White Paper

Bhavin Busa, VP and Head of Clinical Data Services and Operations, Vita Data Sciences
Sally Cassells, Senior Director, Data Exchange Standards, CDISC
Bess Leroy, Head, Data Standards, CDISC
Kaja Najumudeen MS, Manager, Data Standards and Automation, Algorics
Mikkel Traun, Principal System Developer, Novo Nordisk A/S
Tianna Umann, Solution Architect, Technology Strategy Team, Microsoft Consulting Services Office of the CTO
# Table of Contents

Summary ........................................................................................................................................... 2

CDISC 360 Test Data Clinical Data Repository ................................................................................. 4

CDISC 360 Proof of Concept Trial Design and Scope ...................................................................... 5

Biomedical Concept Definition ......................................................................................................... 5

Standards Builder Toolkit ................................................................................................................ 8

CDISC 360 Concept Library Proof of Concept Component ............................................................... 13

Study Designer App 360 Proof of Concept Component .................................................................. 14

Study Builder and Sponsor Standards API CDISC 360 Proof of Concept Component ................. 27

Sponsor Study MDR 360 Proof of Concept Component ................................................................. 27

Study Metadata Queries .................................................................................................................. 37

Neo4j SAS Interface 360 Proof of Concept Component ................................................................. 39

SDTM and ADaM Dataset Automation ........................................................................................... 40

TFL Automation 360 Proof of Concept Component ....................................................................... 47

Data Transformation Engine 360 Proof of Concept Component .................................................... 56
Summary

The CDISC Foundational Standards define research data and metadata structures but writing these standards as documents has yielded more text than metadata. Gaps in standards metadata limit automation opportunities. The inherent flexibility provided by the standards supports a broad range of implementations, yet that flexibility allows for inconsistencies that make scaling automation difficult. The lack of a conceptual foundation for the standards further contributes to these inconsistencies. The relationships that would be expressed by these concepts remain largely implicit in the current versions of the standards.

CDISC 360 seeks to implement standards as linked metadata with a conceptual foundation providing the additional semantics needed to support metadata-driven automation across the end-to-end clinical research data lifecycle. This will enable software developers to develop new tools (proprietary and open source) that consume this novel metadata to ease standards’ implementations, while increasing data processing efficiencies.

The aim of the CDISC 360 is to demonstrate the feasibility of standards-based, metadata-driven automation as a start toward realizing the full benefits expected of the CDISC standards: substantially improved efficiency, consistency, and re-usability across the clinical research data lifecycle. These benefits drive the return on investment in the CDISC standards implementations expected by CDISC stakeholders.

This White Paper describes the output of the CDISC 360 Proof of Concept as well as the technical prototypes developed using CDISC standards as linked metadata. The CDISC 360 standards content provides the additional semantics needed to support metadata-driven automation across the end-to-end clinical research data lifecycle.

Proof of Concept extensions to the CDISC standards have been developed by enabling metadata to support automation of end-to-end, clinical-study-data-artifact creation. A minimal amount of background-processing software was developed to demonstrate and confirm the viability of the standards-metadata extensions to drive such automation.

The Proof of Concept centered on three use cases:

- **Use Case #1**: Create end-to-start, standards-based specification. The intent of this use case was to produce a machine-readable, standards-based specification from concept-based standard analysis output definitions in CDISC Library.

- **Use Case #2**: Generate start-to-end, study-specific metadata. To accomplish this, the project used the standards-based specification from the first use case to generate machine-readable, study-specific metadata artifacts.

- **Use Case #3**: Transform data start-to-end. To do so, the project used the machine-readable study-specific metadata from the second use case to process the study data. This demonstrated the ability to execute data transformations given the study-specific metadata.
CDISC invited employees from our membership community to participate in the project. Project teams were organized into the following seven workstreams:

- **Workstream #1**: Create concepts in knowledge graphs.
- **Workstream #2**: Transform concepts in machine-readable form
- **Workstream #3**: Add transformation semantics
- **Workstream #4**: Identify and select standards specification (Use Case 1)
- **Workstream #5**: Configure study specification and create artifacts (Use Case 2)
- **Workstream #6**: Automatically process and transform data (Use Case 3)
- **Workstream #7**: Document FDA analysis requirements in knowledge graphs. Verify analysis requirements, data and metadata traceability

The seven workstream teams started their work concurrently, requiring them to mock up their own set of inputs from previous workstreams. As the project progressed, they would share their outputs with each of the teams to ensure that they were aligned. In the end, while the Proof of Concept was not able to run automatically, it was able to prove the three use cases individually and showed that the alignment between the teams did indeed prove that an end-to-start specification could be generated by selecting the relevant biomedical concepts; that the specification could be used to generate study specific, machine-readable metadata artifacts; and drive the data transformations needed to implement end-to-end standards automation.

Diagram 1 illustrates the workflow within and across the CDISC 360 teams. A component model that shows the various functional elements the teams focused on in delivering the project follows the workflow diagram. Together, these diagrams provide the context needed to interpret how the project achieved a successful Proof of Concept.
Illustrating the application of concept-based standards that support end-to-end automation can only be achieved using realistic test data. Since it was not possible to use data from real trials, the project generated artificial, subject-level test data for an imaginary study within diabetes. Generated test data do not hold any reference to real trials, protocols, subjects, patients or other personal data for any individuals.

The generated test data focused on CDASH, SDTM and ADaM datasets. Data domains for Trial Design datasets (TS, TA, TE and TV) as well as a few core datasets (DM, DS, SV, VS, LB and AE) were used. The test data included the corresponding CDASH data (DM, VS, LB and AE domain) and a set of ADaM datasets covering ADSL, ADVS, ADAE, and ADLB.

**Technologies**

Test data was generated via metadata-driven, SAS programs external to the project; files were transferred into the CDISC 360 Git repository. The data was then imported into the SAS DevTest Lab SAS computing environment in Azure.

Test datasets were represented as:

- SAS Transport files including Define.xml
- csv files
- Native SAS Dataset files (.sas7bdat)
- Other supporting files as PDF and Text
CDISC 360 Proof of Concept Trial Design and Scope

Test data was created based on a mock Phase III study that compared human insulin with Metformin for subjects with Type 2 diabetes. This approach enabled the team to simulate real drugs in the domains EX and TS – with codes for PCLASS and UNII. A mock protocol document was created based on the Common Protocol Template to document the generated test data in more detail. This document is located with the SDTM dataset so a link can be made from the Define.xml.

The trial design comprises a simple, two-arm, parallel group design with two weeks of screening, 26 weeks of treatment and four weeks follow-up – a total of 32 weeks with regularly scheduled visits during the trial. This is typical for Phase III diabetes trials with efficacy and safety endpoints described in the Diabetes Therapeutic Area User Guides as well as MACE.

SDTM Trial Design datasets were created (TA, TE, TV, TS) and study was added to the CDISC 360 Study Library in Neo4j.

Data for 100 subjects are generated. At the moment all subjects pass screening and completes the trial (i.e., no screening failures, withdrawals or lost to follow-up); this can be added later, if applicable.

Biomedical Concept Definition

The CDISC 360 Workstream 1 team defined Biomedical Concepts as high-level building blocks of clinical research information that encapsulate lower-level implementation details, such as variables and terminologies. A Biomedical Concept is a unit of knowledge created by a unique combination of characteristics that specifies an observation concept in a clinical study, but it does not specify how to capture the data or how to group observations together. Biomedical Concepts exist independently of any given standards implementation, such as SDTMIG v3.2 or CDASHIG v2.0.

The Workstream 1 team was charged with finding a way to represent Biomedical Concepts as linked metadata within the confines of CDISC standards.

An early decision was to represent Biomedical Concepts as concept maps using CMapTools developed by the Institute for Human-Machine Cognition. Multiple styles and formats for producing these maps were experimented with before ultimately deciding on using the ISO 11179 standard for representation of metadata; a logical choice since ISO 11179 is also the basis on which CDISC Library is built. To accommodate all concept attributes required for representing Biomedical Concepts in a manner useful for downstream workstreams, components outside of ISO 11179 were added.

In the course of the Proof of Concept project, the team made a decision to produce higher-level, template maps with a focus on reusability across related concepts (e.g., a high-level, vital signs map that could be instantiated for specific vital signs measurement). Instantiation of these concept maps involved the creation of metadata files to bind the concepts to the specific, required Controlled Terminology in CDISC Library. The maps produced for CDISC 360 are designed to be human and machine-readable.

Finally, files were produced to illustrate the mapping of collected concepts from CDASH to SDTM.
Following this approach, we produced concept maps and associated metadata files for the following:

- **Vital signs**
  - Systolic blood pressure
  - Diastolic blood pressure
  - Temperature
- **Adverse events**
- **Insulin administration**
- **Labs**
  - HbA1C
  - Hemoglobin
  - Total cholesterol
  - LDL cholesterol
- **Subject demographics**
- **Disposition events and milestones**
- **Trial arms**
- **Trial elements**
- **Trial summary parameters**
- **Trial visits**

The final maps and files were handed off to the CDISC 360 Workstream 2 team for further processing.

**Sub-components**

- Biomedical Concept definition
- Concept maps
- Metadata files
  - Binding files
  - CDASH-SDTM mapping files

**Technologies**

- Concept maps were constructed using CMapTools from IHMC.
- Metadata files were created using Microsoft Excel.

**Scope of Functionalities**

Concept maps are visual representations of concepts expressed as linked metadata, including:

- Data elements and their relationships
- Controlled Terminology and associated codelists
- Derivations, where applicable

Most concept maps produced for CDISC 360 represent SDTM-observation concepts (e.g., HbA1c, systolic blood pressure, insulin administration, etc). Some are template maps designed to be re-used for any of a number of related concepts (e.g., labs or vital signs measurements). A small number of analysis concept maps were produced, but more robust testing of these maps is required.
Binding files are tabular representations of the information contained in the concept maps intended as content for CDISC Library. They show the bindings of variable values to the appropriate Controlled Terminology.

