L2 Token List Version 1.0

L2 Token List Version 1.0

Project Specification Draft

Open Project:

Layer 2 Working Group, an initiative of Oasis Open Projects

Project Chair:

Dan Shaw (daniel.shaw@ethereum.org), Ethereum Foundation, Andreas Freund (a.freundhaskel@gmail.com), Ethereum Foundation

Editors:

Kelvin Fichter (kelvin@optimism.io)

Andreas Freund (a.freundhaskel@gmail.com)

Pavel Sinelnikov (pavel.sinelnikov@outlook.com)

Related work:

NA

Abstract:

The document describes the minimal set of business and technical prerequisites, functional and non-functional requirements for a token list that when implemented ensures that two or more Layer 1, Layer 2, or Sidechains can identify tokens from different Layer 1, Layer 2, or Sidechains.

Status:

This document is no longer under active development, and is a Project Specification Draft as of February 2023. The L2 WG is looking for implementers of the specification to move the specification to Full Project Specification.

Comments on this work can be provided by opening issues in the project repository.

Keywords:

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119] when, and only when, they appear in all capitals, as shown here.

Citation format:

When referencing this specification the following citation format should be used:

[l2-token-list-v1.0] *Token List Specification Version 1.0*. Edited by Kelvin Fichter, Andreas Freund, Pavel Sinelnikov. 08 February 2023. OASIS Standard. https://github.com/eea-oasis/L2/tree/main/workitems/tokenlist/l2-token-list-v1.0-psd01.md. Latest stage: https://github.com/eea-oasis/L2/tree/main/workitems/tokenlist/l2-token-list-v1.0-psd01.md.

Notices

Table of Contents

1 Introduction

The L2 WG is an open-source initiative with a scope to

- Identify and document the most relevant use cases and business requirements for Layer 2 and other Blockchain Scalability solutions for EVM compatible public blockchains

- Define a technical standard with identification and differentiation of classes of scalability solutions as required that meet both ecosystem and enterprise requirements, with a particular focus on interoperability between Layer 2 solutions for EVM compatible public blockchains
- For EVM compatible public blockchains, identify, document, and devise solution approaches for Layer 2 Blockchain scalability solution specific challenges such as MEV, block (gas) limits, TVL concentration, etc.
- Identify and document characteristics of Layer 2 Blockchain environments for EVM compatible public blockchains that will be key in addressing mainstream and enterprise adoption.

The work is an Ethereum Community Project, which is managed by OASIS.

1.1 Overview

There is a significant challenge around the definition and listing of tokens on Layer 1 (L1), Layer 2 (L2), and Sidechain systems. Note that for simplicity, this document we will collectively refer to L1, L2 and Sidechain systems as chains below since the challenge described below is valid across all such systems:

- Consensus on the “canonical” token on chain B that corresponds to some token on chain A. When one wants to bridge token X from chain A to chain B, one must create some new representation of the token on chain B. It is worth noting that this problem is not limited to L2s – every chain connected via bridges must deal with the same issue.

Related to the above challenge is the standardization around lists of bridges and their routes across different chains. This will be addressed in a separate document.

Note that both of these issues are fundamental problems for the current multi-chain world.

Therefore, the goal of this document is to help token users to operationalize and disambiguate the usage of a token in their systems.

Also note that a standard for defining tokens is beyond the scope of this document

1.2 Glossary

Blockchain:

An open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way.

Bridge:

Provides a connection that allows for the transfer of digital tokens or data between two different Layer 1, Layer 2 or Sidechain systems.

Layer 1:

A base blockchain network, such as Bitcoin, or Ethereum, and its underlying infrastructure that validates and finalizes transactions without the need of another blockchain network.

Layer 2:

A secondary framework or protocol that is built on top of an existing Layer 1 system in such a way that it inherits the security properties of the Layer 1 system while allowing for a higher transaction throughput that the Layer 1 system.

Sidechain:

A secondary blockchain connected to the main blockchain with a two-way peg and using its own trust assumptions.

