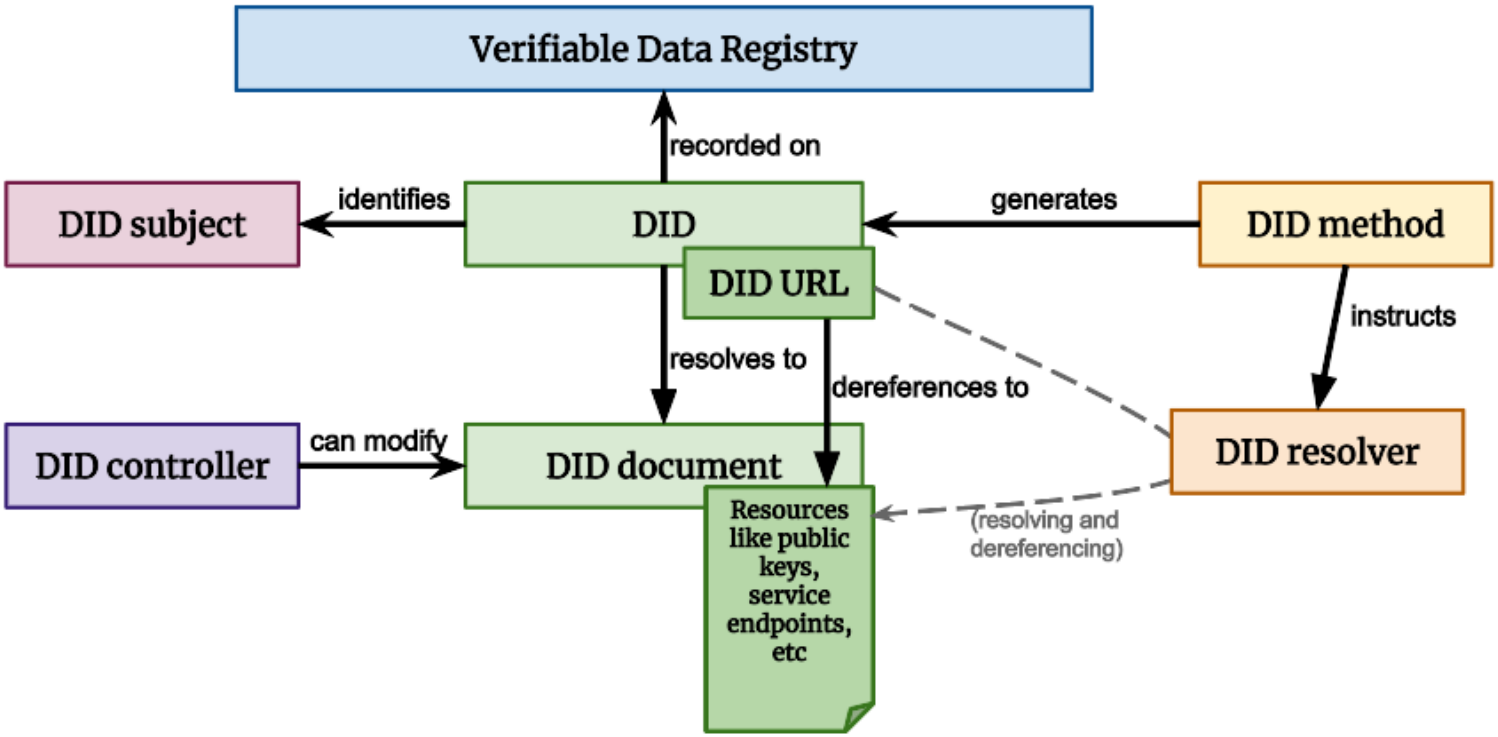
Four Core Properities of DID

!!! notlar ve bura uzerinde biraz dur

1. **A permanent (persistent) identifier**
   It never needs to change
2. **A resolvable identifier**
   You can look it up to discover metadata
3. **A cryptographically-verifiable identifier**
   You can prove control using cryptography
4. **A decentralized identifier**
   No centralized registration authority is required

