Data Standards: Rosetta Stones for the Digital Age

Technical Brief
October 2024

Author: Dean Ritz, Senior Fellow, Data Foundation

Disclosure: The author made use of generative AI tools to assist his research and drafting.

Abstract

Data standards play a crucial role in ensuring efficient and accurate communication and information exchange, as they serve as the *"Rosetta Stone"* for data. Data standards consist of two parts: 1) a *conceptual model* that captures the meaning and relationships of each piece of data in a way that humans can understand, and 2) a *machine-readable implementation* that expresses the conceptual model in a format that computers can process and share. Data standardization efforts primarily, though not trivially, involve documenting in a machine-readable format the knowledge already understood by people, as reflected in existing information collection requirements and practices.

This technical paper on data standards provides some simple examples of machine-readable data standards lined up with the human-readable versions, showing that data standards are just a different form of documentation for the knowledge we already have.

Introduction

The Rosetta Stone was a significant archaeological discovery that helped unlock the mysteries of ancient Egyptian hieroglyphs. The stone contained the same message written in three different scripts: hieroglyphs, Demotic, and Ancient Greek. Since scholars already understood Ancient Greek, they used that as a key to decipher the other two scripts, particularly the hieroglyphs, which had been indecipherable for centuries.

The message was a royal decree. The King could have forced everyone to learn the same script but it was much easier to translate the message than to transform the linguistic skills of the population. The King's goal (likely) wasn't language training, it was getting a particular message transmitted to the people. The situation is similar for data standards. We create data standards because we have information (data) we want to make understandable to people and computers.[1]

A simplified Data Rosetta Stone, below, expresses two named concepts in three scripts: a human readable one, and two machine-readable ones. They are three faithful, consistent expressions of the same information.

XML and JSON are machine-readable data languages that organize data intro structures. They have different approaches. XML organizes data into a hierarchical structure using nested tags, which can be predefined in a template for consistent computer readability. JSON is a simpler approach that organizes its content into a key:value pairs. Both are widely used, and both provide a well-defined mechanism for organizing data into discrete, identifiable parts.

Human Standard expression	XML expression	JSON expression
First Name (string): The first name of the contact person.	<xs:element name="FirstName" type="xs:string"> <xs:annotation> <xs:documentation> The first name of the contact person. </xs:documentation> </xs:annotation> </xs:element>	"firstName": {"type": "string"}, {"description": "The first name of the contact person."}

[1] One could dismiss the need for data standards in the age of AI, but the opposite is true. AI increases the value of data standards. See the article, *[Why Data Standards Matter in the Age of AI](https://www.datafoundation.org/blog-list/why-data-standards-matter-in-the-age-of-ai)* (available here: https://www.datafoundation.org/blog-list/why-data-standards-matter-in-the-age-of-ai).

Last Name (string): The last name of the contact person.	`<xs:element name="LastName" type="xs:string">`
	`  <xs:annotation>`
	`   <xs:documentation> The last name of the contact person.`
	`   </xs:documentation>`
	`  </xs:annotation>`
	`</xs:element>`
		`"lastName":`
		`  {"type": "string"},`
		`  {"description": "The last name of the contact person."}`

A simplified Data Rosetta Stone

Data standards transform your documentation, one byte at a time, making people-readable information machine-readable.

Data Standards in US Law

Data standards should be considered organizational assets, and are so according to recent data laws.[2] The larger the organization, the more necessary data standards are to the organization, preventing the mountain of organization data from becoming a Tower of Babel. Together with Congress, the federal government has made concerted efforts to improve data quality, with an intentional and specific focus on the concept of data standards and their potential significance.[3,4]

The Foundations for Evidence-Based Policymaking Act of 2018 (Evidence Act) and the Financial Data Transparency Act of 2022 (FDTA) are two recent laws within a body of other data laws, policies, statutes, circulars, and best practices.