Two-Way Peg:

A mechanism by which tokens are transferred between a blockchain and a sidechain and back at a fixed or otherwise deterministic exchange rate.

1.3 Typographical Conventions

1.3.1 Requirement Ids

A requirement is uniquely identified by a unique ID composed of its requirement level followed by a requirement number, as per convention **[RequirementLevelRequirementNumber]**.
There are four requirement levels that are coded in requirement ids as per below convention:

[R] - The requirement level for requirements which IDs start with the letter*R* is to be interpreted as **MUST** as described in RFC2119.

[D] - The requirement level for requirements which IDs start with the letter*D* is to be interpreted as **SHOULD** as described in RFC2119.

[O] - The requirement level for requirements which IDs start with the letter*O* is to be interpreted as **MAY** as described in RFC2119.

Note that requirements are uniquely numbered in ascending order within each requirement level.

Example : It should be read that [R1] is an absolute requirement of the specification whereas [D1] is a recommendation and [O1] is truly optional.

---

2 Concepts and Design

For lists of canonical tokens, L2s currently maintain their own customized versions of the Uniswap token list. For example, Arbitrum maintains a token list with various custom extensions. Optimism also maintains a custom token list, but with different extensions. It should be noted that both of these custom extensions refer to the bridge that these tokens can be carried through. However, these are not the only bridges that the tokens can be carried through, which means that bridges and token lists should be separated. Also note that currently, both Optimism and Arbitrum base “canonicity” on the token name + symbol pair.

An example of an Arbitrum token entry is given below:

```
{
logoURI: "https://assets.coingecko.com/coins/images/13469/thumb/1inch-token.png?1608803028",
chainId: 42161,
address: "0x6314C31A7a1652cE482cffe247E9CB7c3f4BB9aF",
name: "1INCH Token",
symbol: "1INCH",
decimals: 18,
extensions: {
  bridgeInfo: {
    1: {
    tokenAddress: "0x111111111117dc0aa78b770fa6a738034120c302",
    originBridgeAddress: "0x09e9222e96e7b4ae2a407b98d48e330053351eee",
    destBridgeAddress: "0xa3A7B6F88361F48403514059F1F16C8E78d60EeC"
     }
   }
  }
}
```

This standard will build upon the current framework and augment it with concepts from the W3C DID Specification [1] based on the JSON linked data model [2] such as resolvable unique resource identifiers (URIs) and JSON-LD schemas which enable easier schema verification using existing tools.

---

3 Token List Specification

The schema for a canonical token list utilizes draft version 7 of https://json-schema.org for consistency purposes with

the W3C CCG effort.

[R1]

The following data elements MUST be present in a canonical token list:

- type
- tokenListId
- name
- createdAt
- updatedAt
- versions
- tokens

Note, that the detailed definition of the data elements in[R1] along with descriptions and examples are given in the schema itself below.

[R1] testability: See suggested test fixtures for the data schema below.

[R2]

The tokens data element is a composite which MUST minimally contain the following data elements:

- chainId
- chainURI
- tokenId
- tokenType
- address
- name
- symbol
- decimals
- createdAt
- updatedAt

Note, that the detailed definition of the data elements in[R2] along with descriptions and examples are given in the schema itself below.

[R2] testability: See suggested test fixtures for this documents' data schema below.

[D1]

All other data elements of the schema SHOULD be included in a representation of a canonical token list.

[D1] testability: See suggested test fixtures for this documents' data schema below.

[CR1]>[D1]

If the extension data elements is used, the following data elements MUST be present in the schema representation:

- rootChainId
- rootChainURI
- rootAddress

Note, that the detailed definition of the data elements in[D1] and [CR1]>[D1] along with descriptions and examples are given in the schema itself below.

[CR1]>[D1] testability: See suggested test fixtures for this documents' data schema below.

[R3]

All properties in the schema identified in the description to be a Universal Resource Identifier (URI) MUST follow in

their semantics RFC 3986.

