Producing and Consuming DID Documents from the Abstract Data Model

2020-09-22—Justin, Markus, Drummond

Introduction

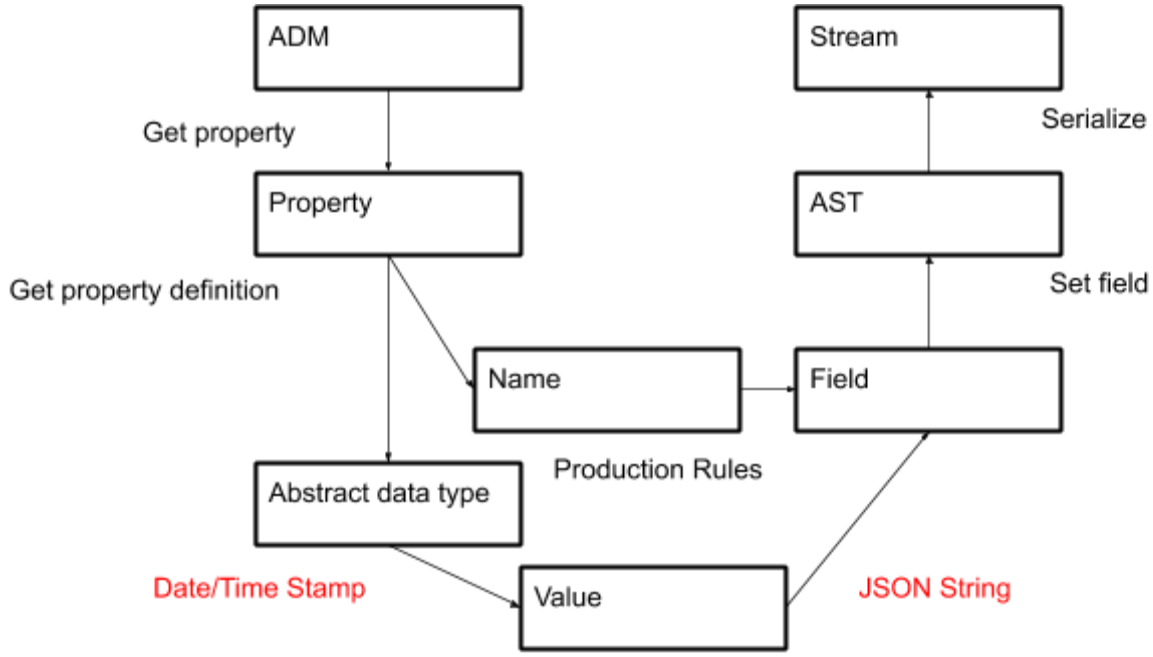
The purpose of this document is to seek alignment among the authors of the multiple issues and PRs dealing with the production and consumption of DID documents in different representations. This document proposes specific terminology and specific production and consumption steps to be specified in DID Core (and reflected as necessary in DID Spec Registries). These steps apply to all representations independent of any specific DID method.

Executive Summary

The proposal is to define the rules for producing and consuming DID documents in different representation types in terms of conversions to and from the abstract data model (ADM). The steps in this process are illustrated by the two diagrams below.

Production Function

The example value being converted here (red text) is a date/time stamp defined as an abstract data type in the ADM into a JSON string in a JSON representation.



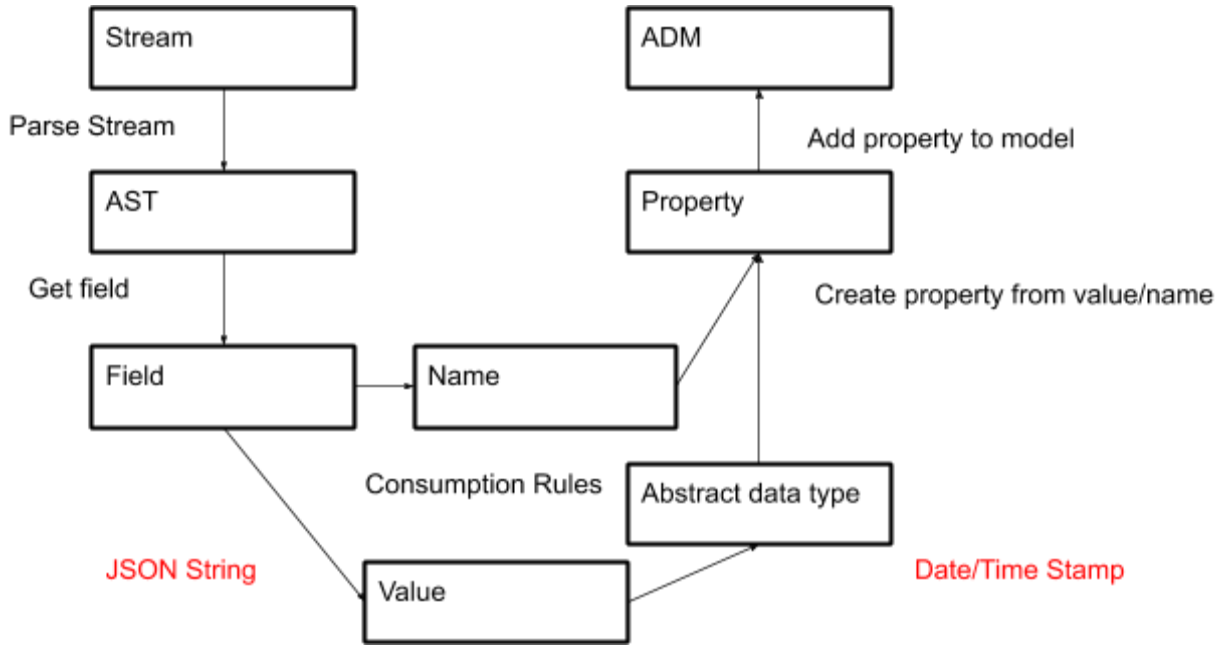
For each property in the ADM, the conversion function is:

(property name [string], property value [abstract data type])

-> (field name [representation label], field value [representation data type])

Consumption Function

This example is the reverse of the above.