The Evidence Act was enacted to improve the use of data and evidence in policy making across federal agencies. It has several provisions that touch on data standards, especially in the context of data governance, sharing, and data-driven decision-making, including:

---

[2] Foundations for Evidence-Based Policymaking Act of 2018 (the Evidence Act), P.L. 115-435, https://datafoundation.org/news/key-laws-evidence-informed-decision-making/123/123-Evidence-Act

[3] Enterprise Data Inventories, Federal Chief Data Officer Council's Data Inventory Working Group, <https://resources.data.gov/assets/documents/CDOC_Data_Inventory_Report_Final.pdf>, "The Chief Data Officers Council Data Inventory Working Group developed this paper to highlight the value proposition for data inventories and describe challenges agencies may face when implementing and managing comprehensive data inventories…The purpose of the comprehensive data inventory is to ensure that agency staff and the public have a clear comprehensive understanding of the data assets in the possession of the agency." p. 1.

[4] Natalie R. Ortiz, "Federal Data Management: Issues and Challenges in the Use of Data Standards," Congressional Research Service, April 29, 2024 <https://crsreports.congress.gov/product/pdf/R/R48053/2>.

1. Open Data and Data Sharing:
    - Under Title II of the Evidence Act federal agencies are directed to make data open and accessible by default, unless there are prohibitions or certain limitations to doing so. To meet this requirement, agencies can standardize data to ensure interoperability and usability.
    - Agencies are required to develop and implement data inventories and ensure that they use open and machine-readable formats for data dissemination, which are data standards-related tasks.[5]
2. Chief Data Officers (CDOs) and Data Governance:
    - Title II of the Evidence Act also requires the appointment of Chief Data Officers in most federal agencies, who are senior officials responsible for developing strategies around data governance, which includes ensuring that data standards are implemented and maintained across their respective agencies.
3. Federal Data Strategy:
    - The Evidence Act aligns withFederal Data Strategy from the Office of Management and Budget (OMB).[6] The strategy promotes the use of data standards to ensure consistency in data collection, sharing, and use across the federal government.
4. Data Councils:
    - The Evidence Act establishes the Chief Data Officers Council, which focuses on improving data management practices, including the development of common data standards across the government.

In summary, the Evidence Act creates a framework that promotes the adoption and implementation of data standards across the federal government, enabling better data sharing, transparency, and evidence-informed policymaking.

The FDTA builds on the Evidence Act by requiring the members of the Financial Stability Oversight Council (FSOC) to use the organizational capacities of the Evidence Act to establish data standards for the member agencies, citing definitions from the Evidence Act to ensure a standard understanding and application of terms, like “metadata”, “machine-readable”, and “open Government data assets”.

Additionally, the FSOC uses data from its member agencies to monitor system system risks to the US financial system, to regulate systemically important financial institutions (SIFIs), promote market discipline, address emerging threats, facilitate coordination and cooperation among regulatory agencies, and advise on regulatory standards, putting a heavy emphasis on the need for data interoperability. A point reinforced by the FDTA.

---

[5] Enterprise Data Inventories, Federal Chief Data Officer Council’s Data Inventory Working Group, <https://resources.data.gov/assets/documents/CDOC_Data_Inventory_Report_Final.pdf>.
[6] <https://strategy.data.gov/>.

Data Standard Components

A ***data standard*** is a formally agreed-upon specification that establishes a consistent structure, format, and meaning for data elements within a particular domain or context. For example, a data standard about a particular concept (i.e., a specific, meaningful data point such as a phone number or the first and last name of a person) would consist of two essential components:

1. a ***conceptual model, which is semantic***, and
2. a ***machine-readable implementation, which is syntactic.***[7]

The ***conceptual model*** defines the semantic meaning of each data element, providing a clear understanding of what the data means and how it would be interpreted by a human expert. The concept captured by the model is *a priori* knowledge of it being expressed as a symbol or symbols readable by humans or machines. Therefore, a human expert's knowledge about a concept serves as a blueprint for the logical structure and organization of the data in both human-readable formats as well as machine-readable formats.

The ***machine-readable implementation***, on the other hand, translates the conceptual model into a specific *data format* such as XML (eXtensible Markup Language), JSON (JavaScript Object Notation), or the more sophisticated XBRL[8] (eXtensible Business Reporting Language) standard. Expressed in a data format, the conceptual model can be processed and shared by computer systems.

The *conceptual model* and the *machine-readable implementation* should be in agreement with each other–like a Rosetta Stone. In the migration from paper and PDF reports, which are not machine-readable data, to formats that are machine-readable data, data standards are the key multi-format documentation necessary to maintain quality and transparency in reporting, and to enable reliable, trustworthy automation.

