PhUSE US Connect 2019

Paper TT10

A Linked Data Model and Hypermedia API for CDISC SHARE 2.0

Sam Hume, CDISC, State College, PA, USA
Frederik Malfait, Nurocor, Wolfenschiessen, Switzerland
Julie Chason, CDISC, Exton, PA, USA

ABSTRACT

The CDISC SHARE Metadata Repository supports the curation and availability of the CDISC standards using a variety of media types and formats. In this paper we explain the architecture of SHARE 2.0 and its implementation of linked data principles. The SHARE 2.0 system supports a broader set of standards and content types, now including Questionnaires, Rating, Scales (QRS), value subsets, draft domains, and rules in addition to the full scope of foundational standards and controlled terminology. The SHARE 2.0 Model is based on the ISO 11179 standard for Metadata Registries (MDR) and implemented using the W3C Resource Description Framework (RDF). There is a natural fit between web-based HTTP requests for resources and information stored as resources in RDF. We show how RDF resources can be mapped and exposed through a REST based web API and how the graph-based nature of RDF translates into hypermedia capabilities of the SHARE 2.0 API.

SHARE 2.0 OBJECTIVES AND SCOPE

INTRODUCTION

This paper introduces the technical foundations of the CDISC SHARE 2.0 metadata repository (MDR). CDISC plans to release SHARE 2.0 during the first quarter of 2019. The SHARE 2.0 release marks a significant upgrade over the first version featuring a new technology stack, an expanded model, and new content. SHARE 2.0 makes the CDISC data standards metadata available as a linked data model accessed via a hypermedia API. Software clients access the expanded model using a correspondingly expanded API. The SHARE 2.0 MDR provides a cloud-based solution that supports the curation, management, and publication of the CDISC standards metadata in several machine-readable formats (Hume et al., 2018). The new SHARE 2.0 software runs on a technology stack that will be simpler to maintain and extend.

SHARE 2.0 DATA STANDARDS CONTENT

The SHARE team, part of the CDISC Data Science team, ported SHARE 1.0 content to the expanded SHARE 2.0 model. The SHARE team also curated and imported new standards metadata not available in SHARE 1.0. **Table 1** highlights much of the SHARE 2.0 content.

Table 1. Highlights of the metadata available in SHARE 2.0

SDTM v1.2	SDTMIG v3.2	ADaM OCCDS v1.0
SDTM v1.3	SDTMIG v3.3	ADaM ADAE v1.0
SDTM v1.4	SDTMIG-AP v1.0	ADaM BDS for TTE v1.0
SDTM v1.5	SENDIG v3.0	CDASH Model v1.0
SDTM v1.6	SENDIG v3.1	CDASHIG v1.1.1
SDTM v1.7	ADaM v2.1	CDASHIG v2.0
SDTMIG v3.1.2	ADaMIG v1.0	Controlled Terminology Packages 2014Q3 - 2018Q4
SDTMIG v3.1.3	ADaMIG v1.1	

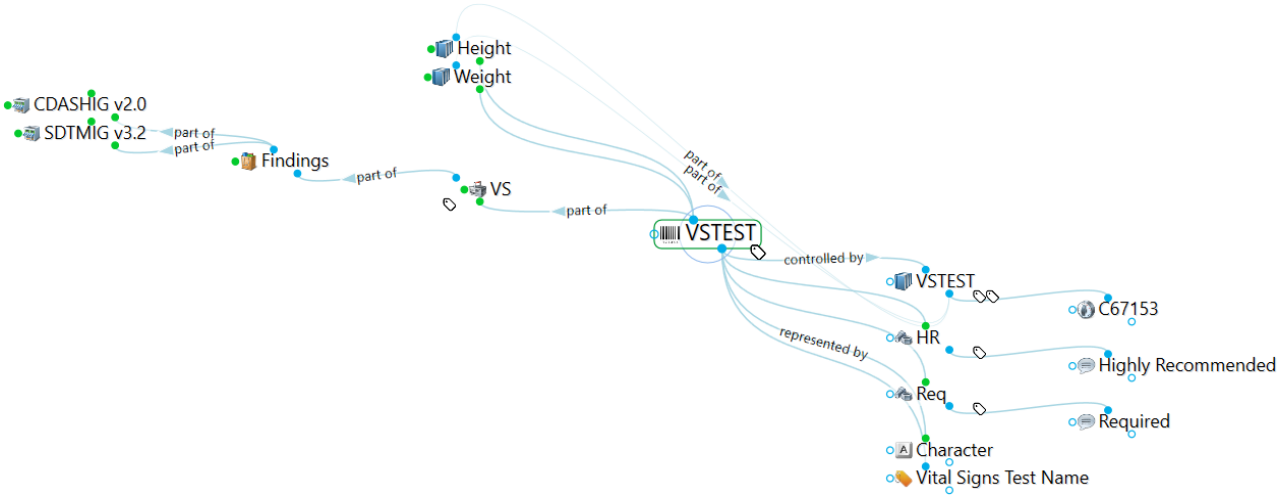
In addition to content made available for launch, the SHARE team created a quarterly content update schedule to publish new standards metadata that includes previously unavailable standards metadata as well as new versions of existing standards content. Examples of new content scheduled for loading into SHARE 2.0 in 2019 include:

standards conformance rules, standards validation rules, SDTMIG-MD v1.1, SDTM PGx v1.0, SEND-DART v1.1, SDTM QRS, ADaM ADQRS, ADaM OCCDS v1.1, controlled terminology code tables, diff files, and TA standards.

