Interchain Identifiers (IIDs) *DRAFT*

Abstract

Interchain Identifiers (IIDs) are a family of DID methods for blockchain-based assets such as tokens, smart contracts, and addresses. Built on Decentralized Identifiers from the World Wide Web Consortium [1], IIDs are 100% conformant DIDs; IIDs ***are*** DIDs and IID documents are DID documents. The IID specification also provides additional features designed for on-chain assets, such as the ability to verifiably yet privately associate arbitrary digital or real-world assets with the underlying IID Subject. IIDs also constrain compatible DID methods in particular ways, including a restriction on the DID Subject—IID Subjects MUST be on-chain resources. Any DID method that meets the requirements in this specification ***is*** an IID method, and its IIDs ***are*** DIDs. Any DID software that is conformant to the W3C specification SHOULD be able to work with IIDs and IID documents, although some IID-specific features may require additional support.

Introduction

Identifiers are needed to refer to both on-chain and off-chain resources. Non-fungible tokens (NFTs) require a cryptographically secure means to both refer to specific tokens and prove control or ownership without reliance on a trusted third party.

We refer to this new class of identifiers as Interchain Identifiers, or IIDs for short. We propose this new identifier would be appropriate for any on-chain asset, including smart contracts, fungible tokens, and non-fungible tokens.

IIDs build on a foundation of global identifiers dating back to Uniform Resource Locators and Uniform Resource Identifiers [2], which are the foundation of the World Wide Web. URLs and URIs allow for different schemes that specify particular mechanisms for interpreting and applying the rest of the identifier. For example, http://example.com specifies a web-based resource that can be retrieved using the hypertext transfer protocol (http) and mailto:joe@example.com specifies a resource that can receive email messages using the SMTP protocol.

IIDs are Decentralized Identifiers (DIDs), which build on URIs and URLs to provide a class of identifiers one can verify without reliance on a trusted third party. DIDs specifically support cryptographic verification methods like public/private key cryptography on any number of curves, which in turn are used for verification relationships like authentication, assertion, capability delegation, and capability invocation. When one uses a DID for those relationships, a verifying party can use cryptographic material associated with the DID to verify the use is appropriate. Extending DIDs into DID-URLs, IIDs use the path and fragment parts of URLs to refer to IID References and IID Resources. We describe how IIDs and DIDs relate in section XXX.

IIDs also provide a consistent, verifiable approach for referring to off-chain resources, whether digital, real-world, or conceptual. In this role, IIDs rely on content-based identifiers and content-based addressing systems, where a hash of the digital asset is used to unambiguously verify the asset is the intended resource. Standards like multihash [3] provide for a wide range of different hash algorithms and representations. Approaches using merkle trees can turn hashes into a hashgraphs, allowing an arbitrary

number of resources to be independently verified without revealing even the number of those assets publicly. These content identifiers can provide robust verifiability no matter where such resources might be located or retrieved.

Specifically for NFTs, IIDs allow you to

1. Identify individual NFTs as well as their token class
2. Identify resources associated with that NFT
3. Exercise rights related to those resources using cryptography anchored to the NFT

IIDs allow these functions a manner that addresses the privacy requirements for such references, with flexible support for use cases that need to make different choices. Our approach offers a robust, flexible, and capable identifier mechanism for NFTs, and on-chain assets, of all types.

Use Cases

This specification was designed for Non-Fungible Tokens. NFT-RFC-002 [4] describes the range of types of tokens as well as several detailed user stories. NFT-RFC-008[5] describes how one might apply IIDs, Verifiable Credentials, and Authorization Capabilities to define, authorize, issue, use, and sell NFT-based Renewable Energy Certificates (RECs).

Throughout this document we use the REC use case for examples. See RFC for details, but there short description is xyz.

Requirements for the IID specification

This specification MUST achieve the following.