**Project Status**

The Workstream 1 team developed concepts as described above for the listed concepts. Work has begun on analysis concept maps; further testing and development is required.

**Sources/Reference documents**

CmapTools: [https://cmap.ihmc.us/cmaptools/](https://cmap.ihmc.us/cmaptools/)

**Illustrations**

Diagram 2: Key for the Development of Concept Maps in Workstream 1.

Diagram 3: Example of an Abbreviated Template Concept Map (Not All Variables Shown) for Vital Signs Measurements
Limitations and Assumptions

ISO 11179 was not a perfect fit for our needs in representing Biomedical Concepts and required augmentation. Implications for interfacing with CDISC Library are still being explored and workarounds may be required.

Suggested Next Steps

Going forward, we intend to expand concept-based development to additional concepts with a focus on the most commonly collected concepts across a survey of study sponsors. Working with the newly formed CDISC Analysis Results Metadata standards team, we aim to further refine the process of using concept maps in standards development in a new use case for representing the dataflow into analysis outputs like tables, listings and figures. Work has begun on analysis concept maps; further testing and development is required.

Standards Builder Toolkit

Workstream 2 used the Standards Builder Toolkit, which is a set of Python tools used to transform the Biomedical Concepts and mappings developed by Workstream 1 into a virtual Metadata Repository Sandbox. The Sandbox can serve as a source for creating Define-XML and ODM exports for the study design, build and execute teams.

Diagram 4: Standard Builder Toolkit Schematic

Sub-components

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CXL_Loader</td>
<td>Python tool</td>
<td>Converts CMAP CXL export to GraphML</td>
</tr>
<tr>
<td>Cmap2JSON</td>
<td>Python tool</td>
<td>Uses GraphML to drive CDISC Library queries, generate Biomedical Concept Metadata and SDTM Clib Metadata output by domain. Domain specific output files are used by WriteDefineXML to create a Define-XML for all available Biomedical Concepts.</td>
</tr>
<tr>
<td>Mappings2JSON</td>
<td>Python tool</td>
<td>Uses Mappings Spreadsheet to drive CDISC Library queries Domain specific output files are used by Write DefineXML to create Define-XML for all available Biomedical Concepts. JSON output is used by Cmap2JSON.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WriteDefineXML</td>
<td>Python tool</td>
<td>Generates Define-XML for a preconfigured list of domains. For each domain, the Biomedical Concept JSON file and mappings files generated.</td>
</tr>
<tr>
<td>C360WriteODMXML</td>
<td>Python tool</td>
<td>Generates ODM forms for Biomedical Concepts.</td>
</tr>
<tr>
<td>c360-define-schemas</td>
<td>XSD files</td>
<td>Schema extensions for Define-XML.</td>
</tr>
<tr>
<td>c360-odm-schemas</td>
<td>XSD files</td>
<td>Schema extensions for ODM.</td>
</tr>
<tr>
<td>define2-1-forC360</td>
<td>XSL file</td>
<td>Adaptation of ‘standard’ Define-XML stylesheet updated to support schema extensions.</td>
</tr>
<tr>
<td>C360 ODM stylesheet</td>
<td>XSL file</td>
<td>Updated version of legacy ODM to CRF stylesheet.</td>
</tr>
<tr>
<td>Biomedical Concept Metadata</td>
<td>JSON Files</td>
<td>For each domain in the C360 and Mace+ scope, xx_bcConcept. json file.</td>
</tr>
<tr>
<td>Define-XML File Archive</td>
<td>Wiki page</td>
<td>See the Define-XML folder in the C360WS2Tools GitHub repository referenced below.</td>
</tr>
<tr>
<td>ODM File Archive</td>
<td>Wiki page</td>
<td>See the ODM-XML folder in the C360WS2Tools GitHub repository referenced below.</td>
</tr>
<tr>
<td>CMAP File Archive</td>
<td>CXL Files</td>
<td>See the data/CMAP-XML folder in the C360WS2Tools GitHub repository referenced below.</td>
</tr>
<tr>
<td>JSON File Archive</td>
<td>JSON Files</td>
<td>See the data/bcJsonFiles folder in the C360WS2Tools GitHub Repository referenced below.</td>
</tr>
</tbody>
</table>

**Table 1: Standards Builder Toolkit Sub-components**

**Github:** [https://github.com/scassells/C360WS2Tools](https://github.com/scassells/C360WS2Tools)

**Biomedical Concepts SDTM Bindings Metadata**

**Standard and Version Metadata**

<table>
<thead>
<tr>
<th>Index</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent</td>
<td>string:$domain-sdtmig</td>
<td></td>
</tr>
<tr>
<td>prodVers</td>
<td>string:sdtmig-$vers</td>
<td></td>
</tr>
<tr>
<td>$domain_bcConcepts</td>
<td>List</td>
<td></td>
</tr>
</tbody>
</table>

**Domain Level Biomedical Concepts**

<table>
<thead>
<tr>
<th>$domain_bcConcepts</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordinal</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>bcID</td>
<td>string</td>
<td>&quot;BC&quot;$domain{bcName}</td>
</tr>
<tr>
<td>$domain_bcConcepts</td>
<td>Content</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>bcName</td>
<td>string</td>
<td>Remove spaces in bcTopicVar</td>
</tr>
<tr>
<td>bcTopicVar</td>
<td>string</td>
<td>Label of Observational Concept in Cmap</td>
</tr>
<tr>
<td>bcCond</td>
<td>string</td>
<td>Specified in short comment from Observational Concept in Cmap</td>
</tr>
<tr>
<td>_links</td>
<td>sdtm-topic:{&quot;href&quot;:URL,&quot;title&quot;:string,&quot;type&quot;:string}</td>
<td></td>
</tr>
<tr>
<td>bcVarList</td>
<td>List</td>
<td>List of Biomedical Concept SDTM variable bindings</td>
</tr>
</tbody>
</table>

**Biomedical Concept SDTM Variable Bindings**

<table>
<thead>
<tr>
<th>index</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>string: SDTM Variable Name</td>
<td>Specified in short comment in first DEC node in the cmap.</td>
</tr>
<tr>
<td>CLibRef</td>
<td>&quot;self&quot;:{&quot;href&quot;:URL,&quot;title&quot;:string,&quot;type&quot;:string}</td>
<td></td>
</tr>
<tr>
<td>cdType</td>
<td>&quot;integer&quot; &quot;text&quot; &quot;ISO 8601&quot;</td>
<td></td>
</tr>
<tr>
<td>cdVal</td>
<td>subsetSpecification:{&quot;subset&quot;:string} valueSpecification:{&quot;value&quot;:string}</td>
<td>The subset string is label of a CD node in the CMAP with semicolon(;:) separator, linefeeds removed. The value string is the label of a CD node in the CMAP with a single value. It may or may not include a C-Code.</td>
</tr>
<tr>
<td>&quot;origin&quot;</td>
<td>MappingType: &quot;Predecessor&quot;, &quot;Assignment&quot;, &quot;Computation&quot; Content dependent on MappingType</td>
<td></td>
</tr>
</tbody>
</table>

**CDISC Library References for Variable and Codelist**

<table>
<thead>
<tr>
<th>ClibRef</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>&quot;href&quot;: URL, &quot;title&quot;:string,&quot;type&quot;:string</td>
<td>get Endpoint</td>
</tr>
<tr>
<td>codelist</td>
<td>&quot;href&quot;: URL, &quot;title&quot;:string,&quot;type&quot;:string</td>
<td>get Endpoint</td>
</tr>
</tbody>
</table>

**Biomedical Concept Mapping Information**

<table>
<thead>
<tr>
<th>Predecessor Origin</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MappingType</td>
<td>Predecessor</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>Description</td>
<td>string</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>SourceVar</td>
<td>string</td>
<td>String is formatted as CDASH-$domain:$sourceVarName</td>
</tr>
</tbody>
</table>
Mapping Method Metadata

<table>
<thead>
<tr>
<th>Method Origin</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MappingType</td>
<td>Computation</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>InputVariables</td>
<td>List{&quot;Standard&quot;:$standardName,&quot;Domain&quot;:$domain,&quot;VarName&quot;:string,&quot;Value&quot;:string}</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>Preferred</td>
<td>“Yes”</td>
<td>from mappings spreadsheet</td>
</tr>
</tbody>
</table>

CMAP CXL Archive

Diagram 5: CXL exports can be imported into the Cmaps tool as shown in the screenshot.
### MACE+ Analysis Concept ADaM Bindings Metadata

<table>
<thead>
<tr>
<th>Index</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent</td>
<td>string: $domain-sdtmig</td>
<td></td>
</tr>
<tr>
<td>prodVers</td>
<td>string: admig-$vers</td>
<td></td>
</tr>
<tr>
<td>$domain_bcConcepts</td>
<td>List</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$domain_bcConcepts</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orinal</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>acID</td>
<td>string</td>
<td>&quot;$BC&quot;$domain(bcName)</td>
</tr>
<tr>
<td>acName</td>
<td>string</td>
<td>remove spaces in bcTopicVar</td>
</tr>
<tr>
<td>acTopicVar</td>
<td>string</td>
<td>Label of Observational Concept in Cmap</td>
</tr>
<tr>
<td>acCond</td>
<td>string</td>
<td>Specified in short comment from Observational Concept in Cmap</td>
</tr>
<tr>
<td>_links</td>
<td>AVAL, PAaram</td>
<td></td>
</tr>
<tr>
<td>acVarList</td>
<td>List</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>string: ADaMVariable Name</td>
<td>Specified in short comment in first DEC node in the cmap</td>
</tr>
<tr>
<td>CLibRef</td>
<td>&quot;self&quot;:{&quot;href&quot;:URL,&quot;title&quot;:string,&quot;type&quot;:string}</td>
<td></td>
</tr>
<tr>
<td>cdType</td>
<td>&quot;integer&quot; &quot;text&quot; &quot;ISO 8601&quot;</td>
<td></td>
</tr>
<tr>
<td>cdVal</td>
<td>subsetSpecification:{&quot;subset&quot;:string} valueSpecification:{&quot;value&quot;:string}</td>
<td>The subset string is label of a CD node in the CMAP with semicolon(;) separator, line feeds removed. The value string is the label of a CD node in the CMAP with a single value. It may or may not include a C-Code.</td>
</tr>
<tr>
<td>&quot;origin&quot;</td>
<td>MappingType: &quot;Predecessor&quot;, &quot;Assignment&quot;, &quot;Computation&quot; content dependent on MappingType</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLibRef</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>&quot;href&quot;: URL, &quot;title&quot;:string,&quot;type&quot;:string</td>
<td>get Endpoint</td>
</tr>
<tr>
<td>codelist</td>
<td>&quot;href&quot;: URL, &quot;title&quot;:string,&quot;type&quot;:string</td>
<td>get Endpoint</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Predecessor Origin</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MappingType</td>
<td>Predecessor</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>Description</td>
<td>string</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>SourceVar</td>
<td>string</td>
<td>String is formatted as CDASH:$domain:$sourceVarName</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Origin</th>
<th>Content</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MappingType</td>
<td>Computation</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>InputVariables</td>
<td>List(“Standard”:$standardName.”Domain”:$domain,”VarName”:string,”Value”:string)</td>
<td>from mappings spreadsheet</td>
</tr>
<tr>
<td>Preferred</td>
<td>“Yes”</td>
<td>from mappings spreadsheet</td>
</tr>
</tbody>
</table>

