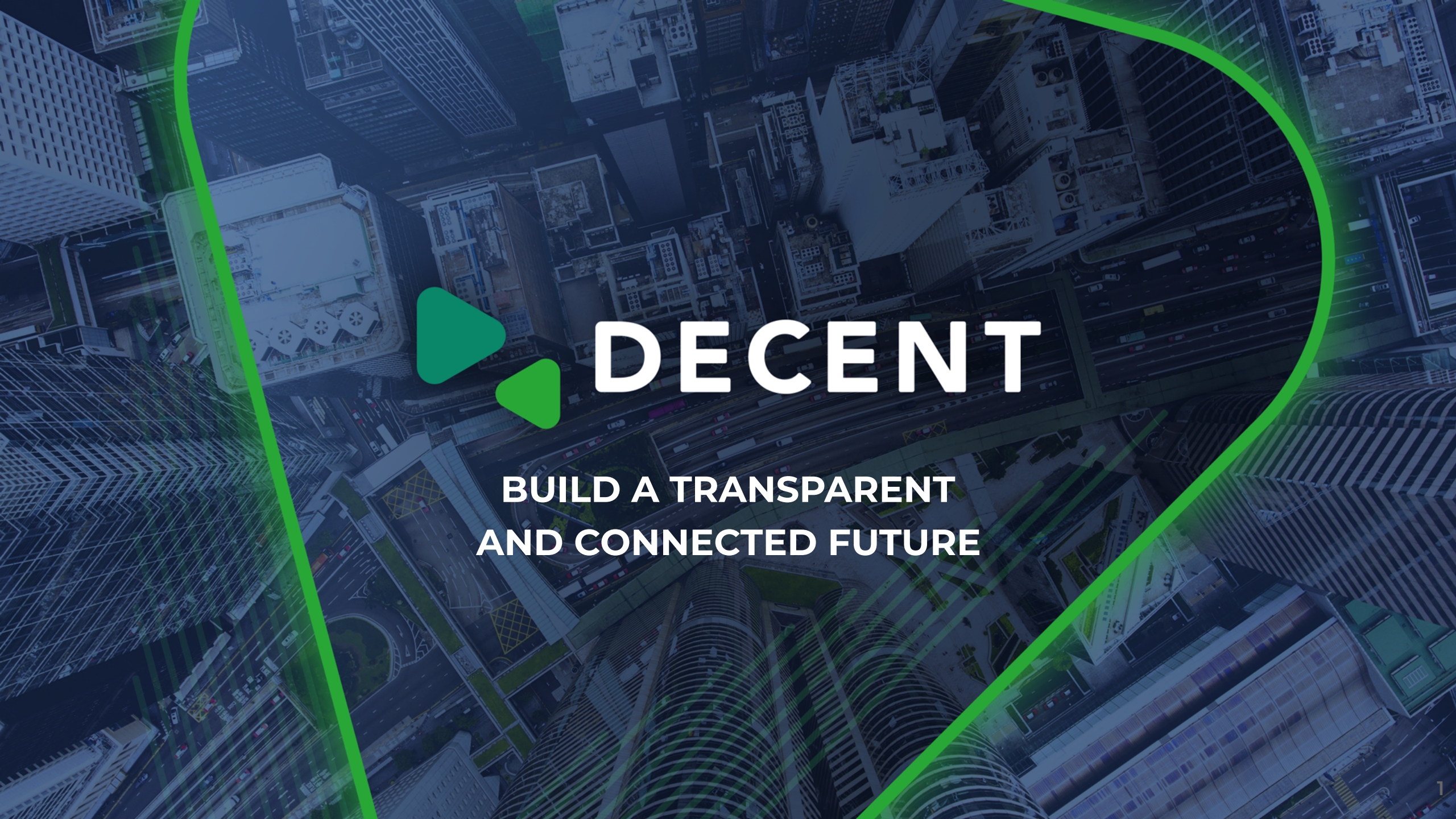
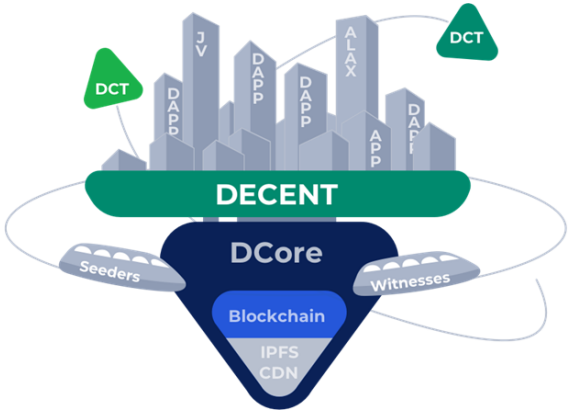