Architecture Overview

!!! not al



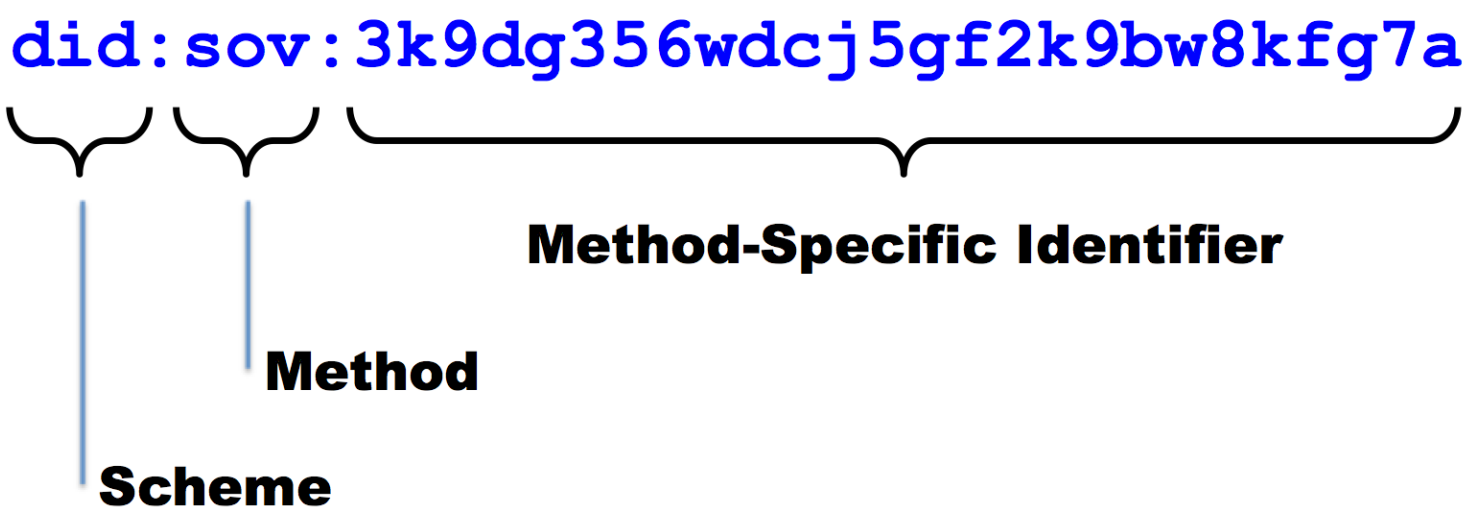
DIDs and DID URLs

!!! not: query ye detayli deginecegiz

A DID, or Decentralized Identifier, is a URI composed of three parts: ***the scheme*** "did:", a ***method identifier***, and a unique, ***method-specific identifier*** generated by the DID method.

DIDs are resolvable to DID documents. A DID URL extends the syntax of a basic DID to incorporate other standard URI components (path, query, fragment) in order to locate a particular resource.

DID Format


did:sov:3k9dg356wdcj5gf2k9bw8kfg7a
Method-Specific Identifier
Method
Scheme

DID Subjects

!!! gorsel ekle ?

The subject of a DID is, by definition, the entity identified by the DID. The DID subject may also be the DID controller. Anything can be the subject of a DID: person, group, organization, physical thing, logical thing, etc.

DID Controllers

!!! kisalt
!!! gorsel ekle ?

The controller of a DID is the entity (person, organization, or autonomous software) that has the capability—as defined by a DID method—to make changes to a DID document. This capability is typically asserted by the control of a set of cryptographic keys used by software acting on behalf of the controller, though it may also be asserted via other mechanisms. Note that a DID may have more than one controller, and the DID subject can be the DID controller, or one of them.

Verifiable Data Registries

!!! kisalt
!!! gorsel ekle ?

In order to be resolvable to DID documents, DIDs are typically recorded on an underlying system or network of some kind. Regardless of the specific technology used, any such system that supports recording DIDs and returning data necessary to produce DID documents is called a verifiable data registry. Examples include distributed ledgers, decentralized file systems, databases of any kind, peer-to-peer networks, and other forms of trusted data storage.

DID documents

DID documents contain metadata associated with a DID. They typically express verification methods (such as public keys) and services relevant to interactions with the DID subject.

Minimal Self-managed DID document Example

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCw..."
  }],
  "service": [{
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

DID Methods

DID methods are the mechanism by which a particular type of DID and its associated DID document are created, resolved, updated, and deactivated using a particular verifiable data registry. DID methods are defined using separate DID method specifications.

DID resolvers and DID resolution

!!! not: detayli spec linkte, burda did res. in detayina girmeyecegiz

A DID resolver is a software and/or hardware component that takes a DID (and associated input metadata) as input and produces a conforming DID document (and associated metadata) as output. This process is called DID resolution.

detailed spec w3c-ccg.github.io/did-resolution/

DID Resolve Example

!!! gecis

`did:example:1234;version-id=4#keys-1 # resolves to`

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi#keys-1",
  "type": "RsaVerificationKey2018",
  "publicKeyPem": "-----BEGIN PUB...0101010..END PUB -----\r\n"
}
```

Identifier

!!! identifier giris slayti

DID Syntax

- The generic DID scheme is a URI scheme conformant with [RFC3988].
- The DID scheme name **MUST** be an ASCII lowercase string.
- The DID method name **MUST** be an ASCII lowercase string.

