2.2.2.3 Node Architecture

return to Node Network View

The DIDO node architecture establishes a component model as a reference for evaluating the functionality or data available within a DIDO implementation. The reference components are conceptual in nature; thus they may or may not represent actual components within a DIDO implementation. However, the functionality of these reference components should be part of the implementations.

At the most elementary level, reference components provide a common vocabulary for DIDO implementations. For example, Bitcoin does not have a separate standalone component that performs secure messaging; however, the Bitcoin software *does* provide secure messaging functionality using a protocol built upon cryptography. This binds the secure messaging and protocol together. It is possible to write applications that adhere to this protocol using the cryptographic rules and standards defined by Bitcoin and not use the Bitcoin software; however, the usual approach is to build upon the open source Bitcoin software and use it "as-is." Another example is where Bitcoin provides an actual component referred to as a Wallet, bearing in mind there are numerous other wallets available that can contain and manage Bitcoins.[1)]

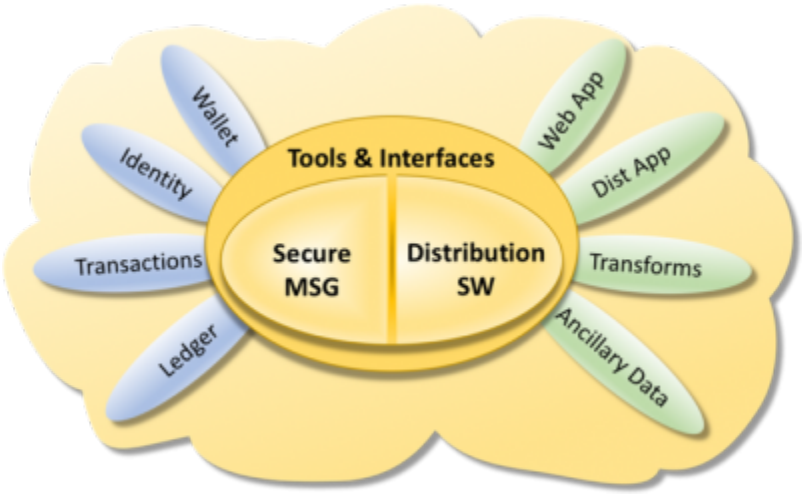

Figure 1: The DIDO Node Component Model

The DIDO node component model presented in this diagram (Figure 1) may look like a traditional stack or layered architecture as described in centralized or decentralized models; however, a very important difference is that the components or a subset of the components of this architecture are usually repeated at every DIDO node that participates in the DIDO network. At a minimum, the secure message and the distribution software components must exist at each node in order for the network to remain distributed and operational. In addition to providing for the core components required to securely communicate, a DIDO node can take on different roles or functions (refer to Figure 2). Some nodes may use all the components within the DIDO node component model while others may only use a subset of these components. For example, a smart contract node may use the identity, transaction, ledger, distributed app, and ancillary data components. In contrast, a Wallet node may just use wallet and identity components.

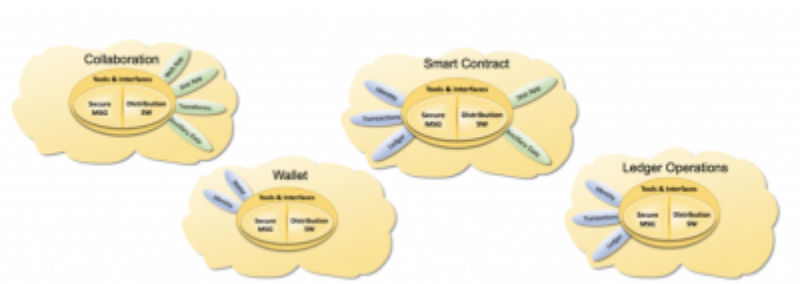

Figure 2: Examples of kinds of DIDO Nodes

When the entire DIDO network is assembled, it may look something like Figure 3. Each node has the basic common core components and the set of components required for it to fulfill its specific responsibilities.