**CDISC 360 Concept Library Proof of Concept Component**

The CDISC 360 Concept Library Proof of Concept Component is intended to serve as the sandbox library holding the new concept-based library. The files were generated to represent the concept-based standards but were not imported directly.

Additional information needed for the Proof of Concept standards was represented in supplemental metadata loaded into the Sponsor Study MDR.

**Sub-components**

**XML files**
Represent Biomedical Concept.

**Supplemental Metadata**
Generated and imported manually into Sponsor Study MDR.

**Technologies**

XML and CSV files.

**Scope of Functionalities**
For the CDISC 360 Proof of Concept, additional metadata was generated for Biomedical Concepts not currently available in the CDISC Library.
### Component Implementation

<table>
<thead>
<tr>
<th>Identifiers</th>
<th>Biomedical Concept ID, Name and DEC bindings</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcCond</td>
<td>Define-XML WhereClause</td>
</tr>
<tr>
<td>bcVarList</td>
<td>Define-XML ItemGroup</td>
</tr>
<tr>
<td>DEC</td>
<td>SDTM Variable names – bound to domain definition in SDTMIG Version identified in Biomedical Concept parent and prodVers</td>
</tr>
<tr>
<td>CLibRef</td>
<td>URIs for CDISC Library variable and codelist endpoints.</td>
</tr>
<tr>
<td>cdVal</td>
<td>Concept Domain - subsets</td>
</tr>
<tr>
<td>origin</td>
<td>CDASH-SDTM mappings and derivations</td>
</tr>
</tbody>
</table>

### Deployment of Component

Standards Builder Toolkit.

### Project Status

Only the minimum needed for the 360 Proof of Concept have been made.

### Sources/Reference Documents

[WS4 Supplemental Metadata](#)

### Study Designer App 360 Proof of Concept Component

SDTM Trial Design datasets were created (TA, TE, TV, TS) and the study was added to the CDISC 360 Study Library in Neo4j.

Workstream 4’s Proof of Concept component described here is a prototype for a CDISC 360 Study Designer App. The Study Designer App starts with a Protocol Outline and delivers a complete Study Specification, including structured protocol elements. This was accomplished via an API-based connection to a Sponsor Study Metadata Repository and can be done via a file-based solution.

The first step was to create basic definitions for the study, covering identifiers, study title and the selected data standard versions. Note: We envision that you can decide at any time to up or down version any data standards.

The linked graph model supported identifying any consistency issues – a benefit of applying a linked graph data model.

Next, we created the specification of the study design, including the planned interventions. Then followed the selection, configuration and scheduling of the Biomedical Concepts in the form of objectives, endpoints, activities and assessments.

Finally, the Build Process for generating various study-specification artifacts was created with the ability to browse and export the study metadata in various representations from the List menu.

The key focus was the use of the new 360 concept-based standards to drive the study specification, which is tool and system agnostic. We illustrated this using the new standards to
create a study specification via a simple, web-based Study Designer App connected to a Neo4j based Study MDR.

**Future State - with Concept-based Standards: Study Specification in a CDISC 360 Study Builder App**

Sub-components

- Front-end application based on a Python-Django framework.
- Back-end application based on a Python-Django-API framework managing the connection to the Neo4j database.
- Various Python packages are also used in the Proof of Concept.

**Technologies**

- Both front-end and back-end parts run as a Python-Django application on an Azure app service.
- The connection to the Neo4j database is fully handled by the Python drivers available at: [https://neo4j.com/docs/api/python-driver/](https://neo4j.com/docs/api/python-driver/)
- The network data visualization intended to follow the user in the study build process and data exploration is handled with vis.js network package available at: [https://visjs.org/](https://visjs.org/)
- The necessary data manipulation between the cypher-query results and the front-end application are handled with Pandas and Numpy.
- The editable tables in the design page are powered by the GIJGO grid package: [https://gijgo.com/grid/](https://gijgo.com/grid/)
- The main CSS framework used across the application is Bootstrap [https://getbootstrap.com/](https://getbootstrap.com/); however, only key components necessary for project scope are fully responsive.

**Scope of Functionalities**

The Study Designer App illustrates the Study Define, Design, Select and Configure Biomedical Concepts and then builds a Study Configuration via an MDR solution. We prototyped one way, but there are many ways of applying the CDISC 360 concept-based standards as long as the solution is tool and system agnostic. A file-based solution design would also work.
Key Features in the Study Designer App

<table>
<thead>
<tr>
<th>Library</th>
<th>Import definitions from external libraries. Manage sponsor defined selections and definitions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define</td>
<td>Identifiers and a general set of trial summary parameters for the study.</td>
</tr>
<tr>
<td>Design</td>
<td>Study design parameters as well as defining study arms, elements, epochs and visits.</td>
</tr>
<tr>
<td>Select</td>
<td>Search and select concept-based standards and define schedule of activities and assessments.</td>
</tr>
<tr>
<td>Build</td>
<td>Generate study specification artefacts that support automation of study setup and execution.</td>
</tr>
<tr>
<td>List</td>
<td>Extract study metadata in tabular format for downstream usage. From within the App with export to multiple format as well as direct from SAS.</td>
</tr>
</tbody>
</table>

Goal for Workstream 4: Create a linked graph model for a Sponsor Study MDR.

We did not create a 360 Sandbox Library, and since all teams had to start in parallel, the file-based, Biomedical Concept representation was not available to us at the start. Therefore, we created supplemental metadata as simple csv files, which we used to load the enhanced metadata currently not available in the CDISC Library. We aligned content and structure on an ongoing basis. The process flow for the Study Designer App is illustrated in the lower half of Diagram 7.

Diagram 7: Study Designer App Process Flowchart

Component Deployment

A release pipeline from the Azure Git repository published the application on the Azure App service.
Examples

On the **Library** menu, the user:

- Creates additional templates for Objectives and Endpoints.
- Creates instantiations of imported or sponsor defined templates.
- Includes references to dependent parameters from instances of Objective and Endpoints.

On the **Define** menu, the user:

- Enters the basic description (e.g., the study phase, title, registry identifiers) of the trial.
- Enters study therapeutic area and CDISC TAUGs used.
- Enters version of CDISC Controlled Terminology.
- Enters version of Data Exchange Standards.

On the **Design** Menu, the user:

- Makes a basic selection of trial-design-related summary parameters (e.g., Intervention Type, Intervention Model etc.).
- Defines the Trial Arms, Epochs, Elements and the Design matrix.
- Defines the Visit Schedule.
• Define the Planned Interventions.
On the **Select** menu, the user:

- Selects the concept-based standards from the libraries to be used in the study. Standards can be based on templates instantiated in the local library.
- Selects Objectives and Endpoints.
- Selects Activities and Assessments.
- Selects Schedule of Activities and Assessments.
On the **List** Menu the user can:

- Browse all study metadata in tabular form.
- Export metadata into various file formats.
- Create a SAS-based interface to the Study Metadata Library enabling extract of study metadata into SAS datasets, including CDASH2SDTM and SDTM2ADaM Bindings.)
Interface View of Trial Summary and Datasets

Technologies

Test data was generated by metadata-driven, SAS programs external to the project; files were transferred into the Git repository. The data was then imported into the SAS DevTest Lab SAS computing environment in Azure.

Test datasets were represented as:

- SAS Transport files, including Define.xml
- csv files
- Native SAS Dataset files (.sas7bdat)
- Other supporting files as PDF and Text

Description of CDASH Generation

In the 360 Proof of Concept, we reverse-engineered the generated SDTM data to create CDASH data. While not optimal, this was the simplest and most pragmatic approach considering the expedited timeframe and lack of a robust test data sample.

Description of SDTM Generation

DM Data

All subjects had a random date for screening from 01-JAN-2019 and 60 days forward; all other dates were offset from generated data of screening according to the planned trial time (currently hardcoded in the program and not driven by metadata).

Subjects were randomly allocated to each trial arm and sex; age was random from 18 to 64 years. Other qualifiers can be added later, if applicable.

DS Data

All subjects had the following disposition events and protocol milestones as completers, according to their planned trial time offset from the generated screening visit:
### SV Data
All subjects attended all planned visits at the planned visit time according to their reference start date in the trial.

### VS Data
All subjects had all the planned Vital Signs assessments at the planned visits at the planned time. Data for Vital Signs are generated in the CDASH Denormalized/Horizontal implementation option and a SDTM.VS dataset is provided with a suggested mapping.

SDTM.VS is defined with the following VSCAT, VSTESTCD, VTEST:

<table>
<thead>
<tr>
<th>VSCAT</th>
<th>VSTESTCD</th>
<th>VTEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>VITAL SIGNS</td>
<td>SYSBP</td>
<td>Systolic Blood Pressure</td>
</tr>
<tr>
<td>VITAL SIGNS</td>
<td>DIABP</td>
<td>Diastolic Blood Pressure</td>
</tr>
<tr>
<td>VITAL SIGNS</td>
<td>PULSE</td>
<td>Pulse Rate</td>
</tr>
<tr>
<td>VITAL SIGNS</td>
<td>TEMP</td>
<td>Temperature</td>
</tr>
<tr>
<td>BODY MEASUREMENT</td>
<td>HEIGHT</td>
<td>Height</td>
</tr>
<tr>
<td>BODY MEASUREMENT</td>
<td>WEIGHT</td>
<td>Weight</td>
</tr>
</tbody>
</table>

### AE Data
20 typical AE terms within MACE+ have been created, and each is assigned to one subject (005, 010, 015, etc.). Some result in hospitalization or are fatal; details in the generated data can be changed to be more meaningful, complete or to support scenarios in TLFs. The AE terms within MACE+ were flagged in the ADaM dataset ADE.

Additionally, 1 to 20 non-MACE AEs of common terms were randomly assigned to each subject from a list of typical AE terms within diabetes. This approach provided approximately 1500+ AE records, the actual number and values change for each regeneration of test data.

### LB Data
Currently the lab data holds the following tests:

<table>
<thead>
<tr>
<th>Lbtestcd</th>
<th>Lbtest</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALB</td>
<td>Albumin</td>
</tr>
</tbody>
</table>
### Description of ADaM Generation

The following ADaM datasets were generated based on the SDTM test data, which were intended to be used within the CDISC 360 project.

- **ADSL**: Subject Level Analysis
- **ADLB**: Lab Analysis
- **ADVVS**: Vital Signs Analysis
- **ADAE**: Adverse Events Analysis dataset
- **ADTTE**: Data for the Time to Event Analyses

The following information describes how the Analysis datasets are generated:

#### ADSL: Subject Level Analysis

ADSL contains one record per subject (USUBJID, the unique identifier); variables, such as subject-level population flags; planned and actual treatment variables, demographic information, subgrouping variables, stratification factors, and important dates.

The input SDTM test datasets (i.e., SDTM.DM and SDTM.DS) have been used to generate ADSL.

The main population flags derived in ADSL are FASFL and SAFFL. TRT01A and TRT01P are the treatment variables, which merged with other BDS and OCCDS datasets.

Additionally, by using disposition SDTM dataset, the following variables are derived:

- **EOTSTT**: End of Treatment Status
- **EOSTT**: End of Study Status

Moreover, the key dates and duration variables are derived and available in ADSL.

- **TRTSDT**: Date of First Exposure to Treatment
- **TRTEDT**: Date of Last Exposure to Treatment
- **EOTDT**: End of treatment date

#### Laboratory Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT</td>
<td>Alanine Aminotransferase</td>
</tr>
<tr>
<td>CHOL</td>
<td>Cholesterol</td>
</tr>
<tr>
<td>CREAT</td>
<td>Creatinine</td>
</tr>
<tr>
<td>HBA1CHGB</td>
<td>Hemoglobin A1C/Hemoglobin</td>
</tr>
<tr>
<td>HCT</td>
<td>Hematocrit</td>
</tr>
<tr>
<td>HDL</td>
<td>HDL Cholesterol</td>
</tr>
<tr>
<td>HGB</td>
<td>Hemoglobin</td>
</tr>
<tr>
<td>LDL</td>
<td>LDL Cholesterol</td>
</tr>
<tr>
<td>PLAT</td>
<td>Platelets</td>
</tr>
</tbody>
</table>
\begin{itemize}
  \item **EOSDT**: End of study date
  \item **TRTDURD**: Total Treatment Duration (Days)
  \item **TRTDURY**: Total Treatment Duration (Years)
  \item **INTRDURD**: In Trial Observation Time (Days)
  \item **INTRDURY**: In Trial Observation Time (Years)
\end{itemize}

Along with the above variables, there are demographic variables (e.g., AGE, SEX, RACE, ETHNIC, COUNTRY and AGEGR1).

AGEGR1 has been created by stratifying AGE variable into the following three groups:

1) 15\(\leq\) to \(<30\) years
2) 30\(\leq\) to \(<45\) years
3) \(\geq45\) years

**ADLB: Lab Analysis (BDS)**

The ADLB dataset is based on the SDTM.LB and contains data for laboratory assessments for BIOCHEMISTRY, GLUCOSE METABOLISM, HAEMATOLOGY and LIPIDS.

ADLB is a BDS dataset that contains one or more records per subject, per analysis parameter, per analysis timepoint. Per the Basic Data Structure definition, analysis timepoint represents Analysis Visit (AVISIT).

For all the parameters, the data collected at week 0 is considered baseline value; ABLFL has been derived by this baseline value.

In addition, other variables like CHG (Change from Baseline), PCHG (Percentage Change from Baseline), BASE (Baseline Value), R2BASE (Ratio to Baseline) and ADY (Analysis Relative Day) are derived for analysis purpose.

As stated in the protocol, the secondary endpoint is “Proportion of Subject with Hba1C < 7\% (Count). Timeframe: after 26 weeks.” As a result, we derived the CRIT1 (Analysis Criterion 1) variable, which indicates whether Hba1C value is \(<7\%\) or not and the corresponding flag variable (i.e., CRIT1FL (Criterion 1 Evaluation Result Flag)), which have the values Y or N.

Along with the above-mentioned analysis variables, the core variables are merged with ADLB from ADSL dataset.

**ADVS: Vital Signs Analysis (BDS)**

The ADVS dataset is based on the SDTM.VS and contains data of assessments for Body Measurement (Height and Weight) and Vital Signs (Diastolic Blood Pressure, Pulse Rate, Systolic Blood Pressure and Temperature).

ADVS is a BDS dataset that contains one or more records per subject, per analysis parameter, per analysis timepoint. Per the Basic Data Structure definition, analysis timepoint represents Analysis Visit (AVISIT).

For all parameters, the data collected at week 0 is considered baseline value; ABLFL has been derived by this baseline value.
In addition, other variables like CHG (Change from Baseline), PCHG (Percentage Change from Baseline), BASE (Baseline Value), R2BASE (Ratio to Baseline) and ADY (Analysis Relative Day) are derived for analysis purpose.

Along with the above-mentioned analysis variables, the core variables are merged with ADVS from ADSL dataset.

**ADAE: Adverse Events Analysis Dataset**

The ADAE dataset contains all collected and reported events in SDTM.AE, meeting the definition of an adverse event (AE). All events from the first trial-related activity, after the subject has signed the informed consent until the end of the post-treatment, follow-up period, are included.

The treatment emergent duration is defined as the duration for which the subject is on treatment, including an ascertain window of 7 days (i.e., TRTEMFL = “Y” when TRTSDT <= ASTDT <= TRTEDT+7)

As stated in the CDISC 360 protocol, the primary objective is “Time to first occurrence of MACE+, a composite endpoint consisting of: CV death, nonfatal MI, nonfatal stroke, or hospitalization for unstable angina.”

The external data have been used here. For example, AE_MACE_1, which includes the list of AETERMS, satisfies MACE criteria and derives the flag variable MACEPFL (MACE Plus Flag). This variable was used in ADTTE dataset for performing time-to-event analysis.

Some of the core AE variables listed in the following table are available in ADAE.

<table>
<thead>
<tr>
<th>Column</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>AETERM</td>
<td>Reported Term for the Adverse Event</td>
</tr>
<tr>
<td>AEDECOD</td>
<td>Dictionary-Derived Term</td>
</tr>
<tr>
<td>AEBODSYS</td>
<td>Body System or Organ Class</td>
</tr>
<tr>
<td>AEBDSYCD</td>
<td>Body System or Organ Class Code</td>
</tr>
<tr>
<td>AELLT</td>
<td>Lowest Level Term</td>
</tr>
<tr>
<td>AELLTCD</td>
<td>Lowest Level Term Code</td>
</tr>
<tr>
<td>AEPTCD</td>
<td>Preferred Term Code</td>
</tr>
<tr>
<td>AEHLT</td>
<td>High Level Term</td>
</tr>
<tr>
<td>AEHLTCD</td>
<td>High Level Term Code</td>
</tr>
<tr>
<td>AEHLGT</td>
<td>High Level Group Term</td>
</tr>
<tr>
<td>AEHLGTCD</td>
<td>High Level Group Term Code</td>
</tr>
<tr>
<td>AESOC</td>
<td>Primary System Organ Class</td>
</tr>
<tr>
<td>AESOCCCD</td>
<td>Primary System Organ Class Code</td>
</tr>
<tr>
<td>AESEV</td>
<td>Severity/Intensity</td>
</tr>
<tr>
<td>AEREL</td>
<td>Relationship to trial product</td>
</tr>
<tr>
<td>AESER</td>
<td>Serious Event</td>
</tr>
<tr>
<td>AEOOUT</td>
<td>Outcome of Adverse Event</td>
</tr>
<tr>
<td>AESHOSP</td>
<td>Requires or Prolongs Hospitalization</td>
</tr>
</tbody>
</table>
Along with the above-mentioned analysis variables, the core variables are merged with ADAE from ADSL dataset.

**ADTTE: Data for the Time-to-Event Analyses**

The ADTTE dataset is based on the ADAM.ADAE and contains data for the following parameters:

- Time to first occurrence of MACE+(days)
- Time from randomization to death (days)

The structure of ADTTE is a BDS dataset that contains one record per subject, per analysis parameter as well as the parameters mentioned above.

As stated above, to create ADTTE, ADAE is the input dataset by filtering MACEPFL = “Y” for the PARAMCD = “MACE+” and AEOUT = “FATAL” for the PARAMCD = DEATH”.

CNSR variable is created based on the event occurrence and the values are 0 (when the event occurs) and 1 (completed the study without having the event). The corresponding AVAL variable (timing variable in days) is derived.

EVNTDESC is the censor description variable derived for both parameters.

The following table provides additional details on analysis variables and the possible values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>PARAMCD</th>
<th>EVNTDESC</th>
<th>CNSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to first occurrence of MACE+(days)</td>
<td>MACE+</td>
<td>FIRST MACE+</td>
<td>0</td>
</tr>
<tr>
<td>Time to first occurrence of MACE+(days)</td>
<td>MACE+</td>
<td>COMPLETED STUDY</td>
<td>1</td>
</tr>
<tr>
<td>Time from randomization to death (days)</td>
<td>DEATH</td>
<td>DEATH</td>
<td>0</td>
</tr>
<tr>
<td>Time from randomization to death (days)</td>
<td>DEATH</td>
<td>COMPLETED STUDY</td>
<td>1</td>
</tr>
</tbody>
</table>

Along with the above-mentioned analysis variables, the core variables are merged with ADTTE from ADSL dataset.

**Project Status**

Selected parts of the application are fully functioning and connected live to the Neo4j database. Some menu items, tagged with ⚠️ have been added to illustrate the intended scope.

**Sources/Reference documents**

- WS4 - Study Designer App
- Git repository StudyDesignerApp

**Limitations and Assumptions**

A mock dataset for 100 subjects based on a Phase V sample protocol, only represents subjects who passed the initial screening and completed the trial (i.e., no screening failures, withdrawals or lost to follow-up).
**Suggested Next Steps**
Use a Phase III sample dataset to enable a larger scope that includes screening failures, withdrawals or lost to follow-up.

**Study Builder and Sponsor Standards API CDISC 360 Proof of Concept Component**

**API Interface for the Sponsor MDR**
Workstream 4’s API Interface manages sponsor-defined extensions to CDISC standards as well as Study Definitions.

The goal of this component is to conduct all interactions with the Sponsor MDR via a standardized API, enabling the use of multiple tools from different vendors. The API service layer manages access control, versioning and audit trail.

**Technologies**
- Python-Django API framework using Cypher statements
- API is documented by Open API (Swagger)

**Scope of Functionalities**
For the CDISC 360 Proof of Concept, only a few API endpoints were defined and tested to validate how this could be done and what it requires in the technical application design.

We started with a limited scope design as we learned how best to work with the API framework and iterated it to a full scope pilot implementation.

**Suggested Next Steps**
A pilot implementation of the API-driven design should demonstrate a redesigned full scope implementation and be available as part of the CDISC Open Source Alliance (COSA) scheduled for deployment in late 2021.

**Sponsor Study MDR 360 Proof of Concept Component**
Linked graph database that holds connected metadata for:
- CDISC Standards
- Sponsor-defined Extensions
- Study-specific Metadata for the Study Definition
Technologies

Neo4j database. A number of Neo4j Cypher scripts create graph model constraint definitions and load test metadata data into the Neo4j database (see: https://neo4j.com/product/).

Scope of Functionalities

The scope and purpose of the Sponsor Study MDR is to support the clinical study process from planning, study design, study specification, and study set up to drive downstream automation. Such a solution is deeply related to the data domain of clinical studies. A domain-driven design is applicable for this IT solution and fits well with an API-based architecture, where the API endpoints are closely related to the data domain.

This section describes the data model for the Sponsor Study MDR component at different levels of abstraction, each with a dedicated focus. The initial purpose of the data model description is to support the design and implementation process during the system development process of a Sponsor Study MDR system. Equally important, the data model description needs to support the usage and maintenance of a Sponsor Study MDR after go live. The data model description is an important outcome of the CDISC 360 Proof of Concept and a crucial part of the coming development of a CDISC 360-based Study MDR solution.

The first step is to define the boundaries of the data domain that the system should cover by identifying its high-level data domains and subject areas. This step is important for the scoping and identifying dependencies needed for the system components as well as for project planning in the development phase and the maintenance phase.

The next step is to design the logical data model needed to solve the tasks and deliverables for the system and identify the data entities, attributes and their relationships.

Final steps: 1) Design how these data elements are to be exchanged via the system interfaces (APIs) between the different system components and the domain data model. 2) Determine how the data model is to be implemented in the actual data storage, the physical data model.

The following table provides an overview of the different types of data models, their definition and purpose.

<table>
<thead>
<tr>
<th>Data Model Type</th>
<th>Definition</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Data Model</td>
<td>A high-level description of informational needs underlying the design of a database.</td>
<td>Define the scope for the data domain and subject areas the data model should cover.</td>
</tr>
<tr>
<td>Logical Data Model</td>
<td>A high-level description of informational needs underlying the design of a database.</td>
<td>Define the scope for the data domain and subject areas the data model should cover.</td>
</tr>
<tr>
<td>Domain Data Model</td>
<td>A domain model is a representation of the data, independent of the way the data is stored in the database.</td>
<td>Define the way data can be exchanged between systems (e.g., by an API based interface). Can be in various exchange formats like JSON, XML, CSV, etc.</td>
</tr>
<tr>
<td>Physical Data Model</td>
<td>A representation of a data design as implemented in a database management system.</td>
<td>The technical specification for the design of the data base implementation.</td>
</tr>
</tbody>
</table>
Conceptual Data Model – Domain Areas

The purpose of the domain areas in the conceptual data model is to define the overall data scope for the Sponsor Study MDR system component. The system includes the following data model domain areas in the following sections:

- System Configuration
- Industry Standards
- Sponsor Standards
- Study Definitions
- Administrative Definitions

Conceptual Data Model – Subject Areas

The purpose of the subject areas in the conceptual data model is to define the main data domains and their relationships in order to define a more detailed scope for identifying dependencies. This insight can be used for planning the implementation order for the different components as well as dependencies when they later are maintained. The conceptual data model diagram is layered by each data domain area and shows each subject area with their main relationships.
**System Configuration**
The System Configuration domain area holds the various entities (e.g., connected external and defined internal libraries) that form the system configuration.

- Library Definitions hold system definitions such as external CDISC Library and internal Sponsor Library.

**Industry Standards**
The Industry Standards domain area holds the imported industry standards, initially only from CDISC Library (i.e., Controlled Terminologies and Foundational Standards). Additional standards (LOINC, SNOMED and MedDRA dictionaries) will be added later.

- **Controlled Terminologies**
  - The Controlled Terminologies subject area in the Industry Standards domain area holds Controlled Terminologies as codelists and terms that are imported into the Clinical MDR. When imported as a reference to the source library, they will be kept as a reference to the external versioning information.

- **Foundational Data Standards**
  - The Foundational Data Standards subject area in the Industry Standards domain area holds the clinical data standards, including models, domains and specifications for data representation. When imported into the Clinical MDR as a reference to the source library, they will be kept as external versioning information.

- **Conceptual Standards**
  - The Conceptual Standards subject area holds the CDISC 360 Biomedical Concepts in the form of Activities, Assessments and Analysis Concepts related to data derivation, analysis and analysis results.

- **Therapeutic Area Standards**
  - The Therapeutic Area Standards subject area holds the definitions from the various CDISC Therapeutic Area User Guides, which will reference the CDISC conceptual standards, applied Foundational Standards as well as usage of Controlled Terminology.

- **Dictionaries**
  - The Dictionaries subject area holds rich and highly specialized, medical terminologies that facilitate sharing and exchange of clinical information (e.g., LOINC, MedDRA, SNOMED, etc.).

**Sponsor Standards**
The Sponsor Standards domain area holds the sponsor-defined extensions to the industry standards as well as sponsor-defined, supplemental standards.

- **Sponsor Defined Terminologies**
  - The Sponsor Defined Terminologies subject area holds extensions to standard terminologies, initially only for CDISC Controlled Terminologies.

- **Sponsor Defined Data Standards Extensions**
  - The Sponsor Defined Data Standards Extensions subject area holds extensions and configuration to Foundational Standards, initially only for CDISC Foundational Standards so that they can be extended (i.e., adding standard SDTM variables to SDTM dataset domains or creating sponsor-defined SDTM domains).
• **Sponsor Defined Conceptual Standards**
  o The Sponsor Defined Conceptual Standards subject area holds sponsor-defined Biomedical Concepts in the form of Activities, Assessments and Analysis Concepts related to data derivation, analysis and analysis results based on the CDISC 360 model.

• **Therapeutic Area and Project Standards**
  o The Therapeutic Area and Project Standards subject area holds the sponsor-defined definitions of Therapeutic Area as well as a project-specific selection of standards and will reference the new CDISC conceptual standards, applied Foundational Standards as well as usage of Controlled Terminology. They can refer to what is required or optional to apply by the sponsor.

**Study Definitions**
The Study Definitions domain area holds the study level metadata for study definitions and specifications.

• **Study Definitions**
  o The Study Definitions subject area holds the basic definition for a study in the form of the study identification, study title, phase, type and the selected data standard versions.

• **Study Designs**
  o The Study Designs subject area holds the structural description of the study design in the form of study arms, epochs, elements, visit schedules and planned interventions.

• **Study Selections and Scheduling**
  o The Study Selections and Scheduling subject area holds the selection, configuration and scheduling of biomedical and analysis concepts (e.g., schedule of activities and assessments) for the study.

**Administrative Definitions**
The Administrative Definitions domain area holds the system administrative definitions.

• **Projects**
  o The Projects subject area holds the project definitions and the relationship to therapeutic area, investigational drugs and indications.

• **Access Groups**
  o The Access Groups subject area holds the defined access groups that can be assigned to users.

• **Users**
  o The Users subject area holds the users of the Study Sponsor MDR system with their relationship to access groups and system roles.

• **System Roles**
  o The System Roles subject area holds the defined system roles that can be assigned to users.
Logical Data Model
The Logical Data Model defines the entities, their relationship and attributes of the data domain independently and how they are implemented or exchanged.

As an example, please see Diagram 8 or the Objectives and Endpoints subject area. The Objectives and Endpoints comprise part of the top levels of the conceptual standards that refer to Activities and Assessments, identical similar to Industry and Sponsor Standards, depending on the relationship to the Library entity.

Diagram 8: Objectives and Endpoints Subject Area
<table>
<thead>
<tr>
<th>Entity</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>Entity holds the name and definition of the library that are the source and owner for the elements in the library.</td>
<td>The CDISC Library and a sponsor-specific library.</td>
</tr>
<tr>
<td>ObjectiveTemplate</td>
<td>A sentence syntax for an objective text, including reference to parameters that can be replaced with standardized values.</td>
<td>To demonstrate superiority in the efficacy of [StudyIntervention] to [ComparatorIntervention] in [Assessment]</td>
</tr>
<tr>
<td>Objective</td>
<td>A sentence that represents a specific objective sentence based on a template where the parameters are replaced with specific standardized values.</td>
<td>To demonstrate superiority in the efficacy of human insulin to Metformin in HbA1c</td>
</tr>
<tr>
<td>EndpointTemplate</td>
<td>A sentence syntax for an endpoint text, including reference to parameters that can be replaced with standardized values.</td>
<td>Mean Change from Baseline in [Assessment] after [Timeframe] ([Unit])</td>
</tr>
<tr>
<td>TemplateParameter</td>
<td>A sentence that represents a specific endpoint sentence based on a template where the parameters are replaced with specific standardized values.</td>
<td>Mean Change from Baseline in HbA1c after 26 weeks (%)</td>
</tr>
<tr>
<td>TemplateParameterValue</td>
<td>Hold the specific standardized values, which are categorized by the specific types of template parameters.</td>
<td>Human insulin (StudyIntervention), HbA1c (Assessment)</td>
</tr>
</tbody>
</table>
**Domain Data Model**

The Domain Data Model represents how the data is returned from the API calls and is made as an Object-Oriented class diagram representing the returned result file from an API call. It can be represented in various exchange file formats (JSON, XML, CSV, etc.); for the Sponsor Study MDR system, JSON is mainly used. For certain API endpoints, other file formats will be supported, which can be used to support exports into other systems.

Diagram 9 is an example of an Objectives Templates data domain, sub-part of the conceptual standards subject area, corresponding to the /objective-templates API endpoint. It is identical for Industry and Sponsor Standards, depending on the relationship to the Library entity.
Physical Data Model
The Physical Data Model represents the actual data model as it is implemented in the database. For the Sponsor Study MDR system, the database is a Labeled Property Graph database (Neo4j). The Physical Data Model, therefore, describes the nodes and relationships with properties as implemented in Neo4j.

Diagram 9 is an example of the Objectives Templates nodes and relationships, sub-part of the conceptual standards subject area, corresponding to the /objective-templates API endpoint. It is identical for Industry and Sponsor Standards, depending on the relationship to the Library entity.

Diagram 10: Objectives Templates Nodes and Relationship
One of the main design elements in the physical Neo4j data model is the support of full versioning and audit trail capabilities. This is achieved by separating nodes that identify data elements from the data element values and capturing all data state attributes as relationship properties between the identifier and value nodes. The identifier nodes will include the 'Root' post fix in their name and the value nodes 'Value' as their name post fix. All the state attributes as action, timestamps, user names, change description etc. will be saved as part of relationship properties.

Deployment of Component
- Neo4j database running on Virtual Machine in Azure DevOps.
- Git repositories linked via pipelines directly to VM, releasing updates directly of test data and scripts.
Project Status

The Neo4j database implemented that supports various parts of the Proof of Concept scope and the database implementation actually cover a bit more than implemented in the Study Designer App.

See also: Import Standards 360 Proof of Concept Component

- Load directly from CDISC Library:
  - Controlled Terminology
  - CDASH
  - SDTM
  - Partial ADaM
- Concept-based standards
  - Load a number of supplemental metadata covering additional metadata needed for concept-based standards, see also: WS4 Supplemental Metadata
- Sponsor-defined standards
  - Load extensions to CDISC standards, included in load of WS4 supplemental Metadata
- Load of Study Definition metadata
  - Used to initialize the study metadata set up for the Study Designer App. Covers a bit more scope than illustrated in the App.

Study Metadata Queries

The Study Metadata Queries extract metadata from the Study Metadata Library for downstream usage. They can be executed from a Neo4j Browser, the Neo4j-SAS Interface, from within the Study Designer App or any other client that connect to a Neo4j graph database.

Technologies

Neo4j Cypher graph query scripts.

Scope of Functionalities

The Study Metadata Queries initially cover the following scope:

- Information for the Study Data Standards Plan
- Information to cover basic study design parameters as well as listing objectives and endpoints for a study matching the table structure of the Common Protocol Template
- Listing metadata for study design (TS, TA, TE, TV) datasets
- Listing metadata for SDTM-Define-XML in SAS CST and Pinnacle 21 format
- Listing metadata for CDASH-SDTM and SDTM-ADaM bindings based on CDISC 360 Proof of Concept model
Examples

Samples of how Study Metadata Queries can be made in Neo4j Cypher:

// CPT Objectives and Endpoints
MATCH (s:Study)-->(po:PlannedObjective)-->(ol:ObjectiveLevel),
    (po)--[r1:ROOT_OBJECTIVE]-->(ro:RootObjective)--[r2:HAS_VERSION]-->(o:Objective),
    (po)-->(pe:PlannedEndpoint)-->(e:Endpoint),
WHERE r2.status = 'Final' and r2.date_from <= r1.date < r2.date_to
RETURN s.id, po.order, pe.endpoint_order, ol.name, ro.uri, o.name, ocl.name, e.name
ORDER BY s.id, po.order, pe.endpoint_order;

// List SDTM.TA
MATCH (s:Study {id: $studyid})-->(n:PlannedDesignMatrix)<-(a:PlannedArm),
    (n)<-(e:PlannedEpoch), (n)<-(l:PlannedElement)
RETURN toUpper(s.id) as STUDYID,
    'TA' as DOMAIN,
    toUpper(a.arm_code) as ARMCD,
    toUpper(a.name) as ARM,
    e.epoch_number as TAETORD,
    'E' + n.element_number as ETCD,
    toUpper(l.name) as ELEMENT,
    " as TABRANCH,
    " as TATRANS,
    toUpper(e.name) as EPOCH
ORDER BY n.arm_number, n.epoch_number, n.element_number;

// List SDTM Variables for a Study in P21 format
MATCH (s:Study {id: $studyid})-->(ig:SDTMIGVersion)-[r:REQUIRED_DOMAINS]->(d:SDTMDataset)-->
    (v:SDTMVariable)
OPTIONAL MATCH (v)-->(rl:RootCTCodeList)<-(l:RootCTCodeListName)
RETURN toInteger(v.ordinal) as order,
    d.name as Dataset,
    v.name as Variable,
    v.label as Label,
    v.xmlldatatype as Data_Type,
    toInteger(v.length) as Length,
    " as Significant_Digits,
    " as Format,
    " as Mandatory,
    coalesce(l.name,"") as Codelist,
    " as origin,
    " as Pages,
    " as Method,
    " as Predecessor,
    v.role as role,
    " as comment
ORDER BY toInteger(d.ordinal), toInteger(v.ordinal)
UNION
MATCH (s:Study {id: $studyid})-->(pm:PlannedAssessment)-->(a:Assessment)-[r:MAPPED_TO]-
    >(d:SDTMDataset)-->(v:SDTMVariable),
    (s)-->(ig:SDTMIGVersion)
OPTIONAL MATCH (v)-->(rl:RootCTCodeList)<-(l:RootCTCodeListName)
RETURN toInteger(v.ordinal) as order,
    d.name as Dataset,
    v.name as Variable,
    v.label as Label,
    v.xmldatatype as Data_Type,
    toInteger(v.length) as Length,
    " as Significant_Digits,
    " as Format,
    " as Mandatory,
    coalesce(l.name,"") as Codelist,
    " as origin,
    " as Pages,
    " as Method,
    " as Predecessor,
    v.role as role,
    " as comment
ORDER BY toInteger(d.ordinal), toInteger(v.ordinal);

Sources/Reference Documents
- Git repository Neo4j-StudyLibrary, folder: List-Study-Metadata
- List Study Metadata Task Team
- https://neo4j.com/docs/api/python-driver/current/
- https://pypi.org/project/neo4j/

Neo4j SAS Interface 360 Proof of Concept Component

The Neo4j SAS Interface 360 Proof of Concept Component developed by Workstream 4 uses the Neo4j REST API to submit Cypher (CQL) statements to extract SAS datasets.

Neo4j SAS Interface 360 Proof of Concept Component developed by Workstream 4 uses the Neo4j REST API to submit Cypher (CQL) statements to extract SAS datasets.

Technologies
- SAS 9.4M6 with SAS/Base (PROC HTTP and PROC LUA).

Scope of Functionalities
- Creates template for SAS datasets to be developed
- Submits a GET request via the REST API to the Neo4j server
- Converts the response JSON file into SAS dataset

Deployment of Component
Download from repository and configure folder paths and credentials.
Sources/Reference documents

- The ABCs of the HTTP Procedure
- REST Easier with SAS®: Using the LUA Procedure to Simplify REST API Interactions
- SAS PROC Lua documentation
- Driving SAS® with Lua
- Simple JSON Encode/Decode in Pure Lua
- Lua Reference Manual
- The Neo4j HTTP API

SDTM and ADaM Dataset Automation

Workstream 6 developed a prototype for automating the process of generating SDTM and ADaM datasets. The automation execution uses CDISC 360-enriched-mapping specification and study-level, CDASH/SDTM data as input and delivers target SAS program datasets in SAS7BDAT format. Submission-ready define.xml were also generated using the same metadata and target datasets.

Sub-components

- Front-end application, based on a R-Shiny framework
- Back-end application, based on SAS

Technologies

- Both front and back-end applications run under Microsoft Azure DevLab VM, using Windows Server 2019 Datacenter ver.1809.
- Front-end application uses R version 3.6.3 and R-Studio version 1.2.5033.
- Back-end application uses SAS 9.4 (TS1M6).
- Version control application uses Visual Studio Code version 1.46.1.

Scope of Functionalities

During the automation execution process, the user will be able to import metadata with multiple file formats (default is XML obtained from Study Designer App), review/edit metadata, generate target SAS programs, corresponding SAS datasets, and submission-ready define.xml.

Deployment of Component

SAS datasets, programs, and define.xml are executed using VM Windows platform and stored under the designated study folders.
Examples

User Interface

On the **Data Browser** menu, users will be able to review the input study data:

![Data Browser](image1)

On the **Metadata Import/Editor** menu, users will be able to import study specification metadata, review, and make adjustment as needed.

![Metadata Import/Editor](image2)

Once the specification metadata are ready to go, users can leverage the Automation Engine to generate target-dataset SAS code, use Review Mapping Code to review the generated SAS codes and go back to make adjustment on metadata using Metadata Import/Editor, generate target SAS datasets using Generate Target Datasets, and generate define.xml using Generate Define.xml.
Specification Metadata

Specification Metadata, including initial version from upstream and updated version by user, are stored under designated study metadata location.

The final mapping specification used for the Proof of Concept can be found at:

CDASH to SDTM: [https://www.cdisc.org/sites/default/files/2021-06/cdash2sdtm_v1.1.xlsx](https://www.cdisc.org/sites/default/files/2021-06/cdash2sdtm_v1.1.xlsx)

SDTM to ADaM: [https://www.cdisc.org/sites/default/files/2021-06/sdtm2adam_v1.1.xlsx](https://www.cdisc.org/sites/default/files/2021-06/sdtm2adam_v1.1.xlsx)

Screenshots of the Specification Metadata:

- Metadata Specification - Dimension 1
- Metadata Specification - Dimension 2

### Dataset Level

<table>
<thead>
<tr>
<th>Source</th>
<th>Mapping</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Sequence</td>
<td>Library</td>
<td>Data</td>
</tr>
<tr>
<td>Dataset Condition</td>
<td>Pre Processing</td>
<td>Value Type</td>
</tr>
<tr>
<td>Target Sequence</td>
<td>Target Library</td>
<td>Dataset</td>
</tr>
<tr>
<td>CDASH</td>
<td>VS</td>
<td>5.5</td>
</tr>
<tr>
<td>SDTM</td>
<td>DM</td>
<td>PRE</td>
</tr>
<tr>
<td>WORK</td>
<td>VS4</td>
<td>VS4.SE.EPOCH</td>
</tr>
<tr>
<td>WORK</td>
<td>VS5</td>
<td>VS5.SE.VBASE64</td>
</tr>
<tr>
<td>WORK</td>
<td>VS6</td>
<td>VS6</td>
</tr>
<tr>
<td>WORK</td>
<td>VS7</td>
<td>VS7</td>
</tr>
</tbody>
</table>

### Variable Level

Mapping Specifications: Dataset Level

```r
data VS1;
  set CDASH.VS1;
  variable level: Source Sequence = 1
  run;

proc sort data=VS1; by SUBJID;
proc sort data=CDASH.DM OUT=DM2; by USUBJID;
data VS2;
  merge DM2(in=a) VS1(in=b);
  by USUBJID;
  variable level: Source Sequence = 2
  run;
```

### Value Level

```r
proc sort data=VS6;
  by USUBJID VISITNUM VISDTC;
run;
data SDTM.VS;
  set VS6;
  by USUBJID VISITNUM VISDTC;
  /* variable level: Source Sequence = 5 */
  variable level: Source Sequence = 6
run;
```
Mapping Specifications: Variable Level

```r
44
```
Automation Engines

The Automation Engine SAS programs are stored under designated study program location. Ideally, when final and well packaged, they should be stored under a central macro location.

SAS Programs

Generated SAS programs are stored under designated study program location.

SAS Datasets

Automation-execution-generated-target-SAS datasets, along with trail-design-domain datasets provided by workstream (WS4), are stored under the designated study data location.

Live Demo

SDTM/ADaM Automation Engine - Live Demo
**Project Status**

**Back-end Applications**
- Automation engine is fully functional based on the latest version of specification metadata structure.
- Define-XML engine was fully functional based on previous version of specification metadata structure.

**Front-end Application**
- Infrastructure is built, but functionalities need to be further developed and connected to back-end applications.

**Sources/Reference documents**
- [WS6 - SDTM ADaM Metadata Structure](#)
- [WS6-360-Automation-CDASH-to-SDTM-ADaM-Final-v1.0.pptx](#)

**Illustrations**
Limitations and Assumptions

Assumptions

• Specification metadata is automatically generated based on current data standards and sponsor-study MDR in XML format.
• Specification metadata in structure of Proof of Concept demonstrated layout.
• Corresponding study data are provided via Neo4j SAS interface utility and loaded to the designated study source data area.

Limitations

• Mapping elements may vary from study to study. The Biomedical Concept used for this prototype might be able to provide some target information in a standard, but incomplete way. This will require the user to take lots of time to fill in the missing pieces before automation starts.
• The current specification metadata structure was developed based on a SAS implementation. It will need to be enhanced to automate the execution of applications developed in other languages, such R or SQL.

Suggested Next Steps

• Explore alternative ways to obtain/establish the elements needed, for example, through Biomedical Concept (source), MDR (template containing both info), and IG (target).
• Explore alternative ways to describe relationship between source and target to make it more machine readable and application independent.

TFL Automation 360 Proof of Concept Component

Workstream 6 also developed a prototype for TFL automation. The usage of the automation execution uses CDISC-360 enriched, TFL metadata (ARM++) and study level ADaM data as input and delivers target tables/figures/listings in RTF format.

Sub-components

• Front-end application based on a R-Shiny framework
• Back-end application based on SAS

Technologies

• Both front-end and back-end applications run under Microsoft Azure DevLab VM, using Windows Server 2019 Datacenter ver.1809.
• Front-end application uses R version 3.6.3 and RStudio version 1.2.5033.
• Back-end application uses SAS 9.4 (TS1M6).
• Version control application uses Visual Studio Code version 1.46.1.

Scope of Functionalities

During automation execution process, the user will be able to select TFLs of interest, select TFL layouts from templates, review input data, customize TFL layout and metadata, and generate SAS programs/outputs and define.xml with ARM.
Deployment of Component

Customized metadata, SAS programs, outputs, and define.xml are executed using VM Windows platform and stored under the designated study folders.

Examples

User Interface

On the **Review Data** menu users will be able to review the input study data.

On the **Select Output** menu users will be able to select input metadata, study data, and output wanted to work on.
On the **Select Template** menu users will be able to select the output template to be applied to the selected output.

Once the selected output displays on the side bar, users will be able to customize the template and generate corresponding SAS program.

SAS program execution and output review will be implemented under Windows platform.
TFL Metadata

The TFL Metadata used in the Proof of Concept was based on CDISC ARM v1.0 for Define-XML v2.0, with some extensions included to support automation. This is referred to as the ARM++ Metadata in the Proof of Concept.

TFL metadata is stored under designated study location.
The final mapping specification used for the Proof of Concept can be found at:

**ADaM to TFL**

**WS6_TFL_Metadata_Views_Final_v1.0.xlsx**

**Screenshots of the ARM++ Metadata**
- Sample TFL Metadata (ARM++)
- TFL Metadata from ARM v1.0 for Define-XML v2.0
• TFL Metadata extensions to ARM v1.0 for Define-XML v2.0
• Added to support automation
• New elements added: OUTPUT and STYLE
• ARM v1.0 for Define-XML v2.0 Elements extended: DISPLAY and RESULT
• Extensions based on consideration of many real-world use cases beyond Proof of Concept requirements

Description of TFL Metadata Extensions

- **OUTPUT.** New element (not in ARM v1) that is used to model the file, which contains the TFL (i.e., the Display). An output has a filename, a type (e.g., PDF, RTF, etc.) and contains one or more Display for a specific Analyses within a specific Study.

- **STYLE.** New element (not in ARM v1) that is used to model the stylesheet (e.g., layout, margins, colors, fonts, etc.), which is used for a specific type of output. This allows a single Display to be included in two different outputs with each output styled differently (e.g., same table in a PDF and PowerPoint presentation).
• **PARENT.** New Attribute added to ARM v1 Display and Results elements to support hierarchical modeling of TFL. Used to model (e.g., repeat tables) or where slight variations of a basic table are used in different analyses, etc.

• **VERSION.** New attribute added to ARM v1 Display and Result elements to support different versions of TFL and allow a plot to be extended for final analysis to include more time in follow-up. However, it is the same TFL used in the first interim analysis with slight modification, etc.

• **GROUPING.** New grouping attributes added to ARM v1 Display and Result elements to support grouping of analysis variables. These variables can be considered the 'columns' in a table, where the Display grouping metadata is used to derive 'big N' numerator, and the Result grouping metadata is the 'small-n' denominator. Typically, these variables will use treatment arms, but could also be used to model data (e.g., shift from baseline, etc.).

• **BYVAR.** New attributes added to ARMv1 Display and Result elements to support TFL that repeat an analysis (e.g., by cohort, by visit, etc.). The ByVar works 'within' the display, so a single Display can repeat its results for each value in APERIOD.

• **CodeReference.** New attribute added to ARM v1 Display and Results to support automation. This new attribute is a machine-readable field intended to pass information as to what type of analysis will be performed to create the results and could include parameters for the specific TFL.

**R-Shiny Programs**

R-Shiny-application-related-UI packages are stored under the designated UI_SERVER location.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>www</td>
<td>5/22/2020 2:45 PM</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>generate_sas_code_e2 tier</td>
<td>5/22/2020 2:45 PM</td>
<td>R History Source File</td>
<td>1 KB</td>
</tr>
<tr>
<td>generate_sas_code_demog</td>
<td>5/22/2020 2:45 PM</td>
<td>R Source File</td>
<td>4 KB</td>
</tr>
<tr>
<td>generate_sas_code_eff1421</td>
<td>5/22/2020 2:45 PM</td>
<td>R Source File</td>
<td>4 KB</td>
</tr>
<tr>
<td>generate_xml_code_e2 tier</td>
<td>5/22/2020 2:45 PM</td>
<td>R Source File</td>
<td>5 KB</td>
</tr>
<tr>
<td>generate_xml_code_demog</td>
<td>5/22/2020 2:45 PM</td>
<td>R Source File</td>
<td>4 KB</td>
</tr>
<tr>
<td>generate_xml_code_eff1421</td>
<td>5/22/2020 2:45 PM</td>
<td>R Source File</td>
<td>124 KB</td>
</tr>
<tr>
<td>server</td>
<td>5/22/2020 2:45 PM</td>
<td>R Source File</td>
<td>31 KB</td>
</tr>
<tr>
<td>ui</td>
<td>5/22/2020 2:45 PM</td>
<td>R Source File</td>
<td></td>
</tr>
</tbody>
</table>
```

**Output Programs**

Generated SAS programs for outputs are stored under designated study program location.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmace_edpt_fas</td>
<td>5/22/2020 2:46 PM</td>
<td>SAS System Progr...</td>
<td>3 KB</td>
</tr>
<tr>
<td>tdemog_saf</td>
<td>5/22/2020 2:46 PM</td>
<td>SAS System Progr...</td>
<td>4 KB</td>
</tr>
<tr>
<td>tae_soc_pt_sof</td>
<td>5/22/2020 2:46 PM</td>
<td>SAS System Progr...</td>
<td>3 KB</td>
</tr>
</tbody>
</table>
```
Outputs

Automation-execution-generated outputs are stored under designated study output location.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date Modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>qc</td>
<td>5/22/2020 2:46 PM</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>tae_soc_pt_saf</td>
<td>5/22/2020 2:46 PM</td>
<td>OpenOffice.org</td>
<td>30 KB</td>
</tr>
<tr>
<td>tdemog_saf</td>
<td>5/22/2020 2:46 PM</td>
<td>OpenOffice.org</td>
<td>11 KB</td>
</tr>
<tr>
<td>tmace_ecpt_fas</td>
<td>5/22/2020 2:46 PM</td>
<td>OpenOffice.org</td>
<td>5 KB</td>
</tr>
</tbody>
</table>

Project status

Both front and back-end applications are fully functional with the current TFL metadata structure.

Sources/Reference documents

- WS6 – TFL Automation
- WS6-360-Automation-ADaM-to-TFL-Final-v1.0.pptx
- Live demo: http://youtube.com/watch?v=FxQjVg-1R2M

Illustrations
Limitations and Assumptions

Assumptions

- TFL metadata follows designated structure.
- Output layout template is available.

Limitations

- Not all the ARM++ metadata extensions have been implemented - only sufficient to create the Proof of Concept shells.
- The Proof of Concept only includes Tables, no figures or listing.
- Metadata is loaded from an Excel file; no direct connection to the metadata repository.

Programs and Metadata file:

- `generate_xml_code_demog.R`
- `generate_xml_code_eff1421.R`
- `server.R`
- `ui.R`
- `generate_sas_code_ae2tier.R`
- `generate_sas_code_demog.R`
- `generate_sas_code_eff1421.R`
- `generate_xml_code_ae2tier.R`
- `WS6_TFL_Metadata_Views_Final_v1.0.xlsx`
Execution dependencies:

Input files needed:
1. TFL metadata file - WS6_TFL_Metadata_views_Final_v1.0.xlsx
2. R programs needed
   a. ui.R
   b. server.R
   c. generate_sas_code_ie2tier.R
   d. generate_sas_code_demog.R
   e. generate_sas_code_eff1421.R
   f. generate_xml_code_ie2tier.R
   g. generate_xml_code_demog.R
   h. generate_xml_code_eff1421.R

Software requirements:
1. R Studio 3.6.3
2. Packages: Shiny, Shinyjs, tidyverse, xlsxjars,
3. SAS 9.3 or higher

Data Transformation Engine 360 Proof of Concept Component

The purpose of this Component is to show and document how an agile metadata design can help in augmenting the conceptual-level information into the metadata allowing anyone to programmatically select the content of the standards and apply them to their study-specific requirements. The Data Transformation Engine (DTE) software was used by the 360 Proof of Concept project as a parallel workstream to ascertain the completeness of the 360 project metadata and the level of effort that would be required for a future implementation (i.e., the gaps).

Sub-components
- DTE Metadata Design (DTE Metadata-Concept)
  - Data State Metadata
  - Data Map Metadata
- CDISC 360 Metadata Model (C360 MDR- Define.xml v 2.1 extension SDTM Define-XML files Based on WS 1 Cmaps)
  - XML based C360 MDR Schemas

Technologies
- DTE Software based on SAS (CDISC360 Proof of Concept DTE Component Diagram)
- MindMap
- Excel templates
- DTE environment running on Microsoft Azure DevLab VM (SAS Execution Environment)

Scope of Functionalities
The component is not intended to describe or detail the software that uses DTE agile metadata to achieve the data transformation from one state to another state. However, this points out the
technical description of the metadata design and its use cases wherever required. Metadata design becomes crucial as it becomes the framework in providing full transparency and content availability programmatically, simplifying the integration of standards for automation.

As a result, a concise, agile metadata design is essential to store all required data attributes and mapping information based on the CDISC Standard Models beyond the submission scope by leveraging higher and more consistent quality.

**Deployment of Component**
The component concentrated on covering the following segments, which are aligned with this project.

- Understanding how an adequate metadata design can accelerate metadata-driven automation for the CDISC Standards Data Model
- Providing a gap analysis for CDISC Standards Metadata Model
- Documenting the proposed metadata design for the different use case that demonstrates the ability of the DTE metadata in processing and transforming the data from one state to other
- Gauging the practicality of scaling from the proposed metadata design

**Examples**

**High-level Technical Description of the DTE Metadata**

Data state metadata comprises six metadata elements that describe the states of data as stored in CDASH, SDTM, ADaM or SEND. Data map metadata also includes six metadata elements that will link each source data state to its target data state.
The six metadata elements that describe each data state include:

- Tables – Describe domain-level data attributes
- Columns – Describe element-level data attributes
- Row Definition – Describe value-level data attributes
- Row Columns – Describe value-level data attributes
- Values – Describe codelists and valid values
- Descriptions – Store plain-language descriptions that can be attached to any table, column, or VLM element

The six metadata elements that describe dataflow include:

- Source Prepare – Describes domain-level mapping information
- Columns Map – Describes element-level mapping information
- Wide Thin Map – Describes value-level mapping information
- Values Map – Describes mapping information to redefine values
- Text Snippet – Describes derivation logic in plain language
- Code Snippet – Describes derivation code
Gap analysis for CDISC Standards Metadata Model

This gap analysis is a commendable example of standardization of the metadata design. It will exhibit the control of data state standards by inhabiting a standard metadata design whose content can drive all aspects of data exchange requirements from data state level to dataflow level. This analysis will craft our working proficiency to decide a standard metadata design. The findings will eventually stretch the boundary to yield an end-to-end description of the dataflow between CDASH, SDTM, ADaM, TFLs, and much more.

The real and original objective of having robust adequate metadata is to enable regulatory agencies to integrate study data across studies and organizations in order to more thoroughly understand the safety and efficacy of a drug or drug class. The gap analysis is conducted by file schema, comparing and associating the define elements and their nested elements against the DTE.
From the gap analysis, it is evident that there is a need for information in the metadata beyond what is available in the Define-XML schema.
Use Case Demo Data Against the DTE Metadata Model

### Source Data State Metadata

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Variable Name</th>
<th>Key</th>
<th>Variable Label</th>
<th>Data Type</th>
<th>Variable Length</th>
<th>Values List Name</th>
<th>Values Column Name</th>
<th>SAS Informat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demog</td>
<td>AGE</td>
<td>N</td>
<td>8 V250_</td>
<td>START</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>AGEU</td>
<td>C</td>
<td>18 V251_</td>
<td>START</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>AGEU_STD</td>
<td>C</td>
<td>18 V253_</td>
<td>START</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>SEX</td>
<td>C</td>
<td>48 V254_</td>
<td>START</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>SEX_STD</td>
<td>C</td>
<td>6 V255_</td>
<td>START</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>SITEID</td>
<td>N</td>
<td>8 V268_</td>
<td>START</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>STUDYID</td>
<td>N</td>
<td>8 V304_</td>
<td>START</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>SUBJECT</td>
<td>C</td>
<td>150 V306_</td>
<td>START</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Source to Target Data Map Metadata

<table>
<thead>
<tr>
<th>Source Dataset</th>
<th>Source Variable</th>
<th>Target Dataset</th>
<th>Target Variable</th>
<th>Derivation Syntax Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demog</td>
<td>AGE</td>
<td>DM</td>
<td>AGE</td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>AGEU_STD</td>
<td>DM</td>
<td>AGEU_STD</td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>SEX</td>
<td>DM</td>
<td>SEX</td>
<td></td>
</tr>
<tr>
<td>Demog</td>
<td>STUDYID</td>
<td>DM</td>
<td>USUBJID</td>
<td>USUBJID</td>
</tr>
<tr>
<td>Demog</td>
<td>SITEID</td>
<td>DM</td>
<td>USUBJID</td>
<td>USUBJID</td>
</tr>
<tr>
<td>Demog</td>
<td>SUBJECT</td>
<td>DM</td>
<td>USUBJID</td>
<td>USUBJID</td>
</tr>
</tbody>
</table>

### Target Data State Metadata

**Project Status**

From the principles followed in the DTE metadata design, it is clear that this design structure is unique. It does not assume any standards; therefore, it is a unified metadata design, not only for all CDISC Foundational Standards, but also for a broad range of implementations, which can describe any relational data. Additionally, the design bridges the gap in the Foundational Standards. The DTE team presented the unified metadata design to project workstreams and demonstrated it using DTE software, which carried out the data transformation from one data state to another using this metadata design.

**Sources/Reference documents**

- [DTE Metadata-Concept](#)
- [DTE Documentations](#)
Suggested Next Steps

A next step could be to publish the Foundational Standards in this standard DTE metadata design. This solution would help us attain efficiency and could serve as a standard way to communicate data and dataflow specifications with CROs, labs, PRO, and other third-party data sources. Additional components are necessary, but a shared metadata design is the logical next step.