Decentralized Identifier (DID) Specification - Bidkee Enhanced Proposal

Draft Version: 1.6

Date: April 08, 2025

Author: [Jun Chen, Individual Contributor]

Submitted to: W3C Decentralized Identifier Working Group

Abstract

This proposal enhances the W3C DID specification (v1.0) through the Bidkee framework, offering blockchain-agnostic dual-signature verification and an extensible identity structure for IoT endpoints (e.g., drones), payments, supply chain tracking, and identity cards. It introduces equipmentID and dynamicData fields, ensures FAA Remote ID compatibility, and invites community validation.

Catalogue

1. Introduction

Decentralized Identifiers (DIDs) provide a foundation for self-sovereign identity, yet traditional DIDs fall short in fully autonomous endpoint and identity certification scenarios requiring regulatory compliance. Bidkee introduces dual signatures—combining authorization and responsibility—supporting platforms like Kaspa or Polygon. Version 1.6 adds an identity card use case, refines the design, focuses on DID method essentials, and envisions future extensions.

---

2. Terminology

- DID: Decentralized Identifier, a unique identifier (did:method:specific-id).
- DID Document: A JSON-LD object containing public keys and service endpoints.
- First Blockchain Address: The holder's blockchain address, controlling the identity and generating signatureMessage, acting as the "structure owner."
- Superordinate Blockchain Address: The issuer's blockchain address, generating superordinateSignature for authorization, distinct from controller (which governs the DID), emphasizing regulatory compliance.
- Identity Structure: A data object with static and dynamic fields.
- Signature Message: An encrypted signature based on specific data.

Abbreviations

Term	Meaning
FAA	Federal Aviation Administration
ECDH	Elliptic Curve Diffie-Hellman
ASTM	American Society for Testing and Materials
JCS	JSON Canonicalization Scheme

---

3. DID Syntax

3.1 Core Syntax

did:bidkee:[blockchain-prefix]:[specific-identifier]

Examples:

- did:bidkee:kaspa:qqc3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf85g4q
- did:bidkee:polygon:0x1234567890abcdef1234567890abcdef12345678

3.2 Enhanced Features

- Dual Signatures: Separates issuer authorization and holder responsibility.
- Blockchain Agnosticism: Supports any blockchain with unique addresses.

---

4. DID Document Structure

4.1 Base Structure

json

```json
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:bidkee:kaspa:qqc3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf85g4q",
  "controller": "did:bidkee:kaspa:qqz3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf85g4q",
  "authentication": [{
    "id": "did:bidkee:kaspa:qqc3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf85g4q#keys-1",
```

```json
    "type": "Ed25519VerificationKey2020",
    "publicKeyMultibase": "z6Mkf...xyz"
  }]
}
```

4.2 Bidkee-Specific Extensions

- Identity Structure (idStructure):
  - Static Fields:
    - firstBlockchainAddress: Holder’s address (required), the structure owner.
    - superordinateSignature: Issuer’s authorization signature (required) based on firstBlockchainAddress and equipmentID, using JCS canonicalization (RFC 8785), SHA-256 hashing, and Ed25519 signing.
    - superordinateBlockchainAddress: Issuer’s address (optional), resolved if absent.
    - checkCode: SHA-256 hash of static fields (firstBlockchainAddress, equipmentID).
  - Dynamic Fields:
    - optionalFields.dynamicData: Device or identity state data (e.g., location, name), with sequenceNumber for versioning, supporting external broadcast protocols.

json

```json
{

  "firstBlockchainAddress":
"kaspa:qqc3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf
85g4q",

  "superordinateSignature": "sig-super-20250407",

  "superordinateBlockchainAddress":
"kaspa:qqz3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf
85g4q" /* optional */,

  "checkCode": "sha256-abc123",
```

```json
  "optionalFields": {
    "equipmentID": "ID-20250407-CN12345678",
    "permissionData": "citizen-20250407",
    "dynamicData": {
      "name": "Zhang San",
      "birthdate": "1990-01-01",
      "timestamp": "20250407T12:00:00Z",
      "sequenceNumber": 1
    }
  }
}
```

- Signature Message (signatureMessage):

json

```json
"signatureMessage": "sig-device-20250407"
```

  - Generated by firstBlockchainAddress based on checkCode, proving owner responsibility.

---

5. DID Method Operations

5.1 Create

1. User generates firstBlockchainAddress.
2. Issuer assigns equipmentID (e.g., identity card number).
3. Issuer generates superordinateSignature based on JCS-normalized firstBlockchainAddress and equipmentID.
4. Computes checkCode, stored on-device or on-chain.

5.2 Resolve

1. Retrieve idStructure and signatureMessage.
2. If superordinateBlockchainAddress is absent, use a fallback resolver (e.g., chain query).
3. Verify signatures.
4. Return DID document.

