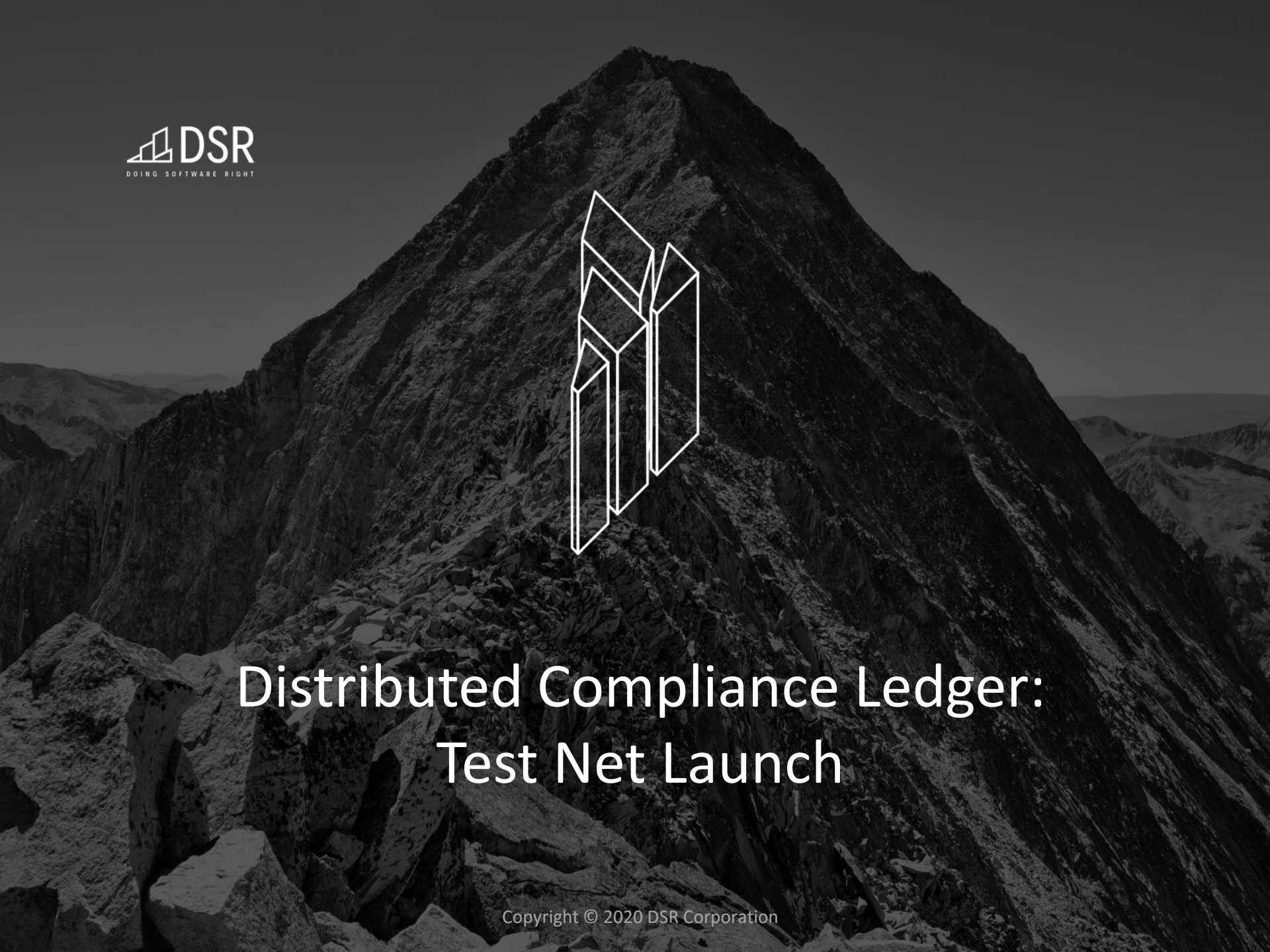


Distributed Compliance Ledger: Test Net Launch

Agenda

- About the Project
- Test Net Launch

About the Project

DSR
DOING SOFTWARE RIGHT

About The Project

Two groups of use cases:

- Device Models (created by Manufacturers) and Compliance tests (created by Testers and Alliances)
- Public Key Infrastructure

Makes device attestation and certification process more

- Simple
- Automated
- Secure
- Transparent and trusted

DSR
DOING SOFTWARE RIGHT

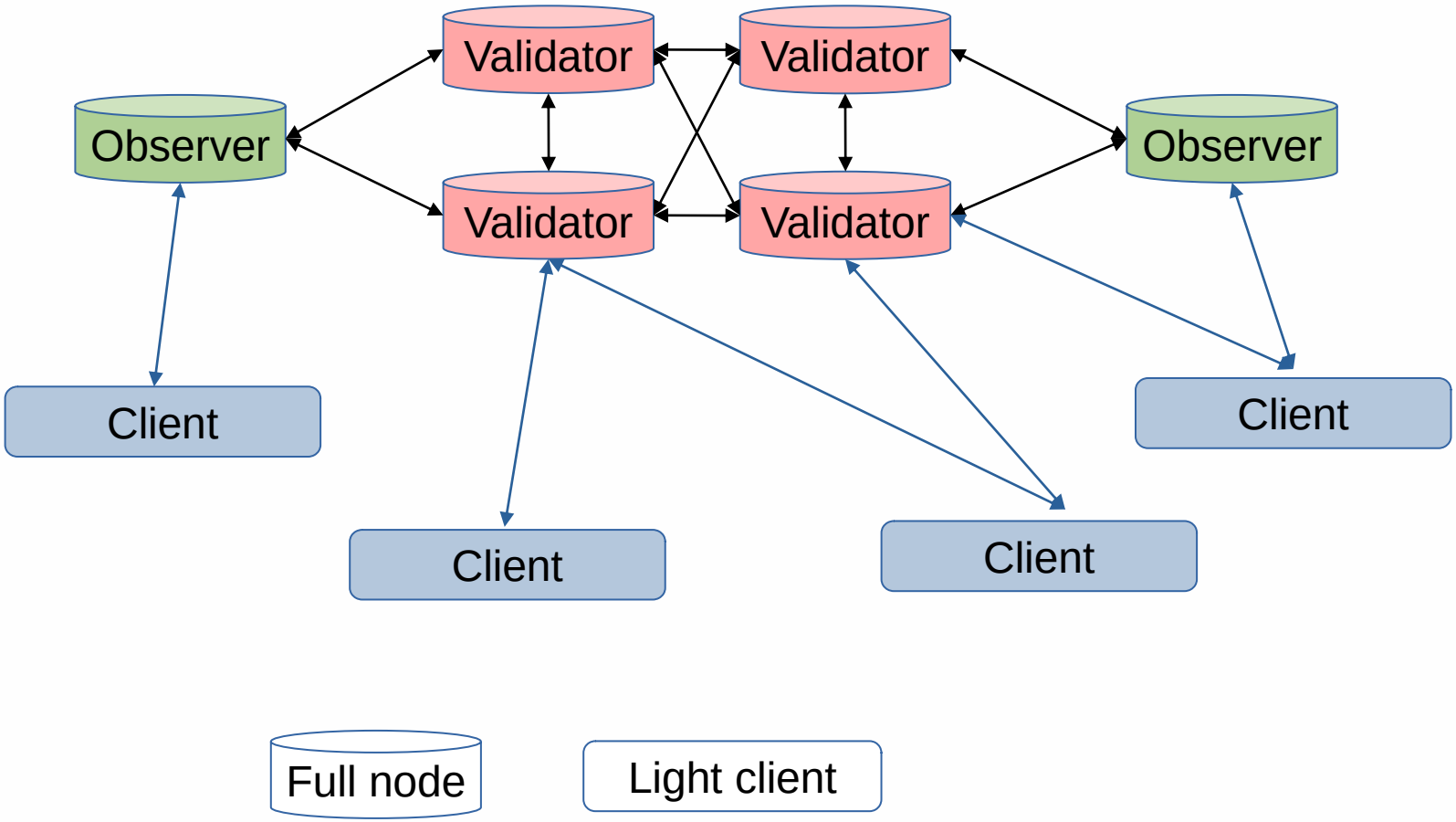
About The Project

- Public Permissioned Distributed Ledger owned and hosted by CHIP and ZB Alliance members
  - Write access to the Ledger is permissioned and restricted
  - Anyone can read from the Ledger

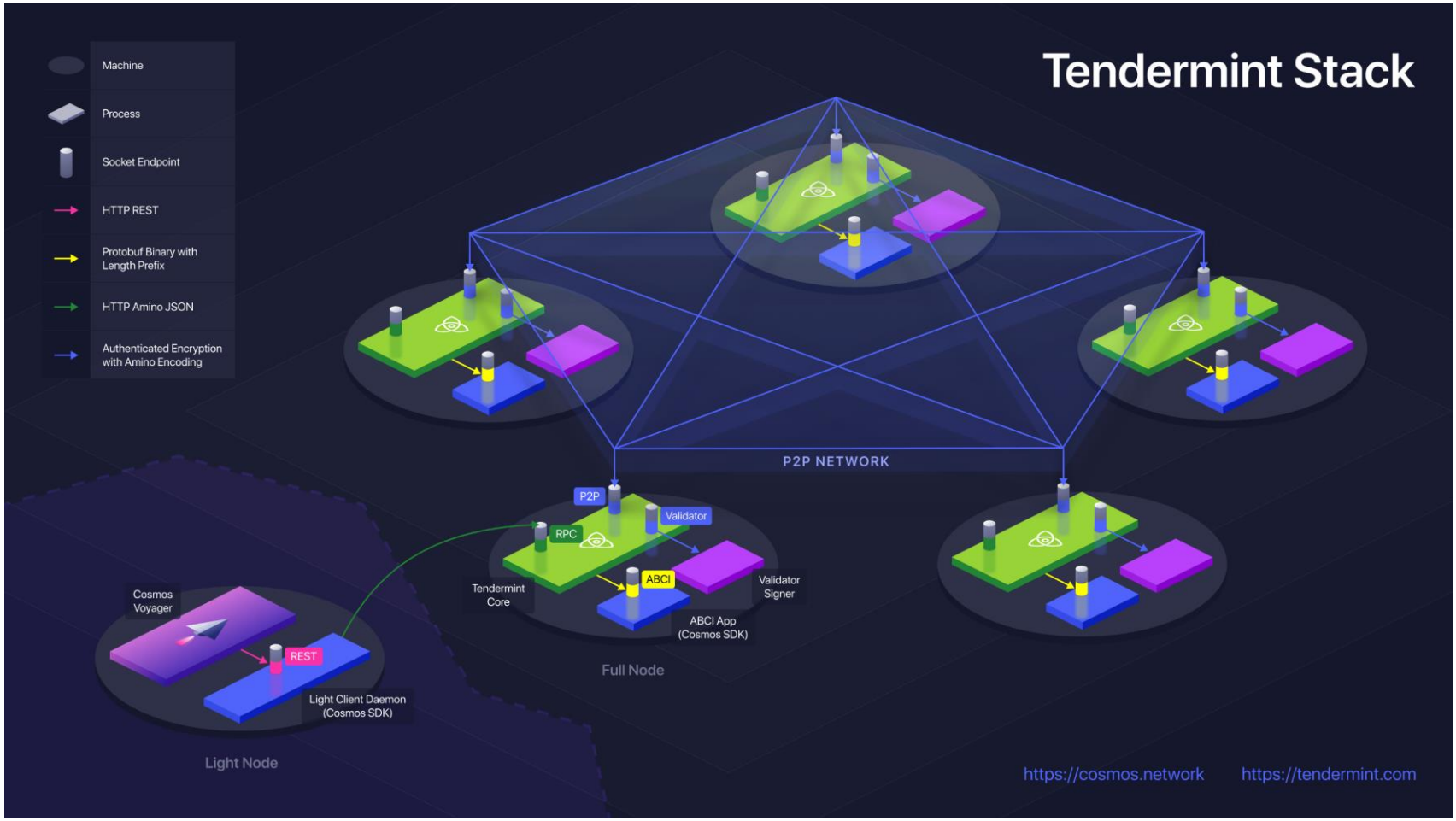
About The Project

- https://github.com/zigbee-alliance/distributed-compliance-ledger
- Open source (Apache 2.0)
- Core logic is implemented on top of Tendermint and Cosmos SDK

Network topology



Network topology (different view)



Network topology: Node (Replica) types

- Validator Nodes
  - Have a full copy (replication) of the ledger
  - Permissioned, not anyone can be a Validator Node
  - Total number of nodes should be limited
  - Initial nodes – genesis; number of nodes can be extended
- Observer Nodes
  - Have a full copy (replication) of the ledger
  - Anyone can have an Observer node

DSR
DOING SOFTWARE RIGHT

Network topology: Clients

- Anyone writing or reading
- Client != Node
- Don’t need to maintain or have a full Ledger
- Number of clients is not limited
- Can read from 1 Node only
- Can connect to a node/replica he doesn't fully know and trust, as there are crypto proofs for the replies
- Write access to the Ledger is restricted (write permission needs to be granted)
- Read access to the Ledger is public (anyone can read)

Client Types

- CLI
  - Run on a user's machine
  - Connected to one of the nodes (either a Validator or Observer)
- REST API
  - Backend server: either a local deployment per organization/application/user, or a global
  - The backend server is connected to one of the nodes (either a Validator or Observer).
- Client Libs / API Used by the application itself
  - Core API is there, but language and platform-specific libraries - TBD

Usage Scenarios

1. The user's organization doesn't have any nodes/replicas in the network.
    - 1A: CLI connected to one of the nodes in the network.
    - 1B: REST API against a backend deployed at the user's organization (or an organization the user trusts). The backend is connected to one of the nodes in the network.
    - 1C: The user's application calls the API lib connected to one of the nodes in the network
2. The user's organization runs a validator node.
    - 2A: CLI connected to the validator node.
    - 2B: REST API with a backend server connected to the validator node.
    - 2C: The user's application calls the API lib connected to the validator node.
3. The user's organization runs an observer node(s).
    - 3A: CLI connected to the observer node.
    - 3B: REST API with a backend server connected to the observer node.
    - 3C: The user's application calls the API lib connected to the observer node.

User Roles

- Trustee
  - create new user accounts
  - assign roles to the account
  - revoke roles from the account
  - approve X509 root certificates
- Node Admin
  - add a new Validator node

User Roles

- Vendor
  - publish device model info
- Test House
  - publish compliance test results
- ZB Certification Center
  - certify or revoke certification of device models

Test Net Launch

Test Net

- The first official deployment of DCL
- Can be used for demos, testing, PoC
- This is not a Production Net yet, so should not be used in production use cases

DSR
DOING SOFTWARE RIGHT

Availability And Persistence

Assumptions for Test Net:

- Should be available as close to 24/7 as possible. Possible exceptions are maintenance, found issues and update of software. In general, running the Test Net can help us to understand the availability for the production net.
- The data (ledger) should be persisted and not lost during the lifetime of the Test Net.
  However, we may clear the data or restart the Network if an issue is found, or during a software update.
- We may have breaking changes during further development.

For the Production Net the assumptions should be more strict.

Hardware Requirements

Recommended for Test Net:

- 2GB RAM
- 100GB SSD
- x64 2.0 GHz 2v CPU

https://docs.tendermint.com/master/tendermint-core/running-in-production.html

Deployment Specific

- For Test Net the recommended deployment option is to deploy on cloud (AWS for example)
- For Production Net we should consider
  - A more secure and DoS protected Sentry Node Architecture: https://docs.tendermint.com/master/tendermint-core/validators.html
  - HSM: https://hub.cosmos.network/master/validators/security.htm

Update software code

Short-term approach:

- Manual update of the software
- We may have to clear the data during the update

Long-term approach:

- Can handle breaking changes and possible migrations of data
- Based on https://github.com/cosmos/cosmos-sdk/tree/master/cosmovisor
- The process will look as follows:
  - Trustees need to approve the Update transaction specifying the location of a new version and the update time
  - Update and migration will be done automatically by the system at the specified time
- Detailed instructions will be provided

Node Admin Responsibilities

- Install the software correctly
- Run the correct version of the software (update or support updates when needed)
- Make sure that the Node is always online
- Make sure that the private keys are not compromised
- Make sure that the Node participates in consensus
- Monitor the Node health
- Participate in troubleshooting if needed

Trustee Responsibilities

- Make sure that the private keys are not compromised
- Send and approve Update transactions
- Approve X509 root certificates
- Create and approve new user accounts
- Approve revocation of user accounts, root certificates and validators if needed

Currently 2/3 of Trustees must approve every action before it’s committed.

DSR
DOING SOFTWARE RIGHT

Committed Participants

- Trustee
  - ZB Alliance
  - DSR
  - Comcast
- Node Admins
  - DSR (2 nodes) – AWS West and Europe
  - Comcast (2 nodes) – AWS East and West
  - Exegin (1 node)
  - ZB Alliance (1 node) – In future

DSR
DOING SOFTWARE RIGHT

Test Net Launch

- Date:
  - 10/01 – DCL Overview; Discuss the Test Net Launch steps
  - ~2d week of October – Test Net Launch
- Communication Channel:
  ZigbeeAlliance Slack, #zigbee-dcl
- Test Net release artifacts (v0.4):
  https://github.com/zigbee-alliance/distributed-compliance-ledger/releases/tag/v0.4

Test Net Launch Procedure

1. DSR starts up the first (genesis) node and creates the genesis transaction block. The genesis has 1 Trustee account owned by DSR.
2. DSR starts up the second node (approved by DSR Trustee).
3. Comcast starts up the third node (approved by DSR Trustee).
4. Comcast creates a Trustee account (approved by DSR Trustee).
5. Comcast starts up the fourth node (approved by DSR and Comcast Trustees).
6. Exegin starts up the fifth node (approved by DSR and Comcast Trustees).
7. Zigbee Alliance creates a Trustee account (approved by DSR and Comcast Trustees).
8. Zigbee Alliance starts up the sixth node (approved by DSR, Comcast and ZA Trustees).

These six nodes will be considered as persistent peers for other participants (clients and other validation nodes).

Instructions

- The first (genesis) node creation: https://github.com/zigbee-alliance/distributed-compliance-ledger/blob/master/docs/running-genesis-node.md (will be done by DSR only)
- Other nodes creation: https://github.com/zigbee-alliance/distributed-compliance-ledger/blob/master/docs/running-node.md
- Trustee Account creation: https://github.com/zigbee-alliance/distributed-compliance-ledger/blob/master/docs/how-to.md#trustee-instructions

Questions And Other Items