DECENT
BUILD A TRANSPARENT
AND CONNECTED FUTURE

Who We Are.



DECENT

Founded in 2015, DECENT is one of the first blockchain companies worldwide. We have developed our own, proprietary blockchain protocol, named DCore.

DCore was developed for the purpose of digital content distribution. However, thanks to its flexibility, stability and speed, it is perfectly suited to be used for a wide range of use cases.

DECENT is dedicated to building an ecosystem upon its proprietary blockchain technology to help developers and businesses adapt to a decentralised future.

Blockchain – “The Trust Machine”.

The Economist 2015

Transparency — Blockchain is a Public List of Transaction Data

Decentralised Trust — Disruption of Middlemen

Immutability — Irreversible History of Transactions

Cost-effectiveness — Elimination of 3rd Party Costs

Scalability — Useable in Various Sectors

Security — Unhackable Technology, Encryption

DCore Use Cases

DECENT: Infrastructure Provider.

Real-world Use Cases of DCore and Its Features:

- Industry 4.0/Smart Energy
- Supply chain management
- Digital content distribution
- Ticketing
- Gaming & eSports
- Digital advertising
- Data tracking & audit in logistics
- Smart energy
- Transparent fundraising and donations management

ALAX

ALAX Overview.

For further information about the project, visit alax.io

DCore Use Case for Mobile Gaming

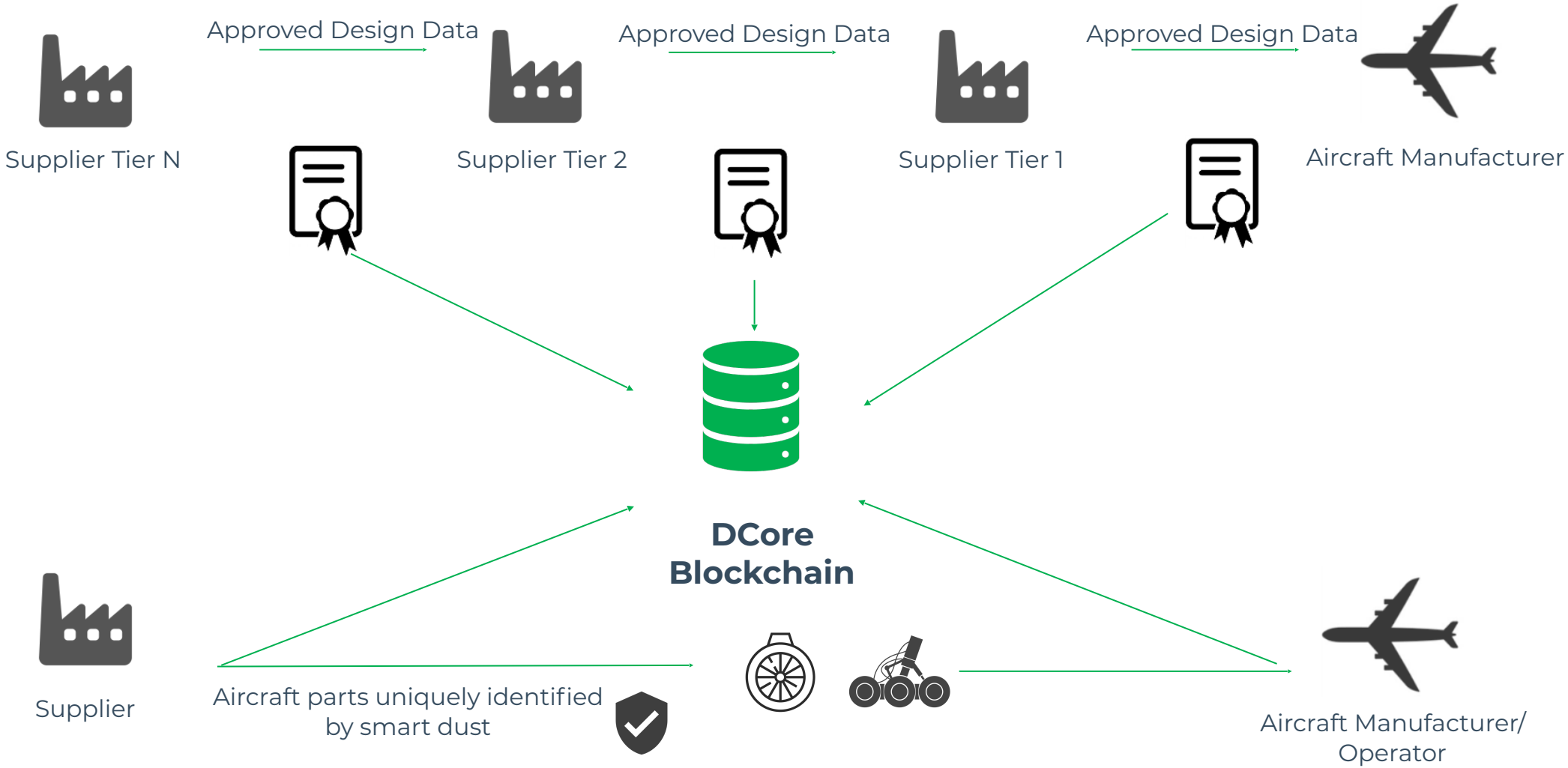
- ALAX is a mobile game distribution platform
- Integrated payment processing system
- Focus on unbanked people (over 2 billion according to Findex).

Simple and cost effective delivery of mobile applications and games to people around the world, utilizing blockchain technology.