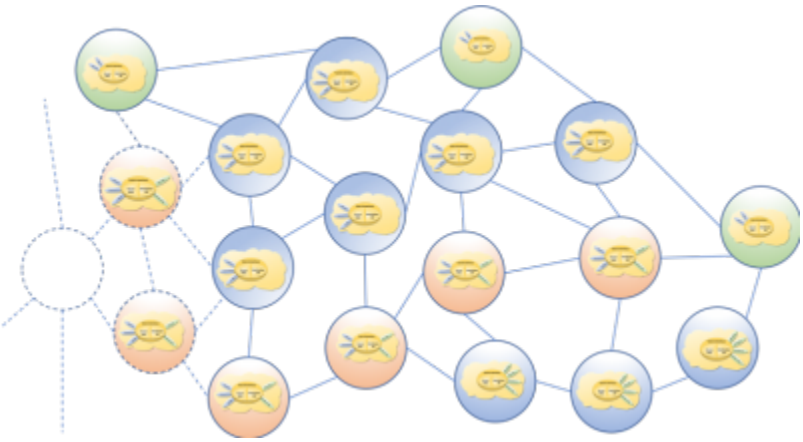

Figure 3: The DIDO Network Model

The DIDO network acts as a single system or entity, in which each node performs the operations required of it as a participant. For example, some DIDO networks may only provide a ledger, account numbers, and support transactions on the ledger. These would require each node to have the secure message, distribution software, ledger, identity and transaction components in common.

All the components within the network must be interoperable at the core functional level. For example, most of the nodes might run the Common Core version 1, whereas a subset might be running version 1.1. As long as the nodes within the network can interoperate and function together, the network is considered viable as a whole.

The interoperability of nodes within the DIDO and overall DIDO network viability become key benefits of DIDO implementations. This means that anything affecting interoperability is critical, and requires that interfaces and interactions (i.e., protocols) between the nodes must be the most conservative of all components. In other words, interfaces and interactions of the nodes are the mission critical aspects of the network and, therefore must be stable from the onset. Interruption or discontinuity in the critical path could result in a fork of the underlying ledger. An example of a fork is the "hard fork" in Ethereum called Byzantium:

> *The Byzantium hard fork is an update to ethereum's blockchain that was implemented in October 2017 at block 4,370,000. It consisted of nine Ethereum Improvement Protocols (EIPs) designed to*

improve ethereum's privacy, scalability and security attributes.[2)]

Common Core

The Common Core contains components that are used by both the ledger and ancillary data subset of components, as well as characteristics and attributes that apply to all components within the DIDO network. There are three classes of common components: tools and interfaces, distribution software, and secure messaging. Common components can be used exclusively by one kind of DIDO node as defined in node architecture but can also span across the components within a node. For example, the transaction Application Programming Interface (API) is available to both the ledger and the ancillary data node and potentially by some of the tools. However, the transaction API may or may not be used as part of the ancillary data stack.

By definition, a ledger operation node must rely on the transaction API to access the ledger; however, a smart contract node may also require access to the ledger. In that case, its access is made exclusively through the transaction API. It is up to the implementation of a particular DIDO whether to access its ancillary data through a transaction API or a transform API. Conversely, a transaction might require ancillary data (i.e., monetary exchange rate, stock quotes, interest rates, market cap, or Beta, etc.) in order to complete. The Common Core contains an API that defines and allows for this access, sometimes referred to as an oracle.

Note: Transaction API and Transform API are defined in more detail in Section 2.2.2.4 Messaging View.

- 2.2.2.3.1 Immutable Data Objects
- 2.2.2.3.2 Ancillary Data
- 2.2.2.3.3 Semantic Web
- 2.2.2.3.4 Software

1)
A. Hertig, "Does the original Bitcoin Wallet Still Matter?," 16 September 2016. https://www.coindesk.com/original-bitcoin-wallet-still-matter/.

2)
R. Sharma, "What is the Byzantium Hard Fork in Ethereum?," 7 March 2018. https://www.investopedia.com/news/what-byzantium-hard-fork-ethereum/