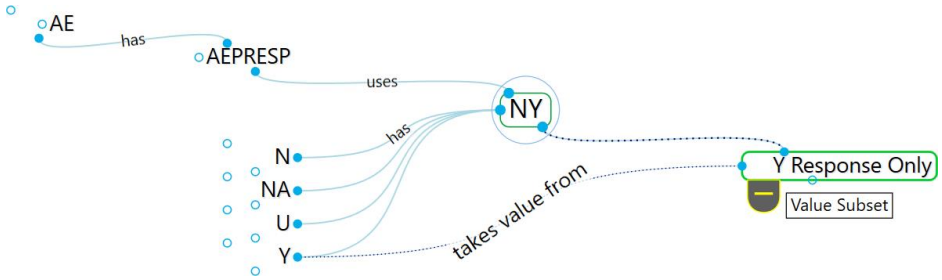
SHARE STANDARDS CONTENT CURATION

CDISC standards are generally described in documents that include explicit metadata. CDASHIG v2.0, for example, presents classes, domains, and variables, all with certain attributes. The metadata curator view of the standards is that we not only define these components of CDISC standards, but also explicitly represent their interrelations with other foundational standards.

Both versions of the CDISC SHARE metadata repository allow us to take these components and build them up in a database that captures not only these CDASHIG classes, variables and domains, but their relationships to the CDASH model, to CDISC Controlled Terminology, and to the variables found in the SDTMIG v.3.2. For example, the VSTEST variable is found in both CDASHIG v2.0 and SDTMIG v3.2; it is part of the VS domain which in turn is part of the Findings Observation class. The VSTEST variable is controlled by VSTEST Controlled Terminology, which has a C-Code C67153. “Height” and “Weight” are two of the values available in the VSTEST codelist. All of this information and more can be represented in the MDR.



This multidimensional representation of CDISC Foundational Standards provides us opportunities to address numerous areas identified by CDISC standards users and development teams, such as subsetting of Controlled Terminology, value level metadata, and conditional controlled terminology. For example, we will have the capability to explicitly represent the specific values of the “NY” codelist that are appropriate for use by the AEPRESP variable.



In order to represent areas such as subsetting of controlled terminology, the metadata curators depend on the standards development teams to formally state the desired metadata and for the information to be vetted under the CDISC publication process. Moreover, as the amount of CDISC metadata and their relationships increase, the curation effort increases, so it is important that curators be involved early in the process to ensure that the information is presented in a way that can be absorbed in the repository. One contribution that the curators can make is to develop structured metadata capture tools. Such tools will reduce the need for interpretation and post hoc consultation with standards development experts.

FROM SHARE 1.0 TO SHARE 2.0

CDISC SHARE 1.0, released in 2014, served essentially the same objectives as SHARE 2.0, using a graph-based model to represent the CDISC standards metadata. SHARE 1.0, however, consisted of an entirely different technology stack comprised of a Java Enterprise Edition application backed by an Oracle database, running in JBoss middleware. SHARE 1.0 implemented the Object Management Group's (OMG) Reusable Asset Specification (RAS) as a base model upon which the SHARE metamodel was layered. SHARE 1.0 organized content in layers of increasingly concrete models, starting with the OMG RAS as the foundational model, next implementing International Standards Organization (ISO)/International Electro-technical Commission (IEC) 11179 as the metamodel, and then adding a CDISC standards model as top-layer. ISO/IEC 11179 forms the foundational basis of CDISC SHARE's metamodel to structure information about metadata assets, versioning, ownership and other core metadata information. RAS and ISO/IEC 11179 are standards commonly implemented in metadata management systems, but the SHARE 1.0 model was novel as it was the first time CDISC structured their standards into a one common model.

SHARE 2.0, released in the first quarter of 2019, provides an expanded and improved model for CDISC standards metadata, and includes the adapted SHARE 1.0 content along with significant new standards content. SHARE 2.0 implements the graph model using the Resource Description Framework (RDF) and Web Ontology Language (OWL) W3C semantic web standards. Decommissioning of SHARE 1.0 MDR will be completed at the time of the SHARE 2.0 release and after porting all the relevant content to SHARE 2.0.

SHARE 2.0 ECOSYSTEM

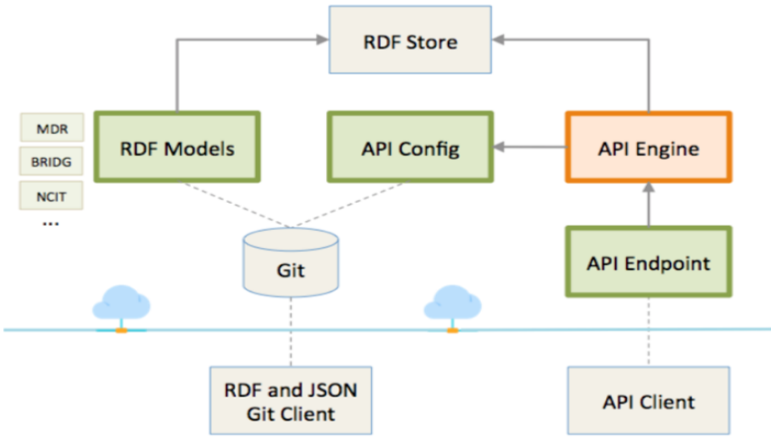
SHARE 2.0 consists of more than an MDR and API. The SHARE 2.0 ecosystem includes tools for the standards teams to create standards, quality check standards, generate value added standards content, generate content for loading into the SHARE triple store, and to quality check SHARE content.

Tools for creating standards content include, for example, the QRS Maker application and the NSV Registry, and standards developers use them to create new standards content. This simplifies the process of creating standards metadata that is ready for loading into the SHARE triple store and supports the CDISC goal of creating all standards as metadata for loading into SHARE. Tools exist to generate controlled terminology and QRS SHARE 2.0 load files from published standards metadata. This provides an automated pipeline for loading SHARE content directly generated from standards content. Tools like the Wiki-based Metadata QC Macros, Preamble Checker, and numerous API automated quality tools support quality control checks for SHARE content. The full suite of tools available to standards developers and SHARE curators help automate the process of generating and quality checking standards metadata for SHARE 2.0.

SHARE 2.0 PLATFORM COMPONENTS

The software that directly implements the SHARE 2.0 MDR software and API offers a platform for loading, storing, and delivering standards content to clients via the API. This platform, shown in the figure below, forms the heart of the SHARE 2.0 ecosystem. The SHARE team implements the SHARE model and transforms metadata developed as part of a CDISC standard for loading into the SHARE MDR. Content loaded into the SHARE RDF store, or triple store, supplies the models and standards content contained in the SHARE API. A Git client uses the Git API to move this content through the content load pipeline. The SHARE team also creates an API configuration file that specifies the API endpoints and the associated content returned to satisfy API requests. The API Engine renders these endpoints based on the API specification. The API specification contains the information needed to generate the SPARQL queries that run against the RDF triple store and retrieve the content to satisfy the client request.