[R3] testability: All requirements for RFC 3986 are testable.

[R4]

The chainId property utilized MUST allow for the requirements of the EIP-155 standard to be met.

Namely, transaction replay protection on the network that is identified by the chainId property value. Note, that for replay protection to be guaranteed, the chainId should be unique. Ensuring a unique chainId is beyond the scope of this document.

[R4] testability: EIP-155 requires that a transaction hash is derived from the keccak256 hash of the following nine RLP encoded elements (nonce, gasprice, startgas, to, value, data, chainid, 0, 0) which can be tested easily with existing cryptographic libraries. EIP-155 further requires that the v value of the secp256k1 signature must be set to {0,1} + CHAIN_ID * 2 + 35 where {0,1} is the parity of the y value of the curve point for which the signature r-value is the x-value in the secp256k1 signing process. This requirement is testable with available open-source secp256k1 digital signature suites. Therefore, [R4] is testable.

[D2]

The chainId property SHOULD follow EIP-3220 draft standard.

[D2] testability: The EIP-3220 draft standard can be tested because the crosschain id is specified as a concatenation of well-defined strings, and using open source tooling can be used to parse and split a crosschain id, the obtained string segments can be compared against expected string lengths, and context dependent, the values for the strings specified in the standard. Consequently, [D2] is testable.

[O1]

The humanReadableTokenSymbol property MAY be used.

[O1] testability: A data property is always implementable in a schema.

[CR2]>[O1]

The humanReadableTokenSymbol property MUST be constructed as the hyphenated concatenation of first the tokenSymbol and then the chainId.

An example would be:

"tokenSymbol" = ETH;
"chainId" = 1;
"humanReadableTokenSymbol" = ETH-1;

[CR2]>[O1] testability: humanReadableTokenSymbol can be parsed and split based on existing open source packages and the result compared to the tokenSymbol and chainId used in the data schema.