3IPK

DCore Solution: Blockchain for Airworthiness Process Management

Certified Data Approvals Stored Securely in DCore



Unique Identifiers of Each Part Stored Securely in DCore

DIGITAL PROOF

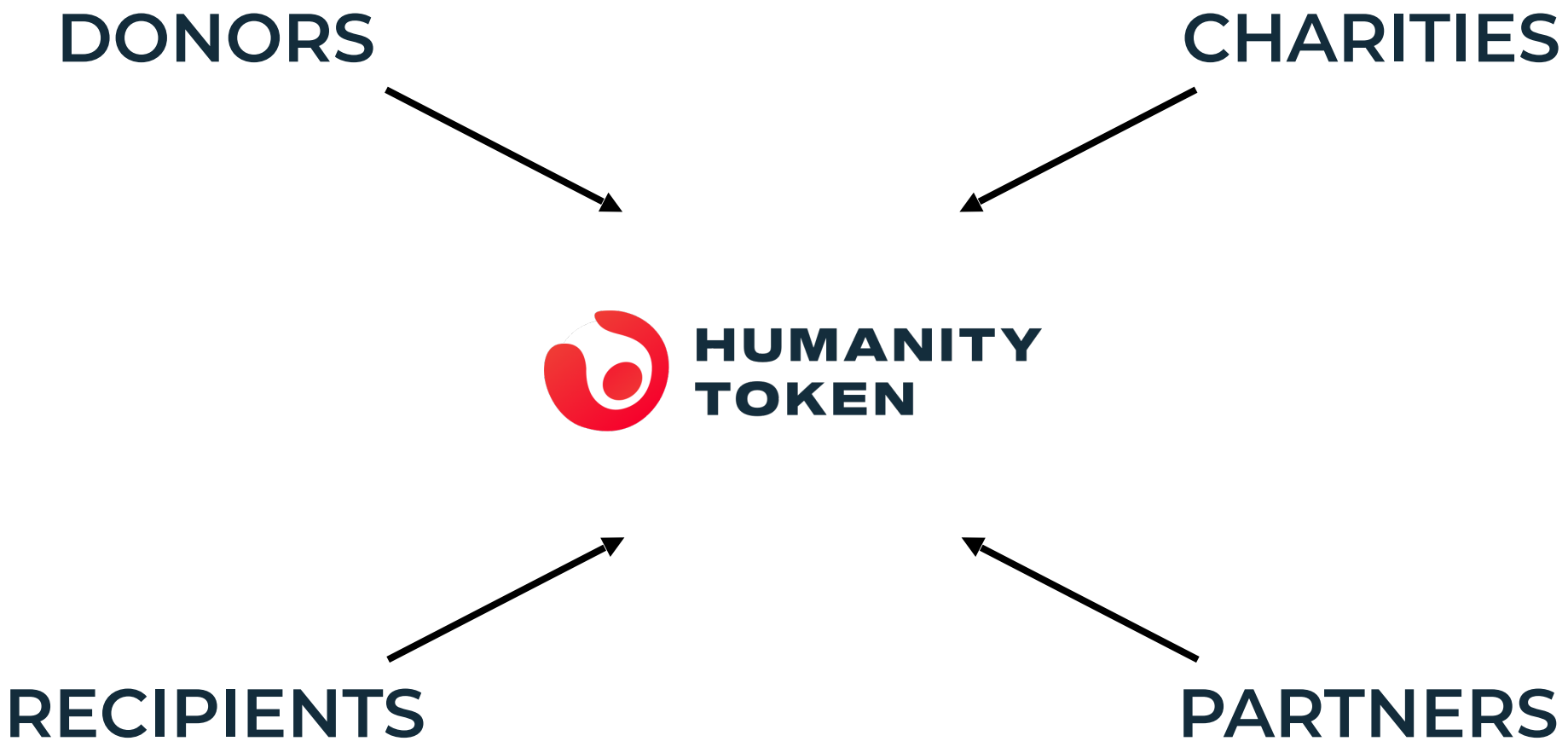
Digital Proof Overview.