R1. IIDs MUST be conformant with the DID Core syntax for DIDs
R2. IIDs MUST be able to refer to any asset on any chain, on any network, on any fork.
R3. IID documents MUST be conformant DID documents in JSON-LD representation
R4. IID documents MUST be able to use custom properties in an unambiguous manner.
R5. IID documents MUST be able to define their own namespace for referring to off-chain assets
R6. IID documents MUST be able to specify how to verify, and optionally retrieve, associated resources, revealing neither the number nor character of those resources.
R7. IIDs must be usable in an interoperable fashion with emerging self-sovereign approaches such as Verifiable Credentials, Authorization Capabilities, and Confidential Storage.
R8. IIDs must be recognizable as IIDs by simple examination of the IID itself.
R9. IIDs must be usable with existing tooling and infrastructure.

Requirements for IIDs, IID Documents, and IID Methods

The following are requirements for IIDs above and beyond conformance requirements of DID Core.

R10. All IIDs MUST refer to a single on-chain asset, such as a token, smart contract, or address.
R11. IID documents MUST be conformant DID documents in JSON-LD representation.
R1.11 All properties used in an IID document must be conformant with DID JSON-LD processing rules, including the definition of such properties in JSON-LD context files specified by the @context property.

R12. IID Methods SHOULD support off-chain creation of identifiers with on-chain updates
R13. IID Methods SHOULD define one and only one service endpoint
R14. IID Methods SHOULD define one and only one Linked Resource

Extensibility

IIDs use two key mechanisms for extensibility, following the DID Extensibility approach. [link to section]

First, IID methods may be created for any existing or new blockchain or distributed system. Each IID method specifies the mechanisms that allow a DID user to create, read, update, and deactivate IIDs of that method. There is no registration required and literally anyone can define an IID method.

Second, JSON-LD enables unambiguous use of JSON properties through the addition or modification of @context properties. This approach allows any RDF statement to be losslessly represented in JSON, avoiding possible confusion between properties with the same name being used differently by different organizations. Thanks to DID Core, IID documents already have a @context property with https://www.w3.org/ns/did/v1 as its initial value. This specification adds a second @context property, (its value TBD, but something like https://internft.org/ns/iid/v1). This context value pulls in a JSON-LD context file, stored at that location, which contains the definitions specific to IIDs.

Like any DID method, any IID Method may add properties in a decentralized way by creating their own JSON-LD context file, publishing that online, and adding it as a third @context, e.g., the context property for a Cosmos customization of IIDs might be

```
"@context" : ["https://www.w3.org/ns/did/v1",
   "https://internft.org/ns/iid/v1",
   "https://cosmos.network/ns/cosmosIid/v1" ]
```

This context value pulls in three JSON-LD contexts, applying them in order, to unambiguously define all of the properties used in the DID document.

This allows unambigious extensibility without a central registry of vocabulary.

New Terminology

T1. **IID References** are DID-URLs with a fragment part, e.g., **did:example:abc#123**. They MUST only be used to refer to resources associated with the IID subject in some way.
T2. **IID Resources** are DID-URLs with a path part, e.g., **did:example:abc/123**. They MUST only be used to for retrievable (online) resources that are associated with the IID subject in some way.

NOTE: Add something about http range 14

New Properties and Values

P1. **LinkedResources** is a new IID document property for specifying how to verify, and optionally retrieve, any and all resources necessary for the proper function of the on-chain asset. LIKE MIME ATTACHMENTS…
P2. **Extension (a type of Linked Resource)**: A JSON-LD extension of the current document. The RDF statements in the extension are to be interpreted as if they were included in the current IID

document. For example, you might provide additional service endpoint definitions in an linked resource. Those endpoints can be verified as associated with that IID, but only by those parties who secure those definitions through other, privacy respecting means.

P3. **Accorded Rights (a type of Linked Resource)**: Similarly, linked resources could specify real-world rights accorded to the IID owner or its designate, such as a digital driver's license or a theater ticket. The representation framework for such rights must be open ended, including both plain text statements of rights, e.g., "The controller of this IID is entitled to ...", or more rigorous, and computationally evaluatable RDF statements which might describe in great detail a range of benefits that accompany the basic rights of the token.

