DID Fluidity

by Juan Caballero jcaballero@centre.io

In Wayne Chang's excellent recent blog post about "Upgradable Decentralized Identity", he writes extensively about the DID Trait system, a more formale exploration of which is presented for RWOT11. Near the end, he mentions upgrade paths for blockchain PKI systems and DID methods, "trading up" and delegating into complex arrangements of control and inheritance. The terminology felt fresh, which was worrisome given how much I have read and written over the years about these topics. Rather than address directly the examples sketched out there, I thought it might help to establish a framework and foundation to supplement this upgradeability idea, the composable-traits idea, and the registries and governance issues also pointed to by many other advanced readings submitted so far.

A paper worth crowd-sourcing, refining, and formalizing at RWOT would be to establish for [D]PKI identity systems a vocabulary or typology of how DID methods or DID-like/DID-compatible identifier schemes could *interrelate*. Much writing and thinking about DID systems tends to treat them in isolation, as function or even load-bearing systems which fundamentally use only one DID method integrally– if other methods are mentioned, it is often as a secondary translation issue. What I would like to develop is a vocabulary for some terms and, ideally, a quick survey of known example of as many of them are currently represented in linkable implementations.

Glossary

Initial list of loose categories to be refined: * Cross-Method Portability - Does DID method A have a mechanism for its controller to deactivate their DID and get another on DID method B? Portability is mentioned in the DID spec, but it is not clear if this refers to user portability across vendors/infra, to and from DIDs from other identifier systems, and/or across DID methods. * Translation - Has anyone documented and tested a lossless, roundtrip translation between the DID Documents of two different methods? * Export - Do any known DID systems produce conformant DID Docs according to multiple DID methods? I.e., could a system which natively registers and manages DIDs according to method A also have a resolution mode that returns method-B DIDs? * External control - In blockchain design, there are many forms of "externally controlled" wallets, proxy wallets, and meta-wallets which wrap or contain wallets, whether on different blockchains or multiple wallets on one blockchain. The `controller` property is defined in the DID specification, with a few warnings about what not to assume, but thinking through and making explicitly assumptions (or how it might interact with other patterns and traits) might be worth doing at length. * Equivalence - Similarly, the section on the `alsoKnownAs` property recommends enforcing/checking the *reciprocity* of `alsoKnownAs` reference, but does not dwell on what possible use-cases or assumptions might be made in the

alternate case of intentionally unidirectional equivalence: which is deprecated, which is primary? * Patching and local DID Documents - In some systems like Sidetree, one base version of a DID Document might be published globally or considered authoritative, yet in certain local environments some form of patch, supplement, or delta could be applied that modifies the Document according to a given resolution path or for a given audience. * Delegation - What happens when a DID Document from method A includes a DID from method B as its authentication key, for example? The `controller` property only considers one or more DIDs to be a valid controller, but what if a DID is controlled by a pseudo-DID method like `did:pkh`?

Form Factor

Each of these could be a whole paper some day, but the goal for RWOT11 of a group wants to engage and work on this topic would be to write a one-sentence definition and a paragraph or two of considerations for each of the above (in some cases, the term might bear splitting into subtypes that each get this treatment) and collecting links and "to-dos" for ongoing research.