The schema for a canonical token list is given below as follows and can be utilized as a JSON-LD schema if a JSON-LD context file is utilized (see [1] for a concrete example in the context of a standard):

```
{
   "$id": "https://github.com/eea-oasis/l2/schemas/CanonicalTokenList.json",
   "$schema": "https://json-schema.org/draft-07/schema#",
   "$comment": "{\"term\": \"CanonicalTokenList\", \"@id\": \"https://github.com/eea-oasis/l2#CanonicalTokenList\"}",
   "title": "CanonicalTokenList",
   "description": "Canonical Token List",
   "type": "object",
   "required": [
      "type",
      "tokenListId",
```



```
      "name",
      "createdAt",
      "updatedAt",
      "versions",
      "tokens"
      ],
      "properties": {
         "@context": {
            "type": "array"
         },
         "type": {
            "oneOf": [
               {
                  "type": "string"
               },
               {
                  "type": "array"
               }
            ],
            "examples": ["CanonicalTokenList"]
         },
         "tokenListId": {
            "$comment": "{\"term\": \"tokenListId\", \"@id\": \"https://schema.org/identifier\"}",
            "title": "tokenListId",
            "description": "A resolvable URI to the publicly accessible place where this list can be found following the
RFC 3986 standard.",
            "type": "string",
            "examples": ["https://ipfs.io/ipns/k51qzi5uqu5dkkciu33khkzbcmxtyhn376i1e83tya8kuy7z9euedzyr5nhoew"]
         },
         "name": {
            "$comment": "{\"term\": \"name\", \"@id\": \"https://schema.org/name\"}",
            "title": "name",
            "description": "Token List name",
            "type": "string",
            "examples": ["Aggregate Canonical Token List"]
         },
         "logoURI": {
            "$comment": "{\"term\": \"logoURI\", \"@id\": \"https://schema.org/identifier\"}",
            "title": "logoURI",
            "description": "URI or URL of the token list logo following the RFC 3986 standard",
            "type": "string",
            "examples": ["https://ipfs.io/ipns/k51qzi5uqu5dh5kbbff1ucw3ksphpy3vxx4en4dbtfh90pvw4mzd8nfm5r5fnl"]
         },
         "keywords": {
            "$comment": "{\"term\": \"keywords\", \"@id\": \"https://schema.org/DefinedTerm\"}",
            "title": "keywords",
            "description": "List of key words for the token list",
            "type": "array",
            "examples": [Aggregate Token List]
         },
         "createdAt": {
            "$comment": "{\"term\": \"createdAt\", \"@id\": \"https://schema.org/datePublished\"}",
            "title": "createdAt",
            "description": "Date and time token list was created",
            "type": "string",
            "examples": ["2022-05-08"]
         },
         "updatedAt": {
```

```
            "$comment": "{\"term\": \"updatedAt\", \"@id\": \"https://schema.org/dateModified\"}",
            "title": "updatedAt",
            "description": "Date and time token list was updated",
            "type": "string",
             "examples": ["2022-05-09"]
         },
         "versions": {
            "$comment": "{\"term\": \"versions\", \"@id\": \"https://schema.org/version\"}",
            "title": "versions",
            "description": "Versions of the canonical token list",
            "type": "array",
             "items": {
               "type":"object",
               "required":[
                  "major",
                  "minor",
                  "patch"
               ],
               "properties": {
                  "major": {
                     "$comment": "{\"term\": \"major\", \"@id\": \"https://schema.org/Number\"}",
                     "title": "major",
                     "description": "Major Version Number of the Token List",
                     "type": "integer",
                      "examples": [1]
                  },
                  "minor": {
                     "$comment": "{\"term\": \"minor\", \"@id\": \"https://schema.org/Number\"}",
                     "title": "minor",
                     "description": "Minor Version Number of the Token List",
                     "type": "integer",
                      "examples": [1]
                  },
                  "patch": {
                     "$comment": "{\"term\": \"patch\", \"@id\": \"https://schema.org/Number\"}",
                     "title": "patch",
                     "description": "Patch Number of the Token List",
                     "type": "integer",
                      "examples": [1]
                  },
               }
            }
         },
         "tokens": {
            "title": "Listed Token Entry",
            "description": "Listed Token Entry",
            "type": "array",
             "items": {
               "type":"object",
               "required": [
                  "chainId",
                  "chainURI",
                  "tokenId",
                  "tokenType",
                  "address",
                  "name",
                  "symbol",
                  "decimals",
```



```
                "createdAt",
                "updatedAt"
            ],
            "properties": {
                "chainId": {
                    "$comment": "{\"term\": \"chainId\", \"@id\": \"https://schema.org/identifier\"}",
                    "title": "chainId",
                    "description": "The typically used number identifier for the chain on which the token was issued.",
                    "type": "number",
                    "examples": [137]
                },
                "chainURI": {
                    "$comment": "{\"term\": \"chainURI\", \"@id\": \"https://schema.org/identifier\"}",
                    "title": "chainURI",
                    "description": "A resolvable URI to the genesis block of the chain on which the token was issued
following the RFC 3986 standard.",
                    "type": "string"
                     "examples": ["https://polygonscan.com/block/0"]
                },
                "genesisBlockHash": {
                    "$comment": "{\"term\": \"genesisBlockHash\", \"@id\": \"https://schema.org/sha256\"}",
                    "title": "genesisBlockHash",
                    "description": "The hash of the genesis block of the chain on which the token was issued.",
                    "type": "string",
                    "examples": ["0xa9c28ce2141b56c474f1dc504bee9b01eb1bd7d1a507580d5519d4437a97de1b"]
                },
                "tokenIssuerId": {
                    "$comment": "{\"term\": \"tokenIssuerId\", \"@id\": \"https://schema.org/identifier\"}",
                    "title": "tokenIssuerId",
                    "description": "A resolvable URI identifying the token issuer following the RFC 3986 standard.",
                    "type": "string",
                    "examples":
["https://polygonscan.com/address/0xa9c28ce2141b56c474f1dc504bee9b01eb1bd7d1a507580d5519d4437a97de1b"]

                },
                "tokenIssuerName": {
                    "$comment": "{\"term\": \"tokenIssuerName\", \"@id\": \"https://schema.org/name\"}",
                    "title": "tokenIssuerName",
                    "description": "The name oof the token issuer.",
                    "type": "string"
                    "examples": ["Matic"]
                },
                "tokenId": {
                    "$comment": "{\"term\": \"tokenId\", \"@id\": \"https://schema.org/identifier\"}",
                    "title": "tokenId",
                    "description": "A resolvable URI of the token following the RFC 3986 standard to for example the