P4. **Executable Rights (a type of Linked Resource)**: The resources could also define executable capabilities that can be invoked by the IID owner or its designate, using cryptographic materials defined elsewhere in the document. Executable Rights are a type of Accorded Rights.

P5. **Assertions (a type of Linked Resource)**: Verifiable credentials, verified claims, claim tokens as described in NFT-RFC-008. This allows arbitrary, yet verifiable attestations to be made either about the asset or about the resources defined by IID references. The attributes represented in these claims can be retrieved via the NFT interface using a query by example (graph query) mechanism.

New Service Types

POLYMORPHIC MEDIATOR

IID Syntax

(Same as DID Syntax Section 3.1)

```
iid                = did

did                = "did:" method-name ":" method-specific-id

method-name        = 1*method-char

method-char        = %x61-7A / DIGIT

method-specific-id = *( *idchar ":" ) 1*idchar

idchar             = ALPHA / DIGIT / "." / "-" / "_"
```

For example, IID References and IID Resources are DID-URLs such as IID References—URIs for making statements about resources—and IID Resources—URLs [NEED DID-URL ABNF]

The IID Architecture

IIDs as DIDs

Decentralized Identifiers (DIDs) are a standards-track specification under development at the World Wide Web Consortium. https://www.w3.org/TR/did-core/ They are "a new type of globally unique

identifier designed to enable individuals and organizations to generate our own identifiers using systems we trust, and to prove control of those identifiers (authenticate) using cryptographic proofs (for example, digital signatures)."

DIDs enable verifiable, decentralized digital identity. They were inspired by blockchain based approaches for "identity", using an RFC3986 compatible syntax. As such DIDs *are* URIs and are designed to work within existing web frameworks.

DID-URLs support additional **path** (/directory/file), **query** (?query=value), and **fragment** (#fragment)parts. Fragments are commonly used to refer to elements within a DID document.

DIDs work because of two structural design choices.

First, each DID specifies a DID Method, which defines how a DID consumer might perform Create, Read, Update, and Deactivate operations. Different types of DIDs may have different Methods, and each Method may define unique approaches for those functions. This allows DIDs that anchor to Bitcoin and Ethereum, as well as bespoke fit-for-purpose ledgers such as Veres One or Sovrin. DID Methods *may* be registered in a common registry for increased interoperability, but such registration is not required; it is merely a convenience.

Second, every DID resolves to a DID document, which contains all the metadata required for interacting securely with the DID Subject.

- Public Keys
- Verification Relationships
    - Assertion Method
    - Authentication Method
- Service endpoints

DID document state is managed by DID Registries; many are blockchains, but alternatives such as did:peer, did:git, and did:key do not rely on blockchains at all. Methods like did:key and did:btcr provide for deterministic creation of DID documents based on rules in their specification; no actual DID document is stored anywhere. A more recent class of DIDs, including did:v1, did:ethr, and did:ion allow the creation of DIDs without registration, relying on the registries purely for updates; did resolution for these methods checks the registry for any updates and if there are none, use a deterministic algorithm to generate a DID document or a cryptographic verification of an initial DID document communicated through alternate channels.

It is this indirection of DID documents that allows key rotation without explicit notification to all parties. It is the open-ended DID Method architecture that allows for future proofed innovation.

Third, DID Methods with publicly inspectable registries (where DID document state is maintained), such as blockchains, may also provide for auditability, and with some effort, even reversibility of changes in case of inappropriate transactions. These advanced features remain areas of active development.

Fourth, DID documents are defined using an abstract data model, allowing any number of serializations, with initial support for JSON and JSON-LD serializations. CBOR and CBOR-LD are also highly anticipated for their ability to dramatically reduce the size of DID documents. Other serializations are also possible, including Protobuf.

Finally, although it is sometimes useful to think of DIDs are an architecture for decentralized identity, the functional mechanism for realizing is a framework for cryptographic verification for identifiers. XXX DIDs link an identifier to cryptographic verification methods for different verification relationships.

