Linkism Protocol Specification

Persistent UI Addressing Standard

Version: 1.0

Date: August 2025

Author: Joel David Trout II

"A web where elements outlive frameworks"

Table of Contents

1. Introduction

Linkism defines a protocol for persistent element identity on the web. LIDs decouple tests, agents, and RPA from brittle DOM structure via immutable contracts, signed bundles, and offline resolution.

This document consolidates the core LID URI specification (RFC-001) and deployment best practices (BCP-001) into a unified reference for implementers and adopters.

Modern web applications suffer from brittle element selectors that break during UI changes. Linkism solves this by providing a standardized URI scheme for persistent element identification that survives framework migrations, redesigns, and time.

1.1 Protocol Architecture

Protocol Flow:
HTML → contains → `lid://app.com/auth#login`
SCR Bundle → maps to → `button.primary`
Resolver → returns → CSS selector + confidence

1.2 Key Benefits

- **Framework Independence:** LIDs survive React → Vue → Angular migrations
- **Design System Resilience:** UI redesigns don't break automation
- **Cryptographic Integrity:** Signed bundles prevent selector tampering
- **Air-Gapped Resolution:** Works without network dependencies

2. Terminology

Term	Definition
`LID`	Linkism ID - persistent identifier for UI elements
`Authority`	Domain name responsible for LID registration
`Path`	Hierarchical context for element grouping
`Fragment`	Specific element identifier
`SCR`	Selector Contract Registry - bundle format for LID contracts
`Resolver`	System translating LID to current selector

3. LID URI Specification RFC-001

The LID URI scheme provides immutable addresses for UI elements that survive framework changes, redesigns, and time. Each LID consists of three components: authority (domain), path (context), and fragment (element identifier).

3.1 Core Syntax

```
lid://authority/path#fragment
lid://app.com/checkout#submit-button
lid://docs.site.com/api/v2#search-input
```

3.2 ABNF Specification

```
lid-URI    = "lid://" authority path [ "#" fragment ]
authority  = domain-label *( "." domain-label )
domain-label  = alphanum [ [ldh] *( alphanum | "-" ) alphanum ]
path       = [ "/" path-segment *( "/" path-segment ) ]
path-segment = *( unreserved | pct-encoded | ":" | "@" )
fragment   = *( unreserved | pct-encoded | ":" | "@" | "/" | "?" )

unreserved  = ALPHA | DIGIT | "-" | "." | "_" | "~"
pct-encoded = "%" HEXDIG HEXDIG
```

3.3 Character Encoding

- UTF-8 encoding MUST be used throughout
- Domain names MUST be lowercase (IDN normalization applied)

Reserved characters MUST be percent-encoded:

- # → %23
- / → %2F
- ? → %3F

3.4 Resolution Semantics

Resolution follows this process:

> LID URI → Parse Components → Verify Authority → Retrieve SCR Bundle → Locate Contract → Return Selector + Confidence

3.5 Key Properties

- **Immutable:** Authority, path, and fragment never change once registered
- **Hierarchical:** Path provides contextual grouping
- **Verifiable:** Optional cryptographic attestations for content integrity
- **Framework-agnostic:** Works across React, Vue, Angular, etc.

> **Implementation Note:** LIDs decouple test automation and UI tooling from brittle CSS selectors that break during redesigns.

4. SCR Bundle Format RFC-002

Selector Contract Registry (SCR) bundles are signed JSON documents that map LIDs to current CSS selectors. They enable offline resolution and provide cryptographic guarantees about element contracts.

4.1 Bundle Structure

```
{
  "spec_version": "1.0",
  "manifest": {
    "authority": "app.com",
    "generated_at": "2025-01-01T00:00:00Z",
    "ttl_days": 365
  },
  "contracts": [
    {
      "lid": "lid://app.com/auth#login",
      "selector": "form#login > button.primary",
      "confidence": 1.0,
      "attestation": "sha256:9f86d08..."
    }
  ],
  "revocations": ["lid://app.com/legacy#submit"],
  "signature": {
    "algorithm": "ecdsa-p256",
    "payload": "MEYCI...",
    "chain": ["x509:org-root", "x509:team-issued"]
  }
}
```

4.2 Required Fields

4.2.1 manifest

- **authority** — domain that owns this bundle (must match LID authority)
- **generated_at** — ISO 8601 UTC timestamp
- **ttl_days** — time-to-live in days (after which bundle is expired)
- **scope** — optional label for organizational grouping (e.g., "design-system")

4.2.2 contracts

Each contract MUST include:

- **lid** — immutable LID address
- **selector** — current valid CSS selector for the element
- **confidence** — float between 0.0 and 1.0
- **attestation** — optional sha256 hash of element content

4.3 Bundle Contents

- **Contracts:** LID → selector mappings with confidence scores
- **Manifest:** Bundle metadata, TTL, and scope information
- **Signatures:** Cryptographic verification chain
- **Revocations:** Retired LIDs with optional successors

> **Security:** All bundles must be cryptographically signed and verified before use in production environments.

5. Resolution Protocol RFC-003

The Resolution Protocol defines how applications translate LIDs into current CSS selectors using SCR bundles. It supports both networked and air-gapped (offline) resolution workflows.