Thus, the effort to create a data standard for existing collections of information is an effort to translate what we know as human experts into a format that permits it to be understood by computers, too. That entails finding a suitably expressive machine-readable syntax standard.

[7] Data languages have rules that define their structure and content, just like human language. These rules include the syntax, which specifies the format, and the grammar, which defines the allowed elements, their hierarchy, and value constraints. For simplicity, this article does not differentiate between syntax and grammar.
[8] The XBRL standard is a semantic data model that builds on XML. XBRL requires that contextual information such as balance type, units of measure, reporting period, etc. be prepared consistently and in a structured manner. It mitigates the data problem of having more than one way for expressing a particular thing.

Data Syntax

A data syntax standard ensures that the data is structured and encoded in a way that is universally understood and interoperable across different platforms and applications. Together, a conceptual model and its complementary machine-readable implementation form a comprehensive data standard that facilitates efficient data exchange, integration, and analysis.

Let's consider a simple example of an address book entry and how it can be represented using a data syntax standard. From the point of view of a human person, here is a conceptual model of an address book entry. It consists of the following items:

- First Name: The first name of the contact person. We would expect it to be a series of letters.
- Last Name: The last name of the contact person. This too, we would expect to be a series of letters.
- Email: The email address of the contact person. We would expect this to be a series of letters, symbols, and digits without any whitespace.
- Phone Number: The phone number of the contact person. We would expect this to be a series of digits and, optionally, some symbols (e.g., a dash “-”).

The conceptual model can be expressed as a machine-readable XML structure, or schema. Thus an XML schema is the use of the XML syntax to define a structure and data types of a conceptual model; in this example we use the conceptual model for an address book entry, as above:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="AddressBookEntry">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="FirstName" type="xs:string"/>
        <xs:element name="LastName" type="xs:string"/>
        <xs:element name="Email" type="xs:string"/>
        <xs:element name="PhoneNumber" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
<xs:schema>
```

In this XML schema:

- The root element is named "AddressBookEntry".
- The complex type defines the sequence of elements within an address book entry. This means the data file can hold multiple address book entries.

- Each element (FirstName, LastName, Email, PhoneNumber) is defined with a specific data type (xs:string).

Based on the above schema, an example of an address book entry in XML format would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<AddressBookEntry>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
  <Email>john.doe@example.com</Email>
  <PhoneNumber>123-456-7890</PhoneNumber>
</AddressBookEntry>
```

This XML data represents a specific instance of an address book entry, and it adheres to the structure defined in the XML schema. Our example data standard, consisting of the address book entry conceptual model and its XML schema implementation, ensures that the address book entry is consistently structured and can be easily processed and exchanged between different systems.

You may also think of the schema as a *specification* for how data should be written or described. This specification perspective is useful because it then makes sense that a data syntax standard would also have software available that does just that – validates data as conformant to a specification.

In fact, nearly all data syntax standards, including those mentioned above (XML, JSON, and XBRL) have open source software validators available. So simply by using an existing data syntax standard to express your specific data standard (e.g., address book entry) you also get automated data validation for free! Who wouldn't want free, automated quality assurance to validate that data conforms to its specification?

There is another flexibility benefit that comes from the separation of the conceptual model from the machine-readable implementation of the model: the benefit of having flexibility in the choice of a data syntax standard. Below we express our conceptual model as a JSON schema:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "firstName": {
      "type": "string"
    },
    "lastName": {
```

```
          "type": "string"
        },
        "email": {
          "type": "string"
        },
        "phoneNumber": {
          "type": "string"
        }
      },
      "required": ["firstName", "lastName", "email", "phoneNumber"]
    }
```

In this JSON schema:

- The root object represents an address book entry.
- Each property (firstName, lastName, email, phoneNumber) is defined with a specific data type (string).
- The "required" array specifies that all properties are mandatory for data entry.

Based on the above schema, an example of an address book entry in JSON format would be:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "phoneNumber": "123-456-7890"
}
```

It represents a specific instance of an address book entry, following the structure defined in the JSON schema.

Below is a table that explicitly matches our conceptual model with two machine-readable expressions of it:

Conceptual Model	XML standard	JSON standard
First Name (string): The first name of the contact person.	`<xs:element name="FirstName" type="xs:string"/>`	`"firstName": {"type": "string"},`
Last Name (string): The last name of the contact person.	`<xs:element name="LastName" type="xs:string"/>`	`"lastName": {"type": "string"},`

Email (string): The email address of the contact person.	<xs:element name="Email" type="xs:string"/>	"email": {"type": "string"},
Phone Number (string): The phone number of the contact person.	<xs:element name="PhoneNumber" type="xs:string"/>	"phoneNumber": {"type": "string"}

Commonalities between XML and JSON:

- Both XML and JSON provide a way to define the structure and data types of the elements in a data standard.
- The conceptual model remains the same regardless of the implementation format (e.g., XML or JSON).
- The machine-readable implementation (XML schema or JSON schema) serves as a contract that defines the expected structure and constraints of the data.
- Both formats allow for the representation of hierarchical and nested data structures.
- The actual data instances (XML data or JSON data) conform to the defined schema and represent specific occurrences of the data standard.

While XML and JSON have different syntaxes and characteristics, they both serve the purpose of representing structured data and enabling interoperability between systems. The choice between data syntax standards often depends on factors such as the specific requirements of the systems involved, the programming languages and tools being used, and the preferences of the developers and stakeholders.

We Build Standards with Other Standards

In the world of data standards, we rarely start from scratch. Instead, we build upon existing data standards to create new, more specialized ones. This layered approach allows for efficient and consistent development of data standards across various domains.

Another example of this layer is the ISO 4217 standard for currency codes. While ISO 4217 defines a standardized format for representing currencies, it is implemented using the XML standard. XML provides a syntactic foundation for creating structured, machine-readable data formats, and ISO 4217 leverages this foundation to define its specific currency codes and attributes.

Similarly, XBRL, the *de facto* standard for business reporting in a machine-readable format, is also built upon the XML standard. XBRL uses XML's syntax and structure to define its own taxonomy, which includes elements, attributes, and relationships tailored to financial reporting.

Furthermore, XBRL demonstrates the interoperability of data standards by defining standards for transmitting XBRL data in other data standard languages, such as CSV (Comma-Separated Values) and JSON. This allows XBRL data to be easily consumed and processed by systems that may not directly support XBRL but can handle these more widely-used data formats.

By building upon existing data standards, new standards can be developed more efficiently, ensuring consistency and compatibility with established systems and processes. This layered and interoperable approach to data standards enables the creation of a rich ecosystem of machine-readable data formats that can be used across various industries and domains.

Data standards transform your documentation, one byte at a time.

Conclusion

In summary, and similarly to the Rosetta Stone, data standards play a crucial role in enabling efficient communication and information exchange across various sectors. Data standards consist of two essential components: a conceptual model that defines the meaning and relationships of each data element, and a machine-readable implementation that translates the conceptual model into specific data syntax standards such as XML, JSON, XBRL, etc. From the point of view of federal agencies implementing data standards, this approach portrays the effort to develop data standards as a project to document conceptual models that already exist at agencies under current information collection requirements and practices.

Appendix: Advanced Examples

In this section, we further develop our examples of how data standards can capture increasing levels of detail and structure to illustrate how standards are divided into smaller, more manageable components.

Thus far, our example schemas do not place restrictions on the syntax of the phone number or the email address. They are just strings. Yet, for the benefit of data quality, we likely want to restrict the format of this data. Data standards help us do that. Below we present a revised XML schema that restricts the phone number to the common U.S. format of three numeric digits, a dash, three more numeric digits, another dash, and four final numeric digits. It then constrains the email address to the standard email address format.

The point of these examples is less to provide technical training and more to further demonstrate that data standards can express in a machine-readable implementation the knowledge that we have captured in our conceptual model. Here is the revised XML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="AddressBookEntry">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="FirstName" type="xs:string"/>
        <xs:element name="LastName" type="xs:string"/>
        <xs:element name="Email">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern
value="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="PhoneNumber">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern value="\d{3}-\d{3}-\d{4}"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

In this revised XML schema:

- The "PhoneNumber" element is defined with a custom simple type.
- The simple type is derived from the base type "xs:string" and applies a restriction.
- The restriction uses a regular expression pattern (\d{3}-\d{3}-\d{4}) to enforce the desired format:
    - \d{3} represents exactly three numeric digits.
    - - represents a literal dash character.
    - The pattern is repeated to match the specified format: three digits, a dash, three digits, a dash, and four digits.

In this updated XML schema:

- The "Email" element is defined with a custom simple type.
- The simple type is derived from the base type "xs:string" and applies a restriction.
- The restriction uses a regular expression pattern ([a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}) to enforce a basic email format:
    - [a-zA-Z0-9._%+-]+ represents one or more characters that can be alphanumeric, dot, underscore, percent, plus, or hyphen.
    - @ represents the literal "at" symbol.
    - [a-zA-Z0-9.-]+ represents one or more characters that can be alphanumeric, dot, or hyphen.
    - \. represents a literal dot character.
    - [a-zA-Z]{2,} represents two or more alphabetic characters for the domain extension.

Finally, to get to the point where we can see that data standards can be broken down into smaller parts, and then those parts are incorporated by reference. “Incorporated by reference” is a method used in the development of standards and documentation where an external document or standard is referenced within a new standard. This approach allows a new standard to make use of existing standards without needing to replicate all of their content, ensuring consistency and reducing redundancy. This is a best practice for these reasons:

1. Efficiency and Simplicity: Instead of repeating lengthy content from another standard, the new standard simply references it. The new standard is thus more concise and easier to manage.
2. Consistency and Updates: By referencing an external standard, any updates to that standard are automatically incorporated into the new standard without needing to amend the new standard each time. This ensures that the new standard remains up-to-date.
3. Authority and Acceptance: Referencing established standards can lend credibility and authority to the new standard, as it shows alignment with widely accepted practices and frameworks.

Using "incorporated by reference," standards developers create more streamlined, authoritative, and up-to-date documents without redundancy.

So, now let us add to our requirements the support for many phone numbers and the ability to classify the number as a particular type (e.g., home, work, mobile). We begin with a revised XML schema with a *separate* schema for the phone number with its phone number type:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="phone.xsd"/>

  <xs:element name="AddressBookEntry">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="FirstName" type="xs:string"/>
        <xs:element name="LastName" type="xs:string"/>
        <xs:element name="Email">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern
value="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="PhoneNumber" type="PhoneNumber"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

And here's the separate "phone.xsd" schema file:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="PhoneNumber">
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{3}-\d{3}-\d{4}"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="PhoneNumberType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Home"/>
```

```
      <xs:enumeration value="Work"/>
      <xs:enumeration value="Mobile"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="PhoneNumberWithType">
    <xs:sequence>
      <xs:element name="Number" type="PhoneNumber"/>
      <xs:element name="Type" type="PhoneNumberType"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

In the revised schema:

- The main schema file includes the "phone.xsd" schema using the <xs:include> element.
- The "PhoneNumber" element in the main schema now refers to the "PhoneNumber" type defined in the "phone.xsd" schema.
- The "phone.xsd" schema defines the "PhoneNumber" type with the same pattern restriction as before.
- A new "PhoneNumberType" simple type is defined in the "phone.xsd" schema, which restricts the values to "Home", "Work", or "Mobile".
- A new complex type "PhoneNumberWithType" is defined in the "phone.xsd" schema, which consists of a "Number" element of type "PhoneNumber" and a "Type" element of type "PhoneNumberType".

Based on the updated schema, an example of a valid address book entry in XML format would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<AddressBookEntry>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
  <Email>john.doe@example.com</Email>
  <PhoneNumber>123-456-7890</PhoneNumber>

</AddressBookEntry>
```

And an example using the "PhoneNumberWithType" complex type:

```
<?xml version="1.0" encoding="UTF-8"?>
<AddressBookEntry>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
```

```
    <Email>john.doe@example.com</Email>
    <PhoneNumber>
      <Number>123-456-7890</Number>
      <Type>Home</Type>
    </PhoneNumber>
  </AddressBookEntry>
```

This revised schema allows for a more modular and reusable structure by separating the phone number-related types into a separate schema file. It also introduces an additional phone number type to provide more information about the type of phone number being stored.