SHARE 2.0 System Diagram

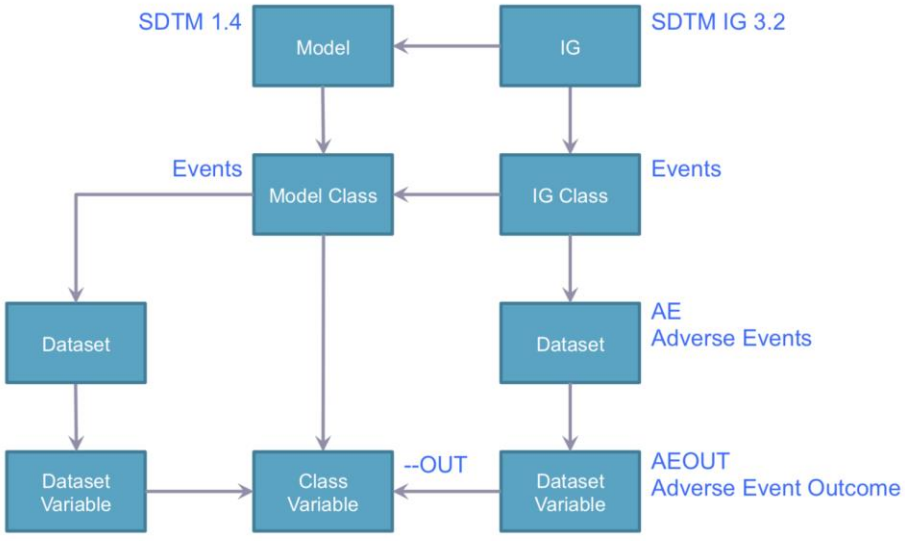


LINKED DATA MODEL

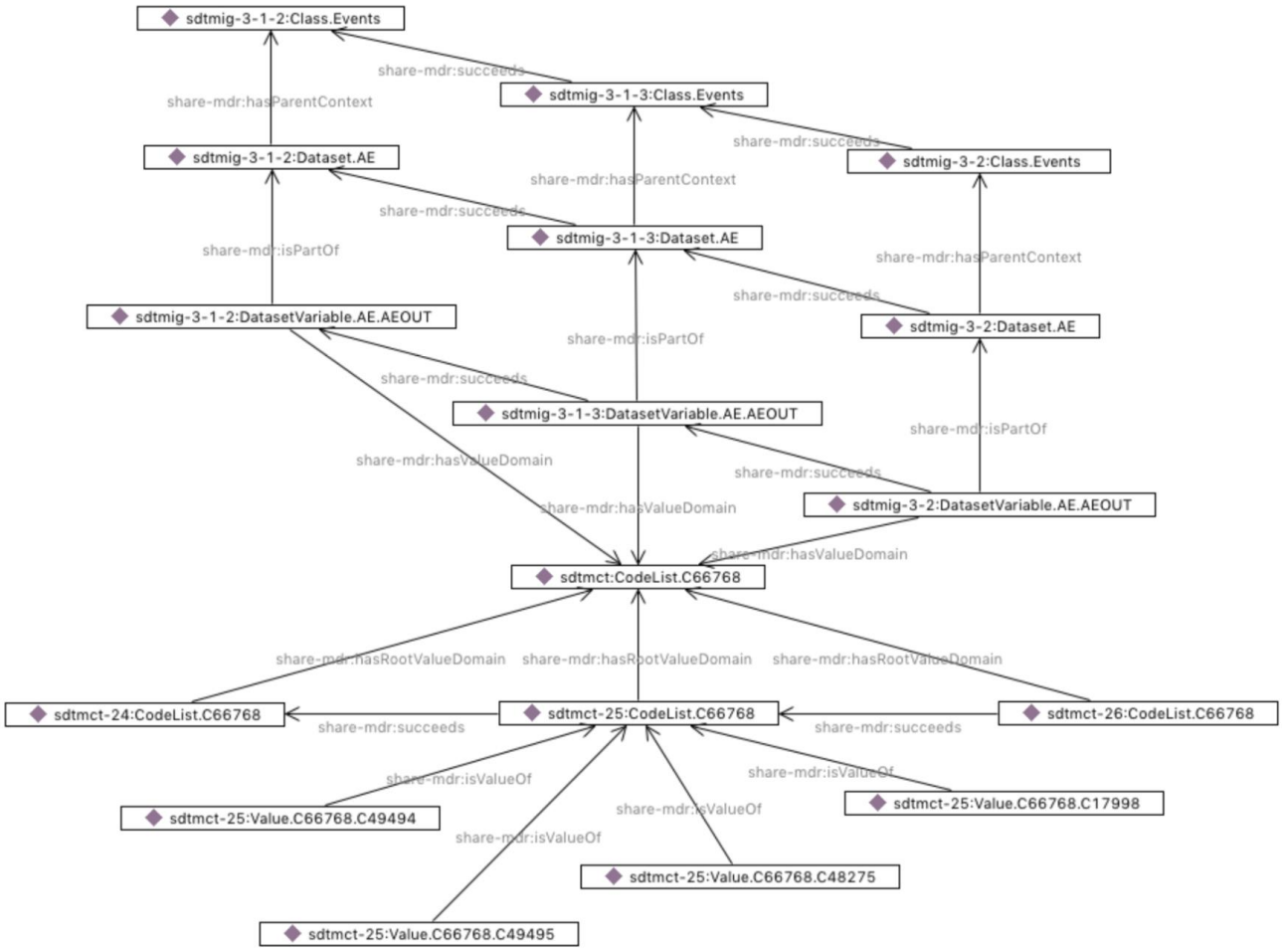
INTRODUCTION TO RDF

SHARE 2.0 uses RDF to implement its data model. RDF is a World Wide Web Consortium standard that can be used to define data models represented as facts consisting of 3-parts: the subject, predicate, and object. These 3-part facts, or triples, are the building blocks of any RDF data model. Triples can express properties of resources or express relationships (links) between resources. The result is a graph-based model where (subject and object) resources represent the nodes of the graph and predicates (links) represent the edges of the graph.

The following example shows a conceptual graph representing the relationships between SDTM model and SDTM IG elements.



The following example is an RDF graph showing resources that represent SDTM elements such as classes, datasets, variables, code lists, and code list elements. The edges represent RDF predicates (relationships or links between resources).



The Web Ontology Language (OWL) extends RDF with additional schema elements that support more extensive knowledge management. Ontologies provide a useful abstraction for representing knowledge facts and their relationships. OWL supports class abstractions, class hierarchies, taxonomies, and conceptual models.

RDF serialization standards define a number of formats for representing RDF content for the purpose of storage or exchange with other systems. RDF supports a broad range of serialization formats, including RDF/XML, Turtle, N-Triples, JSON-LD, and others. Following is an example of a Turtle serialization as used in SHARE 2.0.

```
sdtmig-3-2:DatasetVariable.AE.AEOUT
  rdf:type share-std:DatasetVariable ;
  share-mdr:hasOwner sdtmig-3-2:Product ;
  share-mdr:succeeds sdtmig-3-1-3:DatasetVariable.AE.AEOUT ;
  share-mdr:hasRootDataElement sdtm:Root.AE.AEOUT ;
  share-std:implementsVariable sdtm-1-4:ClassVariable.Events.--OUT ;
  share-mdr:isPartOf sdtmig-3-2:Dataset.AE ;
  share-mdr:ordinal "33"^^xsd:integer ;
  share-mdr:name "AEOUT"^^xsd:string ;
  share-mdr:label "Outcome of Adverse Event"^^xsd:string ;
  share-mdr:description "Description of the outcome of an event."^^xsd:string ;
  share-mdr:hasValueDomain sdtmct:CodeList.C66768 ;
  share-std:hasSimpleDatatype share-std:SimpleDatatype.Character ;
  share-std:hasRole share-std:Role.SDTM.RecordQualifier ;
  share-std:hasCore share-std:Core.SDTM.Permissible ;
.
```