This specification leverages DIDs for all of those benefits by defining IIDs as profile of DIDs. IIDs *are* DIDs. IID documents *are* DID documents. IID Methods *are* DID Methods. Each namespace for IIDs, e.g., Cosmos, Polkadot, ERC1155, is defined by an IID method that specifies how to perform create, read, update, and deactivate operations on IIDs. This IID family of DID Methods share a common purpose and leverage shared property definitions for improved cross-chain interoperability. By design, an IID from any chain can be read, used, and operated on from any chain thanks to IID-specific properties and common DID properties. IIDs may also be used by traditional DID software for standard DID functionality such as authentication and verification.

Because DIDs are URIs, IIDs are also URIs, making IIDs natively compliant with existing RFC3986 compliant libraries, including those used for RDF and the World Wide Web.

Consider the following example minimum DID document from DID Core:

EXAMPLE 1: A minimal DID document (JSON-LD)

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:abc123",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:abc123#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:abc123",
    "publicKeyMultibase":
"zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

This minimal DID document is fully conformant with the specification and includes a single verification relationship and method: how to authenticate on behalf of the DID Subject.

It is useful to not that Verification Methods can be anything*, e.g., ed25519, secp256k, etc.

An equivalent minimal IID document would be

EXAMPLE 2: A minimal IID document

```
{
  "@context": ["https://www.w3.org/ns/did/v1",
               "https://internft.org/ns/iid/v1"],
  "id": "did:example:abc123",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:abc123i#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:abc123",
    "publicKeyMultibase":
"zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }]
}
```

Like Example 1, this minimal IID document simply provides for cryptographic authentication on behalf of the IID; for some purposes this is sufficient. The second context entry pulls in the IID JSON-LD context that indicates this DID document is an IID document. Literally, it means that the property definitions from the IID context—which are defined in this specification—apply to the properties of this DID document, which makes it an IID document.

IID subjects are always on-chain asset like Non-Fungible Tokens. Authenticating on a token's behalf is taken to indicate proof-of-ownership. Since most on-chain assets already employ some form of cryptographic proof-of-control for exercising ownership rights, an IID document of this kind should be readily producible for just about any asset on any blockchain (or at least those assets which have a cryptographic proof-of-control). It may be necessary to provide a new type of verification method if the underlying chain is using an innovative cryptographic proof, but DID documents are extensible for precisely this flexibility.

In this example, the CRUD operations of the DID are managed by method-specific means based on Ed25519. The on-chain state—referred to as the "registry" in DID Core—can be inspected to deterministically generate this IID document. Different registries provide different ways to inspect that state and different rules for deterministic generation. This variability is handled in the individual IID methods based on particular registries: each method describes how to work with its registry. In many cases, there is no need to store a fully formed DID document—which makes DIDs particularly well suited to existing blockchain infrastructure: in many cases, no changes to the underlying chain mechanics are necessary.

EXAMPLE 3: A minimal, privacy-preserving IID document

```
{
```

```
  "@context": ["https://www.w3.org/ns/did/v1",
               "https://internft.org/ns/iid/v1"],
  "id": "did:example:abc123",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:abc123i#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:abc123",
    "publicKeyMultibase":
"zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }],
  "service": [{
    "id":"did:example:abc123#mediator",
    "type": "polymorphicMediator2021",
    "serviceEndpoint": "http://8zd335ae47dp89pd.onion/iid/mediator/
did:example:abc123"
    }],
   "linkedResource" : [{
     “id”: “did:ixo:abc123#resourceHashgraph”,
     “path”: “did:ixo:abc123/resourceHashgraph”,
     “type”: “hashgraph”,
     “proof”: “afybeiemxf5abjwjbikoz4mcb3a3dla6ual3jsgpdr4cjr3oz”,
     “endpoint” : "did:example:abc123?service=mediator"
   }]
}
```

There are four privacy-preserving elements in this privacy-preserving minimal IID document.

NOTE: It would be better to make these properties mandatory for IIDs to maximize herd privacy. However, the current language of "SHOULD" for these requirements allows IID creators to violate herd privacy should they deem it appropriate for their use case. It is discouraged, but allowed. This deserves further discussion.