deployment transaction of the token, or a DID identifying the token and its issuer.",
                    "type": "string",
                    "example": ["https://polygonscan.com/address/0x0000000000000000000000000000000000001010"]
                },
                "tokenType": {
                    "$comment": "{\"term\": \"tokenType\", \"@id\": \https://schema.org/StructuredValue\"}",
                    "title": "tokenType",
                    "description": "Describes the type of token.",
                    "type": "array"
                    "examples"[["fungible","transferable"]]
                },
                "tokenDesc": {
```

```
                    "$comment": "{\"term\": \"tokenDesc\", \"@id\": \"https://schema.org/description\"}",
                    "title": "tokenDesc",
                    "description": "Brief description of the token and its functionality.",
                    "type": "string",
                    "examples": ["Protocol Token for the Matic Network"]
                },
                "standard": {
                    "$comment": "{\"term\": \"standard\", \"@id\": \"https://schema.org/citation\"}",
                    "title": "standard",
                    "description": "A resolvable URI to the description of the token standard.",
                    "type": "string",
                    "examples": ["https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md"]
                },
                "address": {
                    "$comment": "{\"term\": \"address\", \"@id\": \"https://schema.org/identifier\"}",
                    "title": "address",
                    "description": "Address of the token smart contract.",
                    "type": "string",
                    "examples": ["0x0000000000000000000000000000000000001010"]
                },
                "addressType": {
                    "$comment": "{\"term\": \"address\", \"@id\": \"https://schema.org/Intangible\"}",
                    "title": "addressType",
                    "description": "AddressType of the token smart contract.",
                    "type": "string",
                    "examples": ["MaticNameSpace"]
                },
                "addressAlg": {
                    "$comment": "{\"term\": \"addressAlg\", \"@id\": \"https://schema.org/algorithm\"}",
                    "title": "addressAlg",
                    "description": "Algorithm used to create the address e.g. CREATE2 or the standard ethereum
address construction which is the last 40 characters/20 bytes of the Keccak-256 hash of a secp256k1 public key.",
                    "type": "string",
                    "examples": ["CREATE2"]
                },
                "name": {
                    "$comment": "{\"term\": \"name\", \"@id\": \"https://schema.org/name\"}",
                    "title": "name",
                    "description": "Token name.",
                    "type": "string",
                    "examples": ["Matic"]
                },
                "symbol": {
                    "$comment": "{\"term\": \"symbol\", \"@id\": \"https://schema.org/currency\"}",
                    "title": "symbol",
                    "description": "Token symbol e.g. ETH.",
                    "type": "string",
                    "examples": ["MATIC"]
                },
                "humanReadableTokenSymbol": {
                    "$comment": "{\"term\": \"humanReadableTokenSymbol\", \"@id\": \"https://schema.org/currency\"}",
                    "title": "humanReadableTokenSymbol",
                    "description": "A Token symbol e.g. ETH, concatenated with the `chainId` the token was issued on or
bridged to, e.g. ETH-1",
                    "type": "string",
                    "examples": ["MATIC-137"]
                },
                "decimals": {
```



```
                    "$comment": "{\"term\": \"decimals\", \"@id\": \"https://schema.org/Number\"}",
                    "title": "decimals",
                    "description": "Allowed number of decimals for the listed token. This property may be named differently by token standards e.g. granularity for ERC-777",
                    "type": "integer",
                    "examples": [18]
                },
                "logoURI": {
                    "$comment": "{\"term\": \"logoURI\", \"@id\": \"https://schema.org/identifier\"}",
                    "title": "logoURI",
                    "description": "URI or URL of the token logo following the RFC 3986 standard.",
                    "type": "string"
                    "examples": ["https://polygonscan.com/token/images/matic_32.png"]
                },
                "createdAt": {
                    "$comment": "{\"term\": \"createdAt\", \"@id\": \"https://schema.org/datePublished\"}",
                    "title": "createdAt",
                    "description": "Date and time token was created",
                    "type": "string",
                    "examples": ["2020-05-31"]
                },
                "updatedAt": {
                    "$comment": "{\"term\": \"updatedAt\", \"@id\": \"https://schema.org/dateModified\"}",
                    "title": "updatedAt",
                    "description": "Date and time token was updated",
                    "type": "string"
                    "examples": ["2020-05-31"]
                },
                "extensions": {
                    "title": "extensions",
                    "description": "Extension to the token list entry to specify an origin chain if the token entry refers to another chain other than the origin chain of the token",
                    "type": "array",
                    "items": {
                        "type":"object",
                        "required": [
                            "rootChainId",
                            "rootChainURI",
                            "rootAddress",
                        ],
                        "properties": {
                            "rootChainId": {
                                "$comment": "{\"term\": \"rootChainId\", \"@id\": \"https://schema.org/identifier\"}",
                                "title": "rootChainId",
                                "description": "The typically used number identifier for the root chain on which the token was originally issued.",
                                "type": "number",
                                "examples": [137]
                            },
                            "rootChainURI": {
                                "$comment": "{\"term\": \"rootChainURI\", \"@id\": \"https://schema.org/identifier\"}",
                                "title": "rootChainURI",
                                "description": "A resolvable URI to the genesis block of the root chain on which the token was originally issued following the RFC 3986 standard.",
                                "type": "string",
                                "examples": ["https://polygonscan.com/block/0"]
                            },
                            "rootAddress": {
```



```
                        "$comment": "{\"term\": \"rootAddress\", \"@id\": \"https://schema.org/identifier\"}",
                        "title": "rootAddress",
                        "description": "Root address of the token smart contract.",
                        "type": "string",
                        "examples": ["0x0000000000000000000000000000000000001010"]
                    }
                }
            }
        }
    }
  },
  "additionalProperties": false,
}
```