LINKED DATA PRINCIPLES

Tim Berners-Lee presented The Next Web at the TED 2009 conference (https://www.ted.com/talks/tim_berners_lee_on_the_next_web) where he provided the linked data principles as three very simple rules:

1. Linked data represents all kinds of conceptual things, and they have names that start with HTTP. Names that start with HTTP are Uniform Resource Locators (URLs).
2. If one were to look up one of these URLs, or HTTP names, they would get back some useful data about that thing or event in a standard format.
3. The information returned includes not only facts, but also relationships. These relationships link content to other things that also have names represented as URLs. The ability to use these links to retrieve related content supports the *follow your nose* pattern.

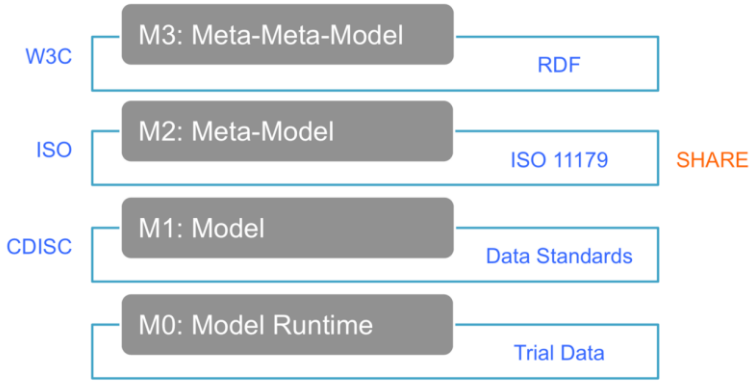
The Uniform Resource Identifier, or URI, plays an essential role in linked data. The URI provides a well-understood, reliably unique way of naming and identifying entities, including those within the CDISC standards. All URLs are URIs, but not all URIs are URLs. URIs share a common format with URLs, but URLs describe the location of a resource on the web in addition to identifying the resource. URIs can uniquely identify resources but may not necessarily reference a location where a representation of those resources can be retrieved. Content returned via the SHARE API includes URLs that identify content related to the retrieved entities, and those URLs provide the means to retrieve the related content via a subsequent API request.

ISO/IEC 11179

The ISO/IEC 11179 Metadata Registry standard is a widely implemented international standard for organizing and representing the semantics and representation of metadata. The standard facilitates the re-use of metadata through the promotion of common meaning and standard descriptions of the metadata. ISO/IEC 11179 has been used as a foundation for the implementation of registry components that support the development of data standards. Within the SHARE MDR, the ISO/IEC 11179-3 Edition 3 standard provides the basis for the metamodel used to represent the CDISC standards metadata. It provides the means of specifying the semantics as well as the representation for the standards metadata. ISO/IEC 11179-3 seeks to promote a common understanding of metadata, to enable re-use and facilitate harmonization.

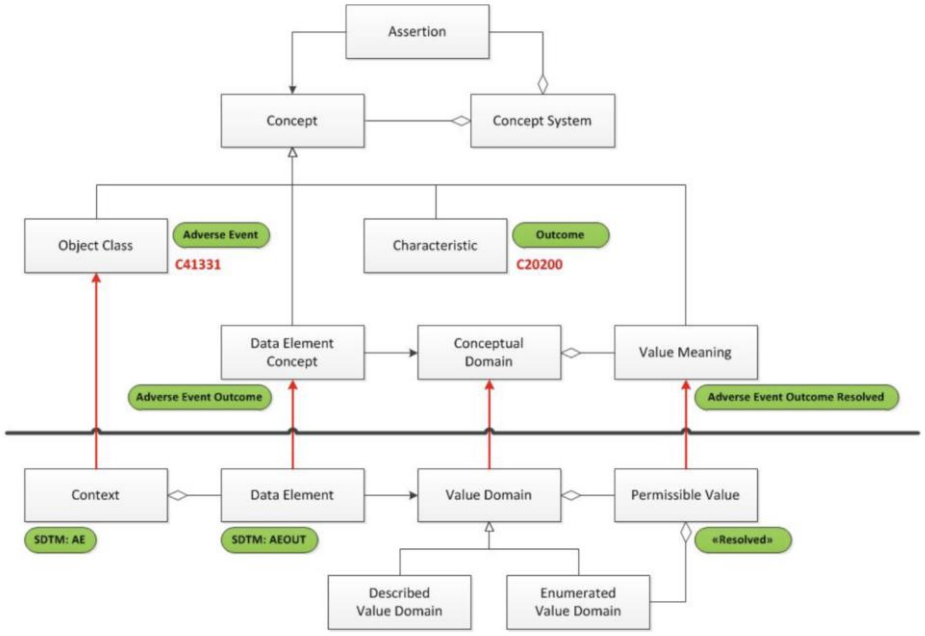
SHARE 2.0 MODEL

SHARE 2.0 uses RDF and OWL as a language to represent the metadata model and the metadata content. The metadata model follows ISO/IEC 11179, with extensions that fully support the SHARE 2.0 use cases. The extended model is sufficient to express all of the CDISC data standards in SHARE 2.0. The following schema shows how the different abstraction layers map to the Meta-Object Facility (MOF) standard defined by the Object Management Group (OMG).



