



KILT: Credentials for web3

# Decentralized Identifiers (DIDs): a practical primer

@KILTProtocol @maudnals

# Agenda

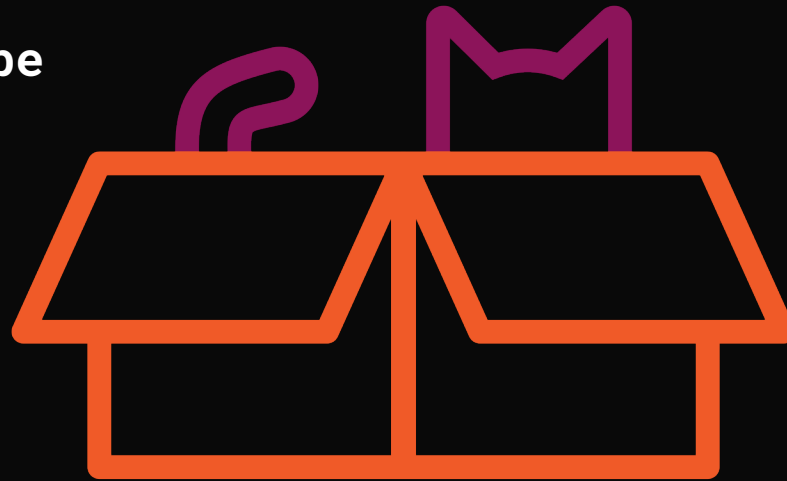


1. KILT
2. DIDs
3. DIDs in our system
4. DIDs in practice: Universal Resolver driver implementation

# But first



- KILT Protocol, Ingo Rübe  
@KILTProtocol  
kilt.io
- Maud @maudnals



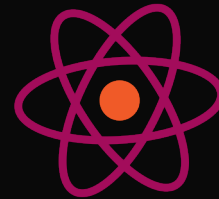
- ... You x DIDs?



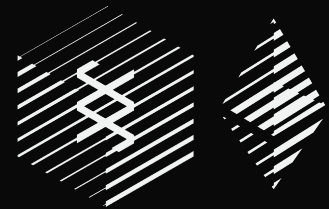
**Permissionless  
trust  
infrastructure**



**Credentials  
for the web3**



**Regulation-friendly  
virtual structures**



**Parity Substrate-  
based**

# DIDs?

Now



With DIDs



Source: SSI Meetup

# DIDs?

A DID is a new type of **globally unique identifier**, that does not require a centralized registration authority; control of the identifier can be proved cryptographically.

<https://w3c-ccg.github.io/did-spec/>

# 4 core properties of DIDs



1. Persistent

2. Resolvable

3. Cryptographically verifiable

4. Decentralized

NOT: human-readable

# DID + DID Document



Scheme

**did:** **kilt** :123456789abcdefghijk

**DID Method**      **DID Method Specific String**

did:ethr:  
did:btc:  
did:sov:  
did:uport:  
...

```
{
  "@context": "https://www.w3.org/2019/did/v1",
  "id": "did:kilt:123456789abcdefghi",
  "authentication": [
    {
      "id": "did:kilt:123456789abcdefghi#keys-1",
      "type": "RsaVerificationKey2018",
      "controller": "did:kilt:123456789abcdefghi",
      "publicKeyPem": "-----BEGIN PUBLIC
KEY...END PUBLIC KEY-----\r\n"
    }
  ],
  "service": [
    {
      "id": "did:kilt:123456789abcdefghi#vcs",
      "type": "VerifiableCredentialService",
      "serviceEndpoint": "https://kilt.com/vc/"
    }
  ]
}
```



# CRUD



- **Create**
- **Resolve**
- **Update**
- **De-activate**

} = f(DID method)  
= `kilt` in our case

# DIDs in KILT



ek, **EK**

Encryption  
keypair

sk, **SK**

Signing  
keypair

did:kilt:5GZPvZad...

