Internet-Draft
Intended status: Informational
Expires: November 2026

M. Norton
Independent
May 2026

SDLP RFC 2: Lifecycle State Machine
draft-norton-sdlp-lifecycle-00

M. Norton
Individual Submission
May 2026

Abstract

This document defines the lifecycle state machine for the Secured Digital Lifecycle Protocol (SDLP). The SDLP lifecycle governs the existence, transitions, and permitted operations of all SDLP objects. Each object MUST exist in exactly one lifecycle state at any given time, and all transitions MUST follow the deterministic rules defined in this document. The lifecycle state machine ensures predictable behavior, prevents invalid transitions, and provides a universal framework for object governance across SDLP implementations.

Status of This Memo

Copyright Notice

to this document.

1. Introduction

The SDLP lifecycle defines the complete set of states through which a digital object may progress. Every SDLP object MUST exist in exactly one lifecycle state at any given time. All transitions MUST be deterministic, authorized, and recorded.

This document defines:

* the SDLP lifecycle states
* the allowed transitions between states
* the rules governing state entry and exit
* the invariants that MUST hold within each state
* the security properties associated with lifecycle transitions

The lifecycle state machine is a core component of SDLP and is referenced by all other SDLP specifications.

2. Lifecycle Overview

SDLP defines seven lifecycle states:

1. Creation
2. Activation
3. Distribution
4. Transformation
5. Verification
6. Retention
7. Retirement

Each state represents a distinct phase of existence. Transitions between states MUST follow the rules defined in this document.

3. State Definitions

3.1 Creation

The Creation state represents the initial formation of an SDLP object. During this state:

* The DigitalID is assigned.
* The initial lineage is established.
* The canonical bitdump is generated.
* The seal is applied.

An object MUST transition out of Creation into Activation.

3.2 Activation

Activation represents the moment an SDLP object becomes usable. In this state:

* The object is bound to a CustomerID.
* The object becomes eligible for distribution.
* The object becomes eligible for transformation.

Valid transitions from Activation:

* Activation ? Distribution
* Activation ? Transformation
* Activation ? Retention

3.3 Distribution

Distribution represents the controlled dissemination of an SDLP object. In this state:

* The object may be delivered to authorized recipients.
* The object may be duplicated, producing descendant identities.
* The object may be exported or transferred.

Valid transitions from Distribution:

* Distribution ? Transformation
* Distribution ? Verification
* Distribution ? Retention

3.4 Transformation

Transformation represents any modification, conversion, or derivative operation performed on an SDLP object. In this state:

* A descendant DigitalID MUST be generated.
* Lineage MUST be extended.
* A new canonical bitdump MUST be produced.
* A new seal MUST be applied.

Valid transitions from Transformation:

* Transformation ? Verification
* Transformation ? Retention

3.5 Verification

Verification represents the validation of an object’s integrity, lineage, and seal. In this state:

* The seal MUST be validated.
* The lineage MUST be validated.
* The DigitalID MUST be validated.

Valid transitions from Verification:

* Verification ? Distribution
* Verification ? Retention
* Verification ? Retirement (if validation fails)

3.6 Retention

Retention represents long-term storage or archival. In this state:

* The object is stable and inactive.
* No transformations may occur.
* The object may be reactivated or retired.

Valid transitions from Retention:

* Retention ? Distribution
* Retention ? Verification
* Retention ? Retirement

3.7 Retirement

Retirement represents the terminal state of an SDLP object. In this state:

* The object is no longer active.
* The object MUST NOT be transformed.
* The object MUST NOT be distributed.
* The object MUST NOT re-enter any previous state.

Retirement is irreversible.

4. State Machine Rules

4.1 Determinism

All lifecycle transitions MUST be deterministic. No implementation may introduce nondeterministic or probabilistic transitions.

4.2 Exclusivity

An SDLP object MUST exist in exactly one lifecycle state at any given time.

4.3 Valid Transitions

Implementations MUST enforce the valid transitions defined in this document. Any attempt to perform an invalid transition MUST be rejected.

4.4 Transition Logging

All transitions MUST be recorded in the object’s lifecycle log, including:

* previous state
* next state
* timestamp
* actor
* justification

5. Security Properties

Lifecycle transitions MUST preserve:

* DigitalID integrity
* lineage integrity
* seal validity
* state invariants

Unauthorized transitions MUST be rejected.

6. Integrity and Tamper Resistance

Lifecycle transitions MUST NOT modify:

* the root DigitalID
* historical lineage entries
* previous lifecycle logs

Any attempt to alter lifecycle history MUST be treated as tampering and MUST cause verification failure.

7. IANA Considerations

This document makes no requests of IANA.

8. Security Considerations

Unauthorized transitions, forged lineage, or invalid seals MUST be treated as security violations. Implementations MUST ensure that all transitions are authenticated and authorized.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

Author’s Address

M. Norton
El Mirage, Arizona
United States
Email: mark433norton@gmail.com