Blockchain PSIG Call Notes

*3 Feb 2021*

# Attendees

* Mike Bennett
* Rob Nehmer
* Lars Toomre
* Nick Stavros
* Bobbin Teegarden
* Robert Rencher
* Ian Stavros

# Agenda

* DIDO-RA

# Meeting Notes

## DIDO

On screen walk through of wiki pages.

### Taxonomy – Data Taxonomy

Cognitive, state, types of data, data modelling taxonomy and so on.

Walks through a number of concepts related to what a programmer needs to know about in creating smart contract code. Examples from Ethereum. Solidity language.

### Object Data Taxonomy

Defines the concepts, the into DIDO and DIDO-specific

Q: What are you talking about as object e.g. OO?

NS: Most people can relate to objects e.g. characteristic data. What are these in terms of smart contracts.

So close to OO but not quite. Trying to highlight the differences in these pages. Intended audience includes those who are familiar with OO.

Q: Are these DIDO specifics just Solidity or have you abstracted SC concepts in general?

NS: All the examples so far as Solidity as we have only been able to look at Ethereum SCs.

e.g. have the concept of Attributes but poor on Properties.

Functional v Procedural.

e.g. OO 'Method' such as Constructor, Destructor

Solidity has self-destructors but does not have Destructors.

Similar for Getters and Setters.

### Comments

Some of this is in response to the SC RFI (the original one)

e.g. can I have the equivalent to things that are Abstract, in SCs? Yes, they are called something else – Base Class and Derived Class.

e.g. Employee is Person and Parson is the abstract class.

Anyone familiar with C++ can identify the concepts of relevance

Considering SysML and systems – the Abstract classes become important.

### ERC20

These are basically interfaces. The other ERC are all interfaces, e.g the ones for NFT.

So a SC that is ERC20 compliant means you are filling in all the functions needed t be ERC20 compliant.

### Special Methods

e.g. how some languages let you override multipliers, etc.

e.g. for matrix operations.

### Solidity Math Ops

Solidity has unsafe math operations so there is a library called Safe Math to provide overflow, underflow and wrap-around.

Everything is usually uint256 s unlikely to get overflows but can get underflows.

Very large data sets can cause overflow- but doesn’t pay to have larg data sets in Blockchain. Safe Math does a wrap-around for this scenario. So for uint256 use Safe Math

So a lot of this is v familiar to OO programmers.

### Gas

Important factor is knowing when you have to pay gas or don't have to pay gas.

### Visibility

e.g. Public, Protected and Private in classic OO languages.

ETH has the same thing but under different names: Public, Internal Private and a separate ting called External.

External: when you declare a Function to be External is starts a Txn. Is the trigger for Txn. When it is finished the Txn is complete. This is unique to DLT.

Consider that DLT is a hybrid between database and OO.

External Contract – see diagram on visibility kinds. On the Visibility page.

Functions == Method in OO. Can call a Public method in another Contract.

### Data Memory Taxonomy

Stacks, Heaps etc.

Construction data

Constants (literal, named)

Then again to the DIDO Specifics.

Shows how you would do a Constant in Solidity, how to do a Literal Constant in Solidity, Java Immutable strings etc. Syntactical differences like single v double quotes.

### Data Taxonomy

 – see Conceptual, Logical and Physical models.

### Data Lifecycles

Can only share never archive or destroy.

Archival nodes – can have some of this – see discussion on this page e.g. pruned and archival nodes.

Nothing is descried but maybe pruned or archived.

### Digital Assets

e.g. coin, FT or NFT.

### Tokens

Asset, payment etc. tokens.

No non-DIDO section on these. Only exist in DIDOs.

### Network Access Control Taxonomy

Whatever you put on a blockchain is publicly accessible.

Implications:

### State of Data Taxonomy

* Data on motion
* Data at ret
* Data in use

Sub-sections for each.

Blockchain public = no physical security

Q: In motion v in use?

* In motion – on the wire between nodes
* In use: I decrypt the data and am actually processing it

Data in Use – security issues etc.

Compared to e.g. mainframe or cloud based data. Can control access t the core, data bus etc.

Q: So on the wire v in use – difference is the encryption status

NS: Basically yes, if encrypted it is in motion

MB: Is that definitional or merely a symptom that something is DiU or DiM?

NS: encryption e.g. encryption on disk etc., secure chips. See IBM Hyperledger. Need a special chip in the machines they were using. Machine has to have the encryption.