5.3 Update

- Modify dynamicData, increment sequenceNumber, regenerate checkCode and signatureMessage, asynchronously commit to blockchain.

5.4 Deactivate

- Issuer signs a revocation message, recorded on-chain.

---

6. Verification and Operations

6.1 Dual Signature Verification

- Issuer: Verifies superordinateSignature using superordinateBlockchainAddress public key, on-demand (local: 1ms, Kaspa chain: 3-5s).
- Holder: Verifies signatureMessage based on checkCode, on-demand.
- Note: Dual signatures separate authorization (superordinateSignature) and responsibility (signatureMessage). In identity card scenarios, this ensures “legitimate issuance + holder possession.”

6.2 Operations

- Retrieve equipmentID or display dynamicData.
- Validate permissions or identity.

---

7. Privacy and Security

- Privacy: dynamicData supports AES encryption, with keys negotiated via ECDH.
- Security: Dual signatures and checkCode ensure integrity.
- Performance: Verification latency varies by chain (e.g., Kaspa 3-5s, Polygon <1s), triggered on-demand.

---

8. Use Cases

8.1 Drone Identification

- DID: did:bidkee:kaspa:qqc3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf85g4q
- Operation: Regulator authorizes (superordinateSignature), user manages (signatureMessage), dynamicData supports FAA broadcast compatibility.

8.2 Bank Card

- DID: did:bidkee:polygon:0x1234567890abcdef1234567890abcdef12345678
- Operation: Bank authorizes, cardholder manages, processes transactions.

8.3 Supply Chain Tracking

- DID: did:bidkee:polygon:0x1234567890abcdef1234567890abcdef12345678
- Operation: Manufacturer authorizes (superordinateSignature), logistics manages (signatureMessage), tracks goods status.

8.4 Identity Card

- DID: did:bidkee:kaspa:qqc3a2j95vhn9jlq9d87mexyg7dwc0lvnyvzwypgwk9hx00h44krvlhf85g4q
- Operation:
  - Government authorizes (superordinateSignature), proving legitimate issuance.
  - Holder (firstBlockchainAddress) generates signatureMessage, proving possession.
  - Example: Logging into government services verifies both signatures for “legitimate + holder.”
- Benefit: Traditional DIDs lack dynamic holder proof; Bidkee’s dual signatures address this gap.

---

9. Community Collaboration

- Validation: Test signature efficiency and use case scalability (e.g., identity cards).
- Contribution: Share via the DID Community Group.

---

10. Intellectual Property

- Statement: Submitted under W3C CLA, granting a royalty-free, non-exclusive license. Involves granted patent US12124600B1 (grant date available upon request) and its patent family member CN202311458093.8 (pending, not yet granted), with identical specifications. Details or claim summaries are available upon community request, or a formal patent statement will be provided upon CN grant.

---

11. Differences from W3C DID Core Specification

Feature	W3C DID Core	Bidkee Enhancement
Signature	Single entity control	Dual signatures: authorization (superordinateSignature) + responsibility (signatureMessage), enabling holder proof
Data Fields	Standard JSON-LD	Adds equipmentID, dynamicData, checkCode for endpoints and identity
Autonomy	Fully self-sovereign	Balances authorization and autonomy for regulated/identity scenarios
Blockchain Support	Optional binding	Requires chain prefix, multi-chain compatibility

---

12. Conclusion

Bidkee enhances the DID ecosystem with dual signatures, separating authorization and responsibility, addressing needs in smart endpoints (e.g., drones) and identity certification (e.g., identity cards). In identity scenarios, dual signatures prove "legitimate issuance + holder possession," filling a gap in traditional DIDs while maintaining a lightweight design. We look forward to community refinement.

Appendix: Implementation Notes

- Blockchain Flexibility: Supports public/private chains.
- Reference Code: Draft at https://github.com/bidkee/did-method-bidkee (TBD).
  - Planned content:
    - checkCode and signature generation scripts (Python/JS).
    - Dynamic field handling examples.
- Signature Verification Performance:
  - Local: Ed25519 signing 0.5ms, verification 1ms.
  - Chain query: Kaspa 3-5s, Polygon <1s, on-demand.
- dynamicData Broadcast Compatibility:
  - Current: Generic state container, supports external protocols (e.g., BLE/Wi-Fi, ASTM F3411-22a, ~50 bytes, 1 Hz).

- Future: Adaptable to V2X (e.g., C-V2X, 5G NR) for low-latency, high-range communication.
- Note: This specification does not define broadcast, only provides data structure support.

- Cross-Specification Mapping:

Bidkee Field	FAA/ASTM F3411-22a	W3C DID
firstBlockchainAddress	-	DID Subject
equipmentID	Serial Number	-
superordinateSignature	-	Verification Method
dynamicData.latitude	Latitude	-