First, there is one and only one service endpoint, which points to a shared, polymorphic mediator, as described in Section 10.6 Service Privacy from the DID Core Specification https://w3c.github.io/did-core/#service-privacy

Service endpoints are optional, but to support herd privacy, it is a recommended practice to use one, and only one service endpoint, which is a SHOULD requirement for IIDs. For maximum privacy, this specification further recommends that the service endpoint specifies a Tor address for a polymorphic service capable for providing a number of functions such as a negotiator, mediator, or confidential storage.

> *It is the goal of herd privacy to ensure that the nature of specific subjects is obscured by the population of the whole. To maximize herd privacy, implementers need to rely on one — and only one — service endpoint, with that endpoint providing a proxy or mediator service that the controller is willing to depend on to protect such associations and that blinds requests to the ultimate service.*

This service endpoint—using an API which is still under development—is capable of acting as any number of services, all in a trusted, privacy preserving manner.

Second, that service endpoint is a Tor endpoint, which provides enhanced network security, mitigating the ability for network observers to detect attempts to access that endpoint. The underlying service is just like any other web service; using Tor obfuscates the networking layer without changing the service or API that runs over that secure connection. ADD THIS ALSO GIVES GEOGRAPHIC shielding

Third, there is one, and only one, linked resource entry which is capable of verifying an unknown number of associated resources. The entry type of "hashgraph" means that the proof is a Merkle tree of hashes of the underlying resources. Anyone who gathers the appropriate resources can independently verify their completeness and authenticity by comparing the hashgraph of the combined resources with the proof property in the service endpoint definition. Anyone considering buying an NFT can be cryptographically assured they have all the resources they need to confirm the NFT is what they believe it to be.

For herd privacy, it is preferred that all IIDs use one, and only one, service endpoint and one, and only one, linked resource. In cases where there is no appropriate service endpoint, it is still preferable to point to a mediator; to support that, we encourage a multitude of publicly accessible mediator services offering a free "null response" service where such "null response" is indistinguishable from a failed access from lack of authorization.

NOTE: It's ok to have multiple resources listed, but it is less preferred.

Fourth, the single linked resource property lists its endpoint as the IID's single service endpoint, using a properly formed DID-URL with a service parameter. This pattern allows every resource to use the same endpoint without duplicating endpoint URLs even when specifying multiple linked resources. This pattern encourages polymorphic mediators, which improves overall privacy across the IID ecosystem.

Lifecycle of an NFT

1. Define Class (delegation set up)
2. Delegate (optional) Capability to Mint

3. Mint Invoke w/ caveats
4. Use
    a. Inspect it
    b. Invoke Services
        i. Request resources
        ii. Communicate with owner
    c. Exercise Accorded Rights (with or without additional requirements)
        i. Bond
        ii. Unbond
        iii. Vote
        iv. See a show
        v. Decrypt Media
    d. Delegate Accorded Rights
    e. Create derivatives
    f. Collateralize it
    g. Use on another chain
5. Offer for sale
6. Verify Linked Resources
7. Buy
8. Burn

Verifiable Credentials

Executable Rights

Delegation, Authorization, and Caveats

Authorization Capabilities

Secure ECMAScript

Confidential Storage

IPFS?

Polymorphic Mediator-Negotiator Endpoints

IBC

Interchain Accounts?

Wallets

Privacy Considerations

How to create an IID Method

Terminology

T1. DIDs

T2. DID-URLs
T3. DID documents
T4. Resolution
T5. Dereference

References

[1] Decentralized Identifiers (DIDs) v1.0. World Wide Web Consortium. Online at https://www.w3.org/TR/did-core/. Accessed February 15, 2021.

[2] URIs RFC 3986

[3] Multihash

[4] NFT-RFC-002

[5] NFT-RFC-008

NOTES

(not for RFC publication)

Future Work

Addressing how to deal with cross-chain transfers of IIDs. New chain means a new identifier, but the link to the original NFT is preserved, on both chains.