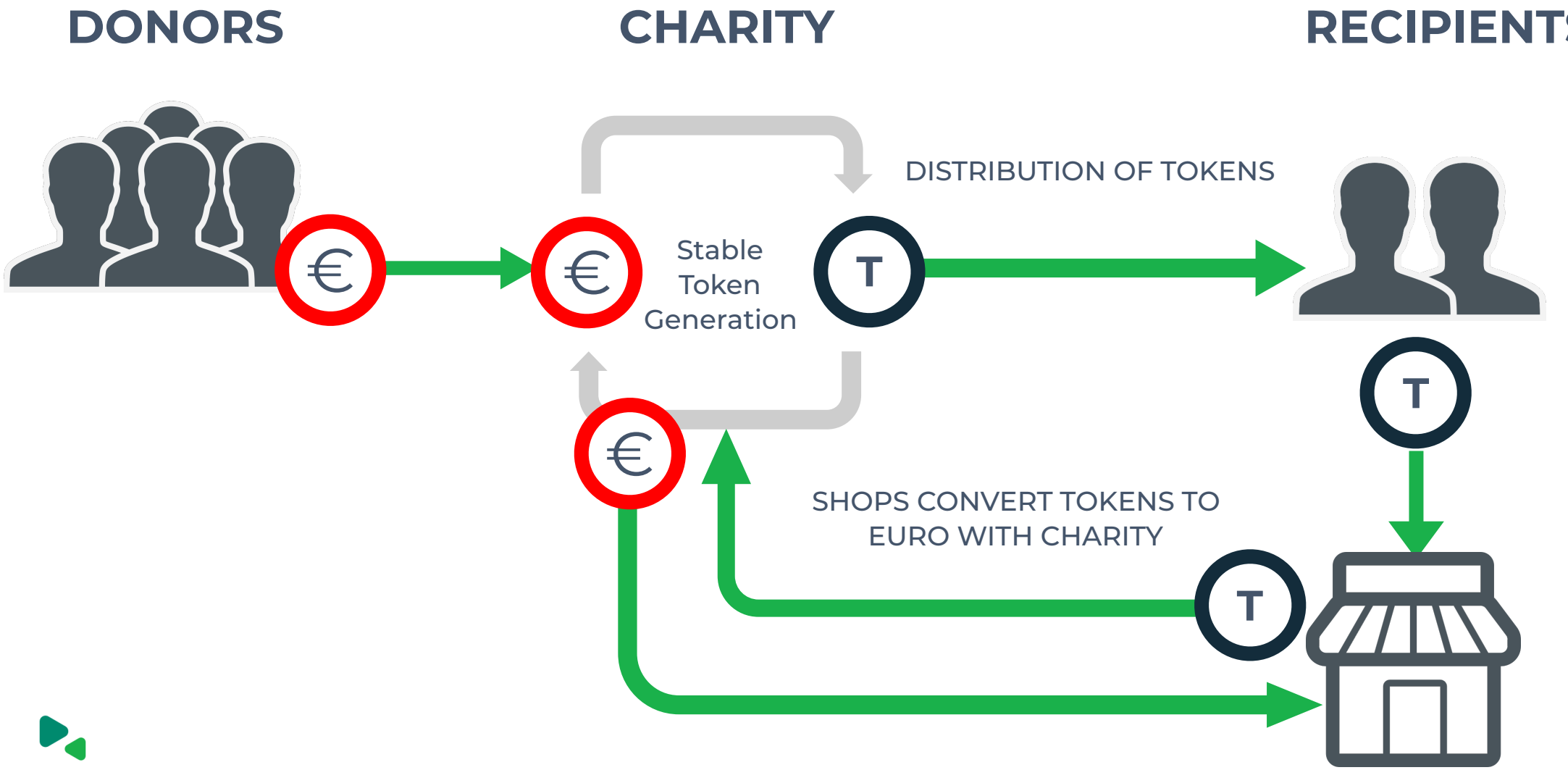
Objectively Verify Authorship and Timestamp of Any Document in 5 Minutes

- Decentralised online notary platform
- Time-stamped Proof of Existence on blockchain
- Created to protect and democratize intellectual property rights

