The v2 specs must include a first draft of the application state encoding format, which we should use to encode resources, data within resources, signatures, etc.

Basic requirements:

- Mapping from algebraic data types
- As defined in Juvix or something likeTypedefs
- As defined in Juvix or something likeTypedefs
- Deterministic (a structure S

always serializes to the same bytes B

).

- Given a type, unique (some bytes B

, given a type T

, always deserializes to the same structure S

)

- Merkleizable with sub-structural verification and efficient updates
- A structure S

has a canonical Merkle root M

- Given some sub-component S'

of S

, S'

can be verified to be part of S

with cost no more than O(log(size(S)))

.

- Given a change to some sub-component S'

of S

, the new canonical Merkle root can be computed with work at most proportional to O(size(S') + log(size(S)))

- A structure S

has a canonical Merkle root M

- Given some sub-component S'

of S

, S'

can be verified to be part of S

with cost no more than O(log(size(S)))

.

- Given a change to some sub-component S'

of S

, the new canonical Merkle root can be computed with work at most proportional to O(size(S') + log(size(S)))

- Reasonably compact (slight overhead is acceptable)
- Canonical representation of any structure S

as JSON (this should be easy)

Options:

SSZ option

With this option, we would use Ethereum's well-specified SSZ format, plus:

- a mapping from algebraic data types to SSZ, and from SSZ to algebraic data types
- perhaps some changes (additions/deletions) to SSZ basic types
- perhaps some changes to make the SSZ spec more abstract (e.g. parameterize out basic types, rethink some of the container/vector logic)

Discuss!