Data Schema Testability: As the above data schema follows a JSON/JSON-LD schema format, and since such formats are known to be testable for schema conformance (see for example the W3C CCG Traceability Work Item), the above data schema is testable.

---

4 Conformance

This section describes the conformance clauses and tests required to achieve an implementation that is provably conformant with the requirements in this document.

4.1 Conformance Targets

This document does not yet define a standardized set of test-fixtures with test inputs for all MUST, SHOULD, and MAY requirements with conditional MUST or SHOULD requirements.

A standardized set of test-fixtures with test inputs for all MUST, SHOULD, and MAY requirements with conditional MUST or SHOULD requirements is intended to be published with the next version of the standard.

4.2 Conformance Levels

This section specifies the conformance levels of this standard. The conformance levels offer implementers several levels of conformance. These can be used to establish competitive differentiation.

This document defines the conformance levels of a canonical token list as follows:

- **Level 1:** All MUST requirements are fulfilled by a specific implementation as proven by a test report that proves in an easily understandable manner the implementation's conformance with each requirement based on implementation-specific test-fixtures with implementation-specific test-fixture inputs.
- **Level 2:** All MUST and SHOULD requirements are fulfilled by a specific implementation as proven by a test report that proves in an easily understandable manner the implementation's conformance with each requirement based on implementation-specific test-fixtures with implementation-specific test-fixture inputs.
- **Level 3:** All MUST, SHOULD, and MAY requirements with conditional MUST or SHOULD requirements are fulfilled by a specific implementation as proven by a test report that proves in an easily understandable manner the implementation's conformance with each requirement based on implementation-specific test-fixtures with implementation-specific test-fixture inputs.