For each field in the AST that is not a **representation-specific AST field**, the conversion function is:

(field name [representation label], field value [representation data type])

-> (property name [string], property value [abstract data type])

Terminology

These terms appear in **bold** in this document. It is not proposed that all these terms be added to the Terminology section of DID Core; a decision about that can be made after we have consensus on the way forward.

Abstract data type. A data type defined in the ADM so that any **ADM property** using this abstract data type can be represented in any representation. Complex types can be composed out of other types, such as an array of URIs.

ADM (abstract data model). The data model defined in Section 4 of the DID Core Specification. Specifically, it is a model for defining: a) **abstract data types,** and b) **ADM properties** (mapped to these abstract data types) so they are independent of any specific representation of a DID document.

ADM properties. Any property in a DID document that describes the DID subject and is independent of any specific representation. All ADM properties must have a **property definition** that maps the property to an **abstract data type**. ADM properties include **core properties**, **registered properties**, and **unregistered properties**. ADM properties do *not* include **representation-specific AST fields**.

AST (abstract syntax tree). An intermediate data representation format used in the **production steps** and **consumption steps** of DID documents.

AST fields. The name-value pairs used in the top level of the AST. The fields themselves are allowed to have their own structure as required by the definition of the **abstract data type** in the ADM.

Bytestream. The final serialization of a DID document in a specific representation type after completing all **production steps** and before beginning any **consumption steps**.

Consumption rules. The rules defined by a specific representation type for transforming **AST fields** into **ADM properties**.

Consumption steps. The set of steps taken by a consumer to transform a representation-specific **bytestream** into an **ADM** following the **consumption rules** for the specific representation type. (See the proposal below.)

Core properties. The **ADM properties** defined in Section 5 of the DID Core Specification.

Production rules. The rules defined by a specific representation type for transforming **ADM properties** into **AST fields**.

Production steps. The set of steps taken by a producer to transform a **ADM** into a representation-specific **bytestream** following the **production rules** for the specific representation type. (See the proposal below.)

Property definition. The definition of property name and **abstract data type** that MUST be defined for any **ADM property**.

Registered properties. All properties registered in the DID Spec Registries. The registered properties include all **core properties**.

Representation-specific AST fields. AST fields that are *not* **ADM properties**, but which are defined by specific representations for their own use. Examples of representation-specific AST fields include `@context` for JSON-LD representations and `$schema` for JSON-schema representations.

Unregistered properties. Any **ADM property** defined by a specific DID method or DID controller that is not a **registered property**. The DID method specification or DID controller is responsible for providing the **property definition** that maps to an **abstract data type**.

Production Steps

1. The producer generates the **ADM** for a specific DID document.
2. For each **ADM property**, the producer gets the **property definition** to determine the **abstract data type**.
3. The producer follows the **production rules** defined for the representation type to convert:
    a. The name (string) of the **ADM property** into the label of an **AST field**.
        i. If the resulting **AST field** is a **representation-specific AST field**, the producer throws an error.
    b. The value of the **ADM property** into the value of an **AST field** based on the **abstract data type** of the **ADM property** from its **property definition**.
4. The producer adds the **AST field** to the **AST**.
5. After all ADM properties are added, the producer follows the **production rules** to add any **representation-specific AST fields** to the **AST**.
6. The producer follows the **production rules** to serialize the **AST** into the **bytestream**.
7. The resulting **bytestream** is the **representation**.

Consumption Steps

1. The consumer follows the **consumption rules** for the representation type to deserialize (parse) the **bytestream** into an **AST**.
2. The consumer follows the **consumption rules** to process any **representation-specific AST fields**.
    a. Processing the **representation-specific AST fields** MAY alter the labels and values of other **AST fields**, for example by prepending a base context URL.
    b. If the consumer finds any unrecognized **representation-specific AST fields** (as declared by other registered representations), the consumer MUST throw an error.
3. For each remaining **AST field** (that is not a **representation-specific AST field**), the consumer follows the **consumption rules** to determine the **abstract data type**.
4. The consumer follows the **consumption rules** to convert:
    a. The label of the **AST field** into the name (string) of an **ADM property**.
    b. The value of the **AST field** into the value of an **ADM property**.
        i. If the consumer knows the **abstract data type** of the **ADM property** from its **property definition**, the consumer MUST convert the value as defined by the **abstract data type**.
        ii. If the consumer does not know the **abstract data type** of the **ADM property** from its **property definition**, the consumer MUST convert the value to a default **abstract data type** based on the type of the **AST field**, as defined by the **consumption rules**.

5. The producer adds the **ADM property** to the **ADM**.
6. The resulting **ADM** is the **DID Document**.

Normative Rules for Representation Types

1. Representation types MAY define their own **representation-specific AST fields**.
    a. Any **representation-specific AST fields** MUST be declared and registered with the representation.
2. Representation types MUST define the **production rules** and **consumption rules** for their representation type.
    a. These rules MUST cover all **abstract data types** defined in the **ADM**.
    b. The **consumption rules** MUST declare a default **abstract data type** for each possible data type in the **AST**.

Normative Rules for DID Methods

1. DID method specifications MAY define their own **registered properties** and/or **unregistered properties**.
    a. These properties MUST include a **property definition** mapping to an **abstract data type**.
2. DID method specifications MUST NOT modify the **production rules** or **consumption rules** for any representation type. In other words, production and consumption of DID documents into/from specific representation types MUST be entirely independent of any DID method.