Immutable Data Object

Go up to Terms and Definitions

Discussion

An Immutable Data Object preserves each committed state without altering that state in place.

After a state becomes committed, an authorized change creates a successor state rather than replacing, overwriting, or modifying the committed state. The predecessor state remains independently identifiable and verifiable.

An Immutable Data Object can preserve relationships among successive states. These relationships can establish:

- The identity of the predecessor state
- The identity of the successor state
- The order of the states
- The change represented by the successor state
- The authority responsible for the change
- The time associated with the change
- The Evidence supporting the change
- The Provenance of each state

Immutability applies to an established state of a Data Object. Immutability does not require the real-world subject represented by the Data Object to remain unchanged.

For example, an asset can change ownership. An Immutable Data Object can represent the change by creating a successor ownership state while preserving the preceding ownership state.

An implementation can protect an Immutable Data Object through one or more mechanisms, including:

- Append-only storage
- Content addressing
- Cryptographic hashing
- Digital signatures
- Chained state references
- Write-once storage
- Replicated state
- Distributed agreement
- Access controls
- Tamper detection

The concept does not require a particular protection mechanism.

Definition

data object that preserves each committed state without in-place alteration

Source

Adapted from:

- DIDO Reference Architecture, Immutable
- DIDO Reference Architecture, Data Object
- DIDO CLI, What is DIDO?
- Distributed Immutable Data Object (DIDO)

The definition preserves the DIDO-RA principle that recorded data cannot be changed while distinguishing in-place alteration from the creation of a successor state.

Note

An Immutable Data Object is not necessarily a Distributed Immutable Data Object (DIDO).

An Immutable Data Object becomes a DIDO when multiple networked Nodes maintain its committed states through a shared state-agreement mechanism.

Immutability does not mean that:

- The represented subject never changes
- A correction cannot occur
- A new state cannot be created
- Every user can access the state
- The state must remain available forever
- The Data Object must use blockchain
- The Data Object must be distributed
- Every copy of the Data Object must use the same physical representation

A correction to an Immutable Data Object creates a successor state that identifies or supersedes the incorrect state. The correction does not overwrite the incorrect state.

Immutability and retention address different concerns:

- Immutability concerns alteration of a committed state
- Retention concerns how long an object or state remains preserved and available

Immutability and confidentiality also address different concerns. An Immutable Data Object can remain encrypted or access-controlled.

Example

An Immutable Data Object records that an asset belongs to Party A.

A later authorized transfer assigns the asset to Party B. The transfer creates a successor state identifying Party B as the owner and referencing the preceding Party A state.

The system preserves the Party A state without alteration. The two states and their relationship provide a verifiable history of the ownership change.