It should be noted that the full framework follows standards all the way through (OMG, W3C, ISO, and CDISC).

The following UML diagram shows the key elements of ISO 11179 annotated with an example from the SHARE 2.0 content. The SHARE team plans to add the concept layer, including the Concept System, Concept, and Assertion classes, after the initial release of SHARE 2.0 and in conjunction with the CDISC 360 project.



EXTENDING THE SHARE MODEL

It was mentioned that the SHARE 2.0 model follows ISO 11179 with extensions that account for all of the SHARE 2.0 use cases. The following OWL class hierarchy is part of the SHARE 2.0 RDF model and illustrates this point.

Classes in the share-mdr namespace correspond with ISO 11179 model elements, whereas subclasses in the share-std namespace represent ISO 11179 extensions for specific SHARE 2.0 use cases.

HYPERMEDIA API

INTRODUCTION TO API PRINCIPLES

SHARE 2.0 features a Representational State Transfer (REST) API for accessing the standards content in the SHARE 2.0 model. REST is not a standard or protocol, but instead is an architectural style for networked applications. REST APIs provide an interface to content or services intended for consumption by machines. As an architectural style, REST describes the software engineering principles and constraints loosely followed by most implementations. Highlights of the REST principles and constraints include:

- A client server architectural style that implements the solution as a layered system for which REST provides the interface. Clients cannot see beyond the provided interface. This constraint supports the separation of concerns principle.
- A resource provides the source of specific information and can include a global permanent identifier such as can be provided by an HTTP Uniform Resource Identifier (URI). A URI uniquely identifies a resource, and makes it addressable, or capable of being manipulated using a protocol such as HTTP. Endpoints are URIs or Uniform Resource Locators (URLs).
- REST requires a standard interface, such as that provided by HTTP, to support the principle of generality. With HTTP, for example, a relatively small number of verbs with well-defined and widely accepted semantics meet the needs of most distributed applications. The interface provides the means to exchange representations of resources. In the case of HTTP, the protocol provides common, well-understood verbs, status codes, and content type.
- REST servers should be stateless to support the architecture principles of visibility, reliability, and scalability. Stateless servers require that each request from client to server contain the information necessary to understand and complete the request without requiring that state information be stored on the server.
- REST responses should be cacheable to improve performance.
- REST supports content negotiation such that clients can request specific content and media types.
- REST uses hypermedia as the engine of application state (HATEOAS) where responses include URLs relevant to the context of the requested resource. A hypermedia-driven system provides information to navigate the system's REST interfaces dynamically by including hypermedia links with the responses.

HYPERMEDIA DESIGN TRADEOFFS

The SHARE 2.0 API implements HATEOAS using hypermedia as the engine of application state. In essence, HATEOAS enables developers to traverse the linked data model as the relationships to related content exist as URLs in the response content. This provides a relatively simple and programming language agnostic means of traversing the graph-based SHARE content. SHARE generates the URLs for all graph nodes in the returned content.

In cases where SHARE returns a large number of nodes this can create a significant processing burden for the server and creates a much larger content response. In the case of recent SDTM controlled terminology packages, for example, SHARE generates URLs for each code list and term included in the package slowing response times and bloating the response content. In cases where a client requests a full controlled terminology package, assuming this content is not used to traverse the graph, SHARE disables URL generation for the terms. When individual code lists or terms are quested, SHARE generates the URLs assuming a user might be working with the content interactively.

In general, when a client requests full standards or controlled terminology packages, SHARE assumes the intended purpose is content retrieval rather than interactive navigation and disables deep linking. Deep linking represents nested content nodes within the returned metadata. This policy improves performance and scalability.

MEDIA TYPES

API clients specify their preferred media types in the HTTP Accept header. SHARE 2.0 supports media types similar to those supported in SHARE 1.0. A broad range of media types are supported by the technology platform, and the most commonly requested types have been made available during the initial release of SHARE 2.0.

Most API developers prefer JSON, and the application/json media type was the first media type implemented for SHARE 2.0. SHARE 2.0 also supports XML as application/xml, another broadly used media type for integrating systems. As a CDISC Data Exchange Standard, clients can request an extended version of ODM. ODM v1.3.2 now has its own standard media type (https://www.iana.org/assignments/media-types/media-types.xhtml). Since ODM

v1.3.2 is based on XML, the media type application/odm+xml should be used to request content as ODM. ODM provides the base model for a number of extensions, including Define-XML. When dataset metadata is requested in ODM, the Define-XML extensions are included. Additional metadata that is not part of the content standards, but available in SHARE 2.0, will be added to the ODM content via a new extension.