[D3]

A claim that a canonical token list implementation conforms to this specification SHOULD describe a testing procedure carried out for each requirement to which conformance is claimed, that justifies the claim with respect to that requirement.

[D3] testability: Since each of the non-conformance-target requirements in this documents is testable, so must be the

totality of the requirements in this document. Therefore, conformance tests for all requirements can exist, and can be described as required in [D3].

[R5]

A claim that a canonical token list implementation conforms to this specification at**Level 2** or higher MUST describe the testing procedure carried out for each requirement at **Level 2** or higher, that justifies the claim to that requirement.

[R5] testability: Since each of the non-conformance-target requirements in this documents is testable, so must be the totality of the requirements in this document. Therefore, conformance tests for all requirements can exist, be described, be built and implemented and results can be recorded as required in [R5].

Appendix A - References

This appendix contains the normative and non-normative references that are used in this document.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

A.1 Normative References

The following documents are referenced in such a way that some or all of their content constitute requirements of this document.

[RFC2119]

S. Bradner, Key words for use in RFCs to Indicate Requirement Levels,http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[JSONLD]

JSON-LD 1.1, M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, Pierre-Antoine Champin, N. Lindström, W3C Recommendation, July 2020 , https://www.w3.org/TR/2020/REC-json-ld11-20200716/. Latest version available at https://www.w3.org/TR/json-ld11/.

[RFC3986]

T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, IETF RFC 3986, January 2005, https://www.ietf.org/rfc/rfc3986.txt.

[EIP155]

Vitalik Buterin, “EIP-155: Simple replay attack protection,” Ethereum Improvement Proposals, no. 155, October 2016. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-155.

[EIP3220]

Weijia Zhang, Peter Robinson, “EIP-3220: Crosschain Identifier Specification [DRAFT],” Ethereum Improvement Proposals, no. 3220, October 2020. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-3220.

A.2 Non-Normative References

[W3C-DID]

Decentralized Identifiers (DIDs) v1.0, M. Sporny, D. Longley, M. Sabadello, D. Reed, O. Steele, C. Allen, W3C W3C Recommendation, July 2022, https://www.w3.org/TR/2022/REC-did-core-20220719/. Latest version available at https://www.w3.org/TR/did-core/.

[W3C-String-Meta]

Strings on the Web: Language and Direction Metadata, R. Ishida, A. Phillips, August 2022, https://www.w3.org/TR/string-meta/

[CVE-2021-42574]

NIST Publication, 2021, https://nvd.nist.gov/vuln/detail/CVE-2021-42574

Appendix B - Security Considerations

There are no additional security requirements apart from the warnings that URIs utilized in implementations of this standard might be direct to malicious resources such as websites, and that implementers should ensure that data utilized for a canonical token list is secure and correct. Since this standard is focused on a data schema and its data properties there are no additional security considerations from for example homoglyph attacks (see CVE-2021-42574).

B.1 Data Privacy

The standard does not set any requirements for compliance to jurisdiction legislation/regulations. It is the responsibility of the implementer to comply with applicable data privacy laws.

B.2 Production Readiness

The standard does not set any requirements for the use of specific applications/tools/libraries etc. The implementer should perform due diligence when selecting specific applications/tools/libraries.

B.3 Internationalization and Localization

The standard encourages implementers to follow the W3C “Strings on the Web: Language and Direction Metadata” best practices guide for identifying language and base direction for strings used on the Web wherever appropriate.

Appendix C - Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged.

Participants:

Tas Dienes

Kelvin Fichter

Andreas Freund

Daniel Shaw

Pavel Sinelnikov

Appendix D - Revision History

Revisions made since the initial stage of this numbered Version of this document have been tracked on Github.

Appendix E - Notices