- ◆ Identifier
- ◆ **EK**
- ◆ **SK**
- ◆ DID Doc Ref



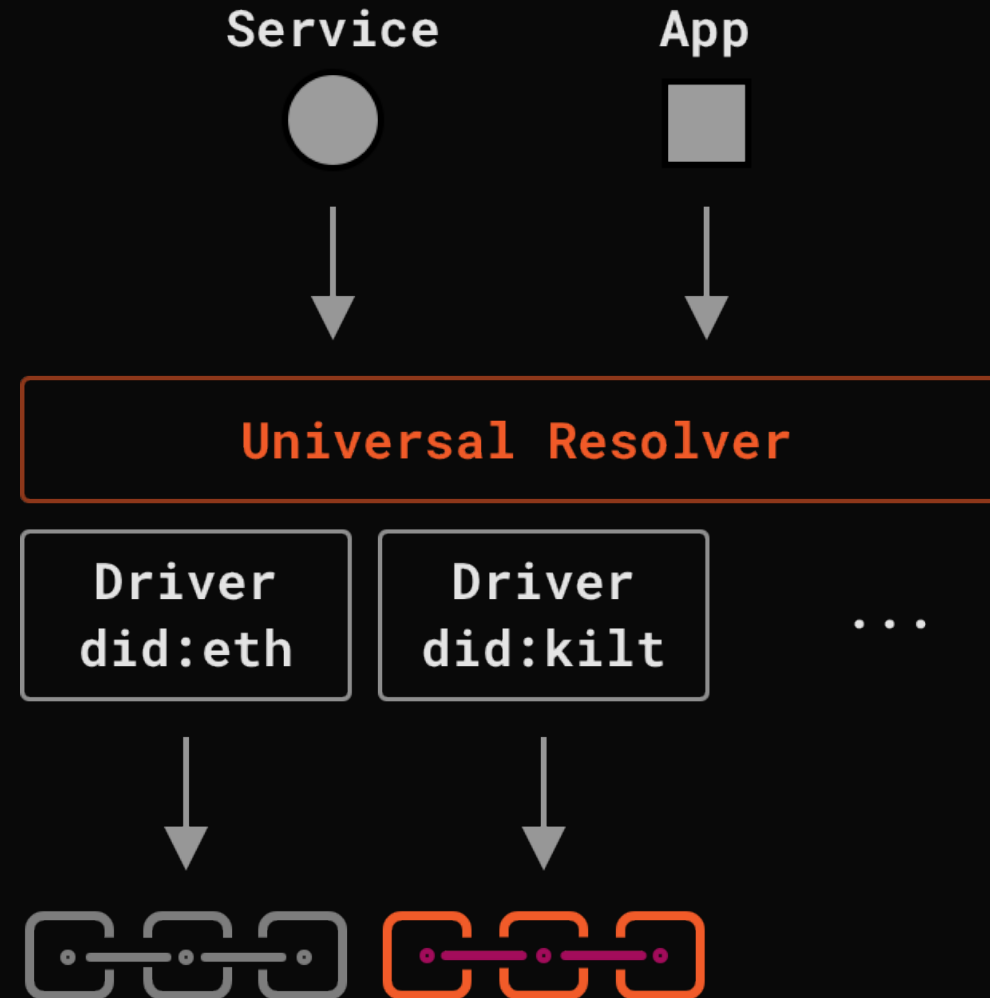
KILT Blockchain

- ◆ **EK**
- ◆ **SK**
- ◆ Service endpoint



Centralized\*  
Service

# Universal Resolver by DIF



👉 [uniresolver.io](https://uniresolver.io)

# Driver guidelines



- **Resolve functionality only:** `resolve(did) -> diddoc`
- **Open-source**
- **Container on dockerhub**
- **Documented methods**

# KILT's dockerized driver



The screenshot shows a VS Code editor window with the following components:

- EXPLORER:** Shows the project structure with folders like `.vscode`, `node_modules`, and `src`. The `src` folder contains `const.js`, `index.js` (selected), and `utils.js`. Other files include `.dockerignore`, `.eslintrc.json`, `.gitignore`, `Dockerfile`, `package.json`, `README.md`, and `yarn.lock`.
- JS index.js:** The main file being edited, containing the following code:

```
src > JS index.js > driver.get() callback
9  const driver = express();
10
11 driver.get(URI_DID, async function(req, res) {
12   const { did } = req.params;
13   const address = getKiltIdFromDid(did);
14   const storageLocation = await getDidDocStorageLocation(address);
15
16   fetch(`${PROTOCOL}:${storageLocation}`)
17   .then(response => response.json())
18   .then(jsonResponse => {
19     const didDocumentAsJSON = JSON.stringify(
20       getDidDocumentFromJsonResponse(jsonResponse)
21     );
22     res.send(didDocumentAsJSON);
23   })
24   .catch(reason => {
25     console.log(reason);
26     res.sendStatus(404);
27   });
28 });
29
30 driver.listen(PORT, () => {
```
- PROBLEMS:** Shows 4 errors.
- OUTPUT:** Empty.
- DEBUG CONSOLE:** Empty.
- TERMINAL:** Shows the command `kilt-did-driver git:(master)` being executed in a `zsh` shell.

# Food for thought



- DID fragments
- Remote or not
- Universal Registrar

**did:kilt:123456#oidc**

# Thank you

@KILTProtocol

@maudnals

kilt.io

Check out TCAs 🔥  
Slides coming soon

DID Resolution Webinar // DIF at SSI Meetup

DID Methods Registry

Universal Resolver: introduction