Beyond JSON and XML, SHARE 2.0 enables users to request content in CSV or Excel formats. While less commonly used by REST developers, these media types represent an easy format for SAS programmers and other developers working in statistical computing environments to implement.

SHARE 2.0 API

REST provides the interface to the SHARE 2.0 model and standards content. It provides a separation of concerns by requiring that clients use the API to access the application and data layers of the SHARE architecture. This enables the other layers to evolve independently from the interface, simplifies each of the layers, and can improve scalability. It also simplifies access to standards content and allows the widest range of clients to consume SHARE 2.0 standards content and services. Participants in loosely coupled distributed applications, such as those developed using SHARE 2.0 and its RESTful API, are free to work with the content they receive in any way they wish.

The SHARE 2.0 REST API specification exists on SwaggerHub in a machine-readable format that simplifies generating documentation and clients from the specification (https://app.swaggerhub.com/apis/CDISC1/share-2.0/1.0.6). SwaggerHub uses the OpenAPI standard to represent the SHARE API specification including the URL paths, path variables, query parameters, HTTP verbs, response codes, response body content, and other information needed to work with the API.

The SHARE 2.0 REST API specification provides the URLs required to request specific resources from SHARE 2.0. The API provides access to all the standards metadata available in the SHARE 2.0 MDR as well as the relationships between the metadata using HATEOAS. The API provides access to multiple versions of the foundational standards and supporting content, including: SDTM model, SDTMIG, SENDIG, CDASH model, CDASHIG, ADaM model, ADaMIG, corrigenda, product list, value lists, controlled terminology, QRS, TA specifications, rules, and search. **Table 2** below highlights how to construct URLs to request SDTM and SDTMIG resources.

Table 2. SDTM and SDTMIG API examples

SDTM API	**Description**
/sdtm/{version}	Get SDTM Product
/sdtm/{version}/classes	Get SDTM Class List
/sdtm/{version}/classes/{class}	Get SDTM Class
/sdtm/{version}/classes/{class}/variables	Get SDTM Class Variable List
/sdtm/{version}/classes/{class}/variables/{var}	Get SDTM Class Variable
/sdtm/{version}/classes/{class}/datasets	Get SDTM Class Dataset List
/sdtm/{version}/datasets	Get SDTM Dataset List
/sdtm/{version}/datasets/{dataset}	Get SDTM Dataset
/sdtm/{version}/datasets/{dataset}/variables	Get SDTM Dataset Variable List
/sdtm/{version}/datasets/{dataset}/variables/{var}	Get SDTM Dataset Variable
/root/sdtm/classes/{class}/variables/{var}	Get Root SDTM Class Variable
/root/sdtm/datasets/{dataset}/variables/{var}	Get Root SDTM Dataset Variable
/sdtm/{version}/corrigenda	Get SDTM Corrigenda
SDTMIG API	**Description**
/sdtmig/{version}	Get SDTMIG Product
/sdtmig/{version}/classes	Get SDTMIG Class List
/sdtmig/{version}/classes/{class}	Get SDTMIG Class
/sdtmig/{version}/classes/{class}/datasets	Get SDTMIG Class Dataset List
/sdtmig/{version}/datasets	Get SDTMIG Dataset List
/sdtmig/{version}/datasets/{dataset}	Get SDTMIG Dataset
/sdtmig/{version}/datasets/{dataset}/variables	Get SDTMIG Dataset Variable List
/sdtmig/{version}/datasets/{dataset}/variables/{var}	Get SDTMIG Dataset Variable
/root/sdtmig/datasets/{dataset}/variables/{var}	Get Root SDTMIG Dataset Variable
/sdtmig/{version}/corrigenda	Get SDTMIG Corrigenda

Table 2 above shows that model content requests can occur separate from IG content requests and that the structure of the requests largely mirror one another. Furthermore, this structure remains consistent across the foundational standards simplifying process of implementing API clients.

SHARE 2.0 PLATFORM

SHARE 2.0 is deployed on Nurocor's cloud-based model driven service platform (MDSP) that provides the following capabilities for ease of implementation.

