Today, we sketched out plans for the node's graphical inspectability capabilities (I don't have great terms for referring to this, sorry. The GUI, but not what that word brings to mind right away.)

Taking existing Observer views as all given, we discussed some of the following at length:

- A node-based view, i.e., a sort of high level dashboard for node parameters, node configuration, editing node configuration, &c.
- Bespoke individual engine views that go beyond simple process state dumps even if the information

is present if you dump enough of them, such as: * Storage view, showing current and historical states in their logical, not their compressed physical data structure form.

- Transaction-oriented view, showing the mempool of pending transactions, their execution state (e.g. running, blocked, completed)
- Block-oriented view, looking from the perspective of the transactional system checkpoints known as "blocks".
- Event-oriented view, showing the replayable event log in an easy-to-read format, allowing you to roll back, roll forward, and inspect.
- Engine constellation view, showing the engines that make up the node at a higher level than inspecting their processes.

- Storage view, showing current and historical states in their logical, not their compressed physical data structure form.
- Transaction-oriented view, showing the mempool of pending transactions, their execution state (e.g. running, blocked, completed)
- Block-oriented view, looking from the perspective of the transactional system checkpoints known as "blocks".
- Event-oriented view, showing the replayable event log in an easy-to-read format, allowing you to roll back, roll forward, and inspect.
- Engine constellation view, showing the engines that make up the node at a higher level than inspecting their processes.

- Resource views; not sure to what extent the shielded RM is inspectable at runtime but the transparent RM is made to be so.
- Data structure views: you should be able to graphically inspect structs, basically.
- Columnizable protocol: oftentimes it's clearer to look at a collection of structures as a table, rather than a bunch of nested structs. I have a clever design for this which deserves its own thread.

- Columnizable protocol: oftentimes it's clearer to look at a collection of structures as a table, rather than a bunch of nested structs. I have a clever design for this which deserves its own thread.

- Transport view: network activity, foreign connections, and the things discussed in the transport thread
- Anything else! The collection of views is a wide open set.