DIGITAL PROOF
BY DECENT

HUMANITY TOKEN


DONORS
CHARITIES
HUMANITY
TOKEN
RECIPIENTS
PARTNERS


DONORS
CHARITY
RECIPIENTS
DISTRIBUTION OF TOKENS
Stable
Token
Generation
SHOPS CONVERT TOKENS TO
EURO WITH CHARITY

FOLLOW US

Website
decent.ch

Telegram
t.me/decentch

Twitter
twitter.com/DECENTplatform

Reddit
reddit.com/r/Decentplatform

Instagram
instagram.com/decentplatform

YouTube
youtube.com/c/DecentCh

LinkedIn
linkedin.com/company/decent-ngo-

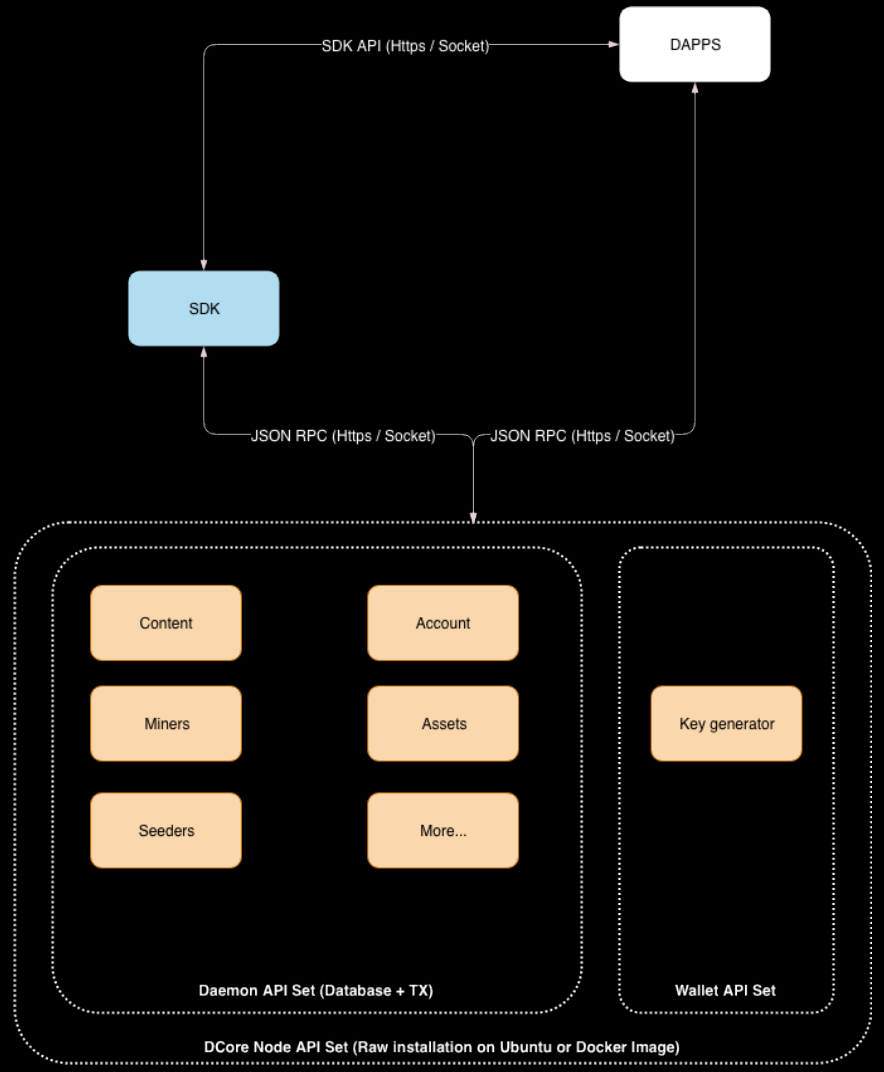
Facebook
facebook.com/DECENTplatform

DECENT

How to Use DCore

SDKs & More

Where to plug in the SDK?



DCore Nodes

- Default (Daemon+Wallet)
- Default (Daemon only)
- Miner
- Seeder

How to use it?

- **JSON-RPC** as raw API connectivity over Https / Socket
- **SDKs** (Encapsulated JSON-RPC raw connectivity/ functionality)
  - Similar DSL patterns (RX) across all platforms (**Interoperability for devs**)
  - Over **Https** all APIs without TX confirmation
  - Over **Socket** all APIs, with TX confirmation callbacks support

DECENT

Supported Platforms

- Typescript (Web, NodeJS, Electron)
- Kotlin (Android, All other JVM)
- Swift (iOS, OS X, Linux) - Developer preview (iOS)
- PHP
- .Net (C#), Python - Future

DECENT

What can we do?