MODEL DRIVEN API SERVICES

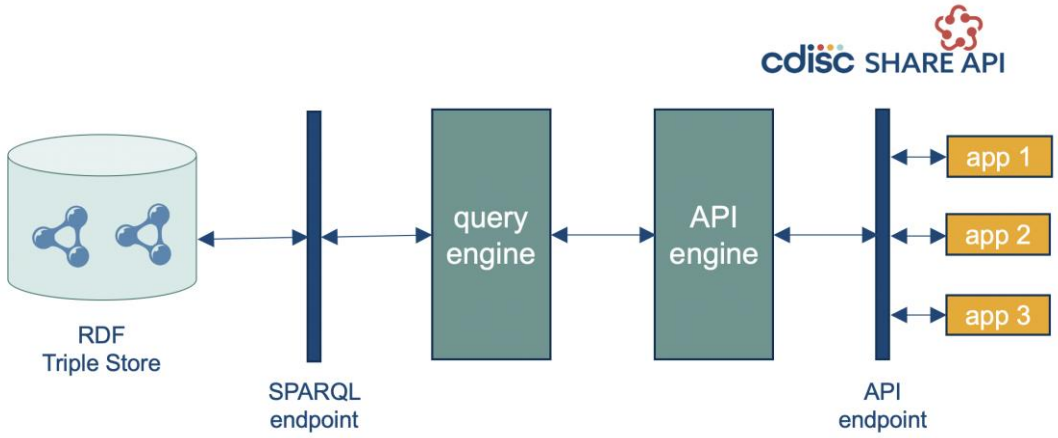
The SHARE 2.0 RDF implementation includes a broad set of standards where each type of standard defines its own specific set of attributes with a corresponding API endpoint (Controlled Terminology, CDASH Model, CDASHIG, SDTM Model, SDTMIG, ADaM, QRS, Rules, TA User Guide). Each API endpoint also comes with a resource/sub-resource hierarchy (e.g. product, class, dataset, variable), with multiple options for sub-resource inclusion (expanded versus hypermedia links), and a rich set of media types. To avoid long implementation cycles that rely heavily on developer availability, the platform allows for full configuration of the API where the API resource design can faithfully represent the RDF resource structure. In this sense, the RDF model drives the API in terms of naming, representing, and linking HTTP resources.

MAPPING A REST API TO A LINKED DATA MODEL

Mapping a REST based API to a data backend is traditionally a fairly complex task that requires custom programming. This complexity is avoided using a graph-based data backend such as RDF.

- HTTP resources map to RDF resources.
- The REST representation of a requested HTTP resource corresponds to a view on the RDF resource.
- A view is defined by a set of data selectors where each selector specifies a path from the targeted RDF resource to the required property. It also defines the representation for the requested media type.
- Navigating a hypermedia linked data API translates into following a path through the RDF graph.

This information can be defined in a set of API endpoint configurations. API requests are served by an API engine that uses the configuration to retrieve the information from an RDF triple store and format the response according to the API endpoint specification.



PUBLISHING PROCESS

As CDISC publishes new standards, the SHARE platform must be able to support this standards metadata publication process. The platform supports two types of publication requests.

- Updates may be required to the triple store in case RDF graphs for new standards need to be loaded into the triple store.
- Updates may be required to the API configuration in case new endpoints need to be added or existing endpoints need to be updated.

The RDF graphs and API endpoint configurations are created and maintained by CDISC in a Git repository that allows for managing different environments in a version-controlled way. The cloud-based Nurocor platform can pull from this Git repository to publish the required version of the RDF graphs and API configuration.

All operations are handled through the platform Admin API to drive the publication pipeline:

- git delete, git clone, git pull, and git log requests
- publish RDF triple store
- publish API endpoint configuration
- view published API endpoint modules
- view currently running API requests

Updating and redeploying a new API endpoint configuration is executed dynamically without the need for a server shutdown or restart.

TECHNICAL CONSIDERATIONS

The platform is cloud-based and all graph operations are based on the W3C suite of RDF specifications, with the intention to guarantee open standards, avoid vendor lock-in, and ensure platform and tool interoperability.

User management, authentication, and authorization is handled by an LDAP server. Authorization is defined by a permission model where permissions represent actions (HTTP verbs) on resource groups identified by URI patterns with path parameters. Hence, the authorization model follows closely the REST design of the API endpoint.

The current implementation of the platform publishes the full SHARE content of 360,000 resources with 4,138,000 triples under three minutes. Publication of a new API endpoint configuration using the Admin API takes a few seconds. Small data requests complete sub-second. Medium data requests can take from a few seconds to thirty seconds to compute depending on the number of sub-resources that must be included in the response. For larger requests, the API configuration offers the option to enable caching on an endpoint basis. Once cached, API requests for top level standards return sub-second with an additional lag time depending on network latency.

REFERENCES

Hume, S., Chow, A., Evans, J., Malfait, F., Chason, J., Wold, J. D., Kubick, W., & Becnel, L. B. (2018). CDISC SHARE, a Global, Cloud-based Resource of Machine-Readable CDISC Standards for Clinical and Translational Research. *AMIA Summits on Translational Science Proceedings*, *2017*, 94.

ACKNOWLEDGMENTS

The authors would like to thank the CDISC SHARE team for their work on SHARE 2.0, including Anthony Chow, Mike Hamidi, Sally Cassells, Darcy Wold, Marcelina Hungria, Dorina Bratfalean, Chris Gemma, and Ann White. The authors also thank the Nurocor team for their SHARE 2.0 contributions, including Steve Castellano, Roopa Kandukuri and Barrie Nelson for content loading and PM support, and Aaron Wheeler and Mark Watson for their platform development work.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Sam Hume
CDISC
shume@cdisc.org
https://www.cdisc.org/

Frederik Malfait
Information Architect, Nurocor
frederik.malfait@nurocor.com

http://www.nurocor.com