5.1 Resolution Flow

1. Parse LID URI → Extract authority, path, fragment
2. Load SCR Bundle → Verify signature and TTL
3. Lookup Contract → Find matching LID entry
4. Return Result → Selector + confidence + metadata

5.2 Resolution Request

Endpoint Details:

- **Endpoint:** `/v1/resolve-many`
- **Method:** `POST`
- **Content-Type:** `application/json`

Request Schema

```
{
  "lids": [
    "lid://app.com/checkout#submit",
    "lid://auth.app.com/login#continue"
  ],
  "bundle_fingerprint": "sha256:d6c7a4...",
  "options": {
    "strict": true,
    "fallback": false
  }
}
```

5.3 Response Format

```
{
  "lid://app.com/checkout#submit": {
    "selector": "button.checkout-final",
    "confidence": 0.95,
    "ttl": 43200,
    "attestation": "sha256:6b86b273..."
  },
  "lid://auth.app.com/login#continue": {
    "selector": "form#login .btn-continue",
    "confidence": 1.0,
    "ttl": 86400
  }
}
```

5.4 Error Codes

Status Code	Meaning
`404`	LID not found in SCR bundle
`410`	LID explicitly retired
`423`	Bundle expired (TTL exceeded)
`498`	Invalid signature or fingerprint
`409`	Conflicting LIDs from multiple SCR

6. Best Current Practice BCP-LID-001

6.1 LID Naming Conventions

6.1.1 Authority Selection

✅ Do:

- Use domains you control long-term (`lid://company.com`)
- Prefer subdomains for teams (`lid://team.company.com`)

❌ Avoid:

- Ephemeral domains (`lid://staging-env-38.com`)
- IP addresses (`lid://192.168.1.1`)

6.1.2 Path Design

- Mirror URL structure: `lid://app.com/checkout/payment#submit`
- Use @ for owned namespaces: `lid://app.com/@design-system/button#primary`
- Avoid over-nesting (/page/section/component/subcomponent)
- Use versioned paths (/api/v2 vs. /api)

6.1.3 Fragment Semantics

- Be specific but concise (#submit-order, not #btn)
- Include element type (#search-input, #avatar-image)
- Avoid PII (#user-email)
- Avoid transient states (#expanded)

6.2 SCR Bundle Hygiene

6.2.1 Bundle Segmentation

- **Small Apps:** Single bundle
- **Large Apps:** Split by team/domain

6.2.2 TTL Policies

Bundle Type	Recommended TTL
Design Systems	365 days
Checkout Flows	90 days
A/B Test Variants	30 days

6.3 Security & Privacy

6.3.1 Attestation Practices

- **Critical Elements:** Always hash content
- **Non-Critical:** Skip hashing for performance

```
lid://bank.com/transfer#amount::sha256:6b86b2...
```

6.3.2 Key Rotation

1. Generate new keys annually
2. Publish old keys in `_linkism-revoked` DNS TXT

6.4 Examples

6.4.1 Good LIDs

```
lid://shop.com/checkout#apply-promo
lid://design.acme.com/@ds3/button#primary
lid://gov.uk/tax-form#error-ssn::sha256:...
```

6.4.2 LIDs to Avoid

```
lid://temp.com/modal#close          // Ephemeral authority
lid://app.com/#main-button           // No path context
lid://health.com/patients#diabetes   // PII in fragment
```

7. Reference Implementation

A complete reference implementation in Rust that demonstrates LID parsing, SCR bundle verification, and resolution. Designed for readability and specification compliance rather than production optimization.

7.1 Key Features

- **Zero Dependencies:** Pure functions with no external requirements
- **WASM Compatible:** Runs in browsers, servers, and embedded systems
- **Crypto-First:** Built-in signature verification and attestation
- **Test Coverage:** Comprehensive test suite for all edge cases

7.2 Core Structures

```
/// LID URI representation (RFC-001 §3)
#[derive(Debug, Clone, PartialEq, Eq, Serialize, Deserialize)]
pub struct Lid {
    authority: String,
    path: String,
    fragment: String,
    attestation: Option<String>,
}
```

7.3 Quick Start

```
// Parse and resolve a LID
let lid = Lid::parse("lid://app.com/auth#login")?;
let result = bundle.resolve(&lid)?;
println!("Selector: {}", result.selector);
```

> **Note:** This is a reference implementation. Production systems should use established cryptographic libraries for signature verification.

8. Quickstart Guide

8.1 Bundle Generation

```
# Generate bundle from existing selectors (migration mode)
linkism bundle \
  --lid lid://app.com/auth#login \
  --selector 'form#login > button.primary' \
  --key ./team-key.pem \
  --output auth.scr

# Batch generation from URL crawl
linkism bundle --url https://app.com --out app.scr
```

8.2 Resolution

```
# Resolve offline
linkism resolve \
  --lid lid://app.com/checkout#submit \
  --scr ./app.scr

# Validate bundle integrity
linkism verify-scr app.scr --trust-ring ./keys.pub
```

8.3 Integration Examples

8.3.1 Cypress

```
// cypress/support/commands.js
Cypress.Commands.add('getLID', (lid) => {
  return cy.window().then((win) => {
    return win.linkismResolve(lid).then(result => {
      return cy.get(result.selector, { timeout: result.ttl * 1000 });
    });
  });
});

// Usage in tests
cy.getLID('lid://app.com/auth#login').click();
```

8.3.2 Playwright

```
// playwright.config.js
export default {
  use: {
    // Auto-resolve LIDs to selectors
    lidResolver: 'https://api.linkism.org/v1'
  }
};

// Usage in tests
await page.lid('lid://app.com/checkout#submit').click();
```

9. References

9.1 Normative References

[RFC3986]	Berners-Lee, T., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
[RFC7519]	Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, May 2015.
[RFC5280]	Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

9.2 Informational References

[BCP-LID-001]	Trout, J., "Deployment Guidelines for LID Infrastructure", BCP-LID-001, August 2025.
[LINKISM-IMPL]	Linkism Protocol Foundation, "Reference Implementation in Rust", August 2025.

Open specification, royalty-free implementation

"A web where elements outlive frameworks"