```
# Ethr-DID
did:ethr:0xb9c5714089478a327f09197987f16f9e5d936e8a
```

A DID is expected to be persistent and immutable. That is, a DID is bound exclusively and permanently to its one and only subject. Even after a DID is deactivated, it is intended that it never be repurposed.

- aciklama nota alinabilir

DID URL Syntax


```
did-url = did path-abempty [ "?" query ] [ "#" fragment ]
```

DID Parameters

The DID URL syntax supports a simple format for parameters based on the query component. Adding a DID parameter to a DID URL means that the parameter becomes part of the identifier for a resource.

DID Parameters

Relative Reference

A relative URI reference according to RFC3986 Section 4.2 that identifies a resource at a service endpoint, which is selected from a DID document by using the service parameter. Support for this parameter is **REQUIERED**

Relative Reference Example

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:1234",
  "verificationMethod": [{
    "id": "did:example:1234#key-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:1234",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwn..."
  }, ...],
  "authentication": [
    // relative DID URL to 'did:example:1234#key-1
    "#key-1"
  ]
}
```

DID Parameters

service

Identifies a service from the DID document by service ID. Support for this parameter is **REQUIRED**

```
did:foo:21tDAKCERh95uGgKbJNHYp?service=agent
```

DID Parameters

version-id

Identifies a specific version of a DID document to be resolved (the version ID could be sequential, or a UUID, or method-specific). Support for this parameter is **OPTIONAL**

DID Parameters

version-time

Identifies a certain version timestamp of a DID document to be resolved. Support for this parameter is **OPTIONAL**

```
did:foo:21tDAKCERh95uGgKbJNHYp?version-time=2002-10-10T17:00:00Z
```

DID Parameters

hl

!!! not: iyi bir ozellik ama mekanizmasini tam olarak anlayamadim

A resource hash of the DID document to add integrity protection, as specified in Hashlink RFC. This parameter is *non-normative*

```
# url encoded hash link
hl:zm9YZpCjPLPJ4Epc:z3TSgXTuaHxY2ts...7DYuQ9QTPQyLHy
```

DID URL Syntax

Path

A DID path is identical to a generic URI path

```
did:example:123456/path
```

Query

A DID query is derived from a generic URI query and **MUST** conform to DID URL Syntax rules.
If a DID query is present, it **MUST** be used with DID Parameters.

```
did:example:123456?query=true
```

DID URL Syntax

Fragment

A DID fragment is used as method-independent reference into a DID document or external resource. DID fragment syntax and semantics are identical to a generic URI fragment and **MUST** conform to RFC 3986

```
did:example:123#agent # service endpoint
did:example:123#public-key-0 # verification method
```

Relative DID URLs

!!! buraya biraz daha bak !!! ornegine deginmistik

A relative DID URL is any URL value in a DID document that does not start with `did:<method-name>:<method-specific-id>`.

```
// ... relative DID URL to ‘did:example:1234#key-1‘
"authentication": [ "#key-1" ]
// ...
```

Example DID URLs

!!! gecis !!! not: did url ye degin

```
# A DID URL with a 'service' DID parameter
did:foo:21tDAKCERh95uGgKbJNHYp?service=agent
# A DID URL with a 'version-time' DID parameter
did:foo:21tD...gKbJNHYp?version-time=2002-10-10T17:00:00Z

did:example:1234/
did:example:1234#keys-1
did:example:1234;version-id=4#keys-1
did:example:1234/my/path?query#fragment
did:example:1234;service=hub/my/path?query#fragment
```

Core Properties

!!! core prop giris slayti
!!! hepsini anlatmaya gerek yok

- id
- authentication
- controller
- service
- verificationMethod
- assertionMethod
- keyAgreement
- capabilityDelegation ?
- capabilityInvocation ?

id

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

The value of id in the resolved DID document **MUST** match the DID that was resolved, or be populated with the equivalent canonical DID specified by the DID method, which SHOULD be used by the resolving party going forward.

Software/Tools

!!! did core sonu ekle

did-common-java

Sources

- https://www.w3.org/TR/did-core/
- https://w3c-ccg.github.io/did-resolution/
- https://github.com/WebOfTrustInfo/rwot5-boston/blob/master/topics-and-advance-readings/did-primer.md
- https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/