DATABASE API

- Account
- Asset
- Balance
- Content
- More… (Miners, Seeders, Validation, TX, etc…)

TRANSACTION API

- Broadcast (Operations)
  - Create Content
  - Create Account
  - More …

DECENT

Useful Terms

- **Block + TX** - Basic constructs for blockchain network
- **ChainObject** - ID of an item in blockchain DB (e.g. 1.2.3)
    - Space . Type . Identification
- **ECKeyPair** - Public and Private key pair
    - Brain key, Private key, Public key

DECENT

Start It Up

TYPESCRIPT

```
import { DCoreSdk } from "dcorejs-sdk";

// Create HTTP api

const apiHttp = DCoreSdk.createForHttp( ... );

// Or Socket api

const apiWss = DCoreSdk.createForWebSocket( ... );
```

KOTLIN

```
import ch.decent.sdk.DCoreApi
import ch.decent.sdk.DCoreSdk

// Create HTTP api

DCoreApi apiHttp = DCoreSdk.createForHttp( ... )

// Or Socket api

DCoreApi apiWss = DCoreSdk.createForWebSocket( ... )
```

RPC Background

SDK Call

```
// Simple API call for account object

api.accountApi
   .get(“1.2.3".toChainObject());
```

RPC Raw call

```
// Raw RCP call api calls for account object

// Blockchain login api call
{“method”:"call","params":[1,"login",["",""]],"id":3}

// Blockchain login api result
{“id”:3,"result":true}

// Blockchain database api call
{“method”:"call","params":[1,"database",[]],"id":2}

// Blockchain database api result
{"id":2,"result":2}

// Blockchain database get object api call
{"method":"call","params":[2,"get_objects",[["1.2.35"]]],"id":1}

// Blockchain database get object api result
{“id”:1,”result”:[{“id":"1.2.35","registrar":"1.2.15",...}
```

Read from DB

TYPESCRIPT

```
import { DCoreSdk } from "dcorejs-sdk";

const api = ... // Initialized API

// Returns Observable

api.accountApi.getByName("some name");

// If Promise required

api.accountApi.getByName("some name").toPromise();

// Other calls

api.contentApi
   .get(ChainObject.parse("1.2.3"));
api.balanceApi
   .get(ChainObject.parse(“1.2.3"), [“DCT","ALX"]);

// More …
```

KOTLIN

```
import ch.decent.sdk.DCoreApi
import ch.decent.sdk.DCoreSdk

val api = ... // Initialized API

// Returns Observable

api.accountApi.get("some name")
api.accountApi.get(account.objectId)

// If blocking call required for JVM (Servers)

val account = api.accountApi
    .get("some name")
    .blockingGet()

// Other calls

api.contentApi
   .get("1.2.3".toChainObject())
api.balanceApi
   .get(account,"1.2.3".toChainObject())
```

DECENT

Write to Blockchain

TYPESCRIPT

```
import { DCoreSdk } from "dcorejs-sdk";

const api = ... // Initialized API
const op = new AccountCreateOperation(data)

// Broadcast is using TX API
// Broadcast op with TX confirm result (only Socket)

api.broadcastApi.broadcastWithCallback(private key, op);

// Broadcast op without TX confirm (Https, Socket)

api.broadcastApi.broadcast(private key, op);

// Shortcuts

api.accountApi.transfer(…);
api.messagingApi.send(…);
```

KOTLIN

```
import ch.decent.sdk.DCoreApi;
import ch.decent.sdk.DCoreSdk;
Import ch.decent.sdk.model.*

val api = ... // Initialized API
val op = AccountCreateOperation(data)

// Broadcast is using TX API
// Broadcast op with TX confirm result (only Socket)

api.broadcastApi.broadcastWithCallback(private key, op)

// Broadcast op without TX confirm (Https, Socket)

api.broadcastApi.broadcast(private key, op)

// Shortcuts

api.accountApi.transfer(…)
api.messagingApi.send(…)
```

DECENT

Thank you!

mike@decent.ch

Links

https://docs.decent.ch

http://github.com/DECENTfoundation