### Multi-party encryption

Q: Are you capturing who does what to whom? Eg. IBM Hyperledger – who else does that. Is NS capturing these differences for all of these?

NS: Not at present. Once we have the taxonomy we can evaluate any platform on these distinction. This is the basis for Functional Requirements e.g if someone needed platform security.

For example if the Fed wanted to use this, would use these terms to describe the machines they want to run on. Probably not doing that right now e.g. have they distinguished DiM, DaR (e.g. need for full-disk encryption)

We have to be able to work with the people who need t specify what they want to specify any crypto currency for e.g. a country.

## Summary

Been a lot of work.

Some pages will go away.

Some of this needs more work

BT: Almost need a spreadsheet for these platform-specific distinctions.

NS has that for some things e.g. algorithms (e.g. Consensus) by platform

e.g. Delegated Proof of State and the rest

See table with pros and cons, collated from various sources.

BT: IF you came in and needed to figure out what kind of blockchain you wanted to us, could use this

#### Potential OMG Activities

NS: OMG could start defining these thigs

* Functional
	+ Public v Private
		- For each, what restrictions do you need on (various things)

**Importance:** get this in front of Government people who have to write requirements for a system.

Rather than simply choosing between one or another ecosystem, start with the requirements, using the terms set out in this wiki.

B: would like to be able to sot through what ar the major platforms that do these things.

NS: IF I were government, would do an RFI asking questions in terms of these distinction e.g. what are you doing in terms o data in motion. Use these definitions. Not perfect but a useful starting point.

So a spreadsheet of eval criteria, a DB to let you filter through and find the ones the want. Gov should not need to do the research on the different platforms but give them something to fill out based on the concepts defined here.

BT: Gives them a structured way of thinking about blockchain. Impression is they are not thinking this kind of way.

Comes back to Agile – avoiding over-specification.

### Security etc.

Ask e.g. what are you doing to protect Data in use - is asking the question, not over-specifying. Have defined what these things mean. Gov or end user can determine that maybe they already have physical security so don't need that in the solution.

These are the things you should be thinking about.

Q: How much of this is generic for any architecture?

NS: the generic parts are for any architecture, based on experience.

Use as checklist.

e.g. TRESOR (page)

Describes a specific approach.

#### Protection Rings

e.g. Ring 0 s where only the Kernel can get to it.

Additional rings e.g. Hypervisor, system management mode, management engine.

Q: In blockchain what do you call the Hypervisor?

In Blockchain I can install a VM, can be a hypervisor on a node, maybe can snoop on certain things.

e.g. can someone put a Bitcoin node on a Hypervisor node and snoop on the internal e.g. private keys.

BT THE OS would deal with those things surely.

NS: on old single stack architecture.

BT: Blockchain is app running on top of the existing things – like a nested version of Hypervisor (e.g. the BC has its own version of a hypervisor) distinct from what's there lower in the stack Is nested.

App or OS level?

Hypervisor is above the OS level.

Distribute something as a VM or (something) e.g. in a Container, now that Container has an OS.

So Blockchain inside Contain on an OS = triple nesting.

People need to be asking the questions on these things. Government ( or other end user) needs to be able to write things down. Rather than simply asserting 'W need Bitcoin' or similar.

Need to address the layering / security problem if it is possible to get under the layers fo the Blockchain level.

e.g. Data Logging. Error Log typically not encrypted.

Solidity 'Events' are like exceptions.

BT: How does it address UDP from the network side?

NS: Hence DDS Security is important.

But no Blockchain Architecture is using DDS right now. Uses e.g. Gossip protocols.

AS per how core network (IP) uses a Gossip protocol.

6 - 7 layers have security considerations all the way up.

See page on Gossip Protocol and the backlinks to where this is mentioned.

### Potential OMG Activities

Char will come up with a PDF of this material (some time in April). Will be large.

Then from Gov / user perspective will need to know who is doing which pieces and how.

e.g. IBM – what they are using and how e.g. network disk encryptions, dedicated encryption chips. Ned to be able to specify all these things, then compare with e.g. the IBM solution and compare prices versus functionality. So can then do cost / benefit analysis.

We cant' tell Gov (users) that, just give them the questions

MMB: NS keeps saying only 'Government' – confirm he means any end user?

Yes. Was thinking about CBDC but can also apply to Supply Chain end users lie e.g., Boeing.

e.g. if Boeing forms an ecosphere via a Memorandum of Agreement, that Memorandum can define requirements in the terms given here.