PharmaSUG 2023 - Paper SS-140

Working with Biomedical Concepts and SDTM Dataset Specializations for Define-XML v2.1 using SAS© openCST

Lex Jansen, CDISC[1]

ABSTRACT

Biomedical Concepts are units of knowledge that relate to real-world entities. Getting Biomedical Concepts off the ground has been a long and challenging journey. There is little to debate that in theory Biomedical Concepts make sense, but implementation has been a great challenge with little to no realization of benefits. Perhaps that is because the scope has been too large and complex, making implementation extremely difficult. For this reason, CDISC has developed a simplified approach and model which includes an abstract conceptual layer that provides semantics as well as a simplified implementation layer of preconfigured Dataset Specializations (CDASH, SDTM, ...) linked to Biomedical Concepts. SDTM Dataset Specializations are ready to use building blocks for Define-XML. This provides immediate benefits to SDTM programmers and opens the door to efficient programming and automation. Biomedical Concepts are now available in CDISC Library via the API. This paper shows how SAS© can work with Biomedical Concepts and SDTM Dataset Specializations. This paper will show how SDTM Dataset Specializations can be used by the Open Source release of SAS Clinical Standards Toolkit (openCST) for the creation of Value Level Metadata in Define-XML v2.1.

Keywords: CDISC, Biomedical Concepts, SDTM Dataset Specializations, Define-XML, define.xml, Value Level Metadata, metadata

INTRODUCTION

CDISC kicked off the Conceptual and Operational Standards Metadata Services (COSMoS) project in 2022 [1], taking a pragmatic, iterative approach to creating Biomedical Concepts (BCs) and representing them in the Foundational Standards as Dataset Specializations with Value Level Metadata definitions. Biomedical Concepts fill gaps in the current standards by adding semantics, variable relationships, and the detailed metadata needed to generate Case Report Forms (CRFs) or a Define-XML document.

CDISC Biomedical Concepts include a two layered approach:

- Conceptual/abstract layer which provides a standards-agnostic, unambiguous semantic definition largely based on concepts from the NCIt (National Cancer Institute Thesaurus).
- Implementation layer based on valid CDISC Dataset specializations that provide value level metadata definitions and facilitate metadata driven automation.

The data model is flexible and can accommodate other standards (e.g., CDASH, HL7 FHIR) by defining additional dataset specializations.

This paper will concentrate on the implementation layer, which is based on CDISC Dataset Specializations. Especially, we show how SDTM Dataset Specializations can be used as building blocks for Value Level Metadata (VLM) in Define-XML v2.1.

All code used in this paper is available on GitHub: https://github.com/lexjansen/sas-papers

CDISC BIOMEDICAL CONCEPTS

What is a Biomedical Concept? We use the definition from the following text:

> BCs address metadata gaps in the current CDISC standards. They provide the conceptual definitions supporting the existing CDISC Foundational Standards metadata. This conceptual metadata is necessary to generate operationally ready Data Elements (DE). These operational DEs represent the

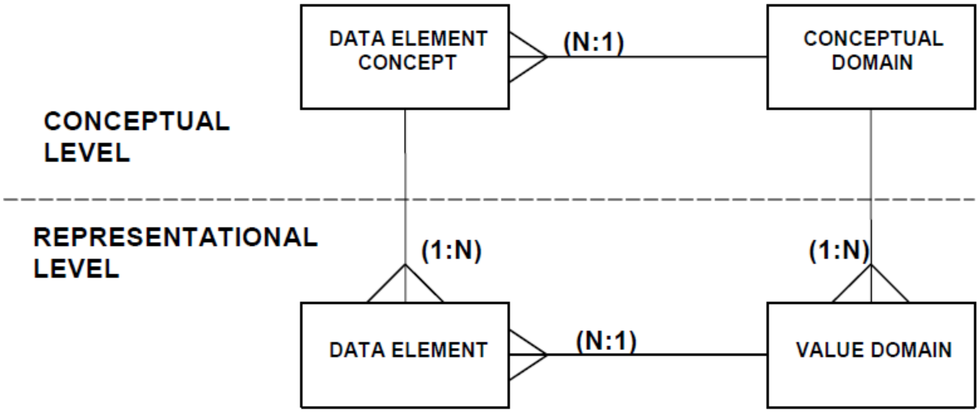
[1] Contract through Lex Jansen Consulting LLC

> detail needed to create the dataset variable definitions and value level metadata needed to generate a Define-XML document.
>
> A BC is a unit of knowledge created by a unique combination of characteristics. As noted above, BCs complement the existing standards, but omit the operationalization of the standards. That is, BCs exist independent of any given standards implementation, such as SDTMIG v3.2 or CDASHIG v2.0. A BC specifies an observation concept, or what should be observed for a specific subject assessment in a clinical study, but not how to capture the data or how to group observations together [2].

The CDISC Biomedical Concepts are a pragmatic and simplified implementation that takes inspiration from the ISO 11179 standard. The following quote is also from [2]:

> An observation concept consists of one or more Data Element Concepts (DEC) as defined in the ISO 11179 standard [3]. DECs represent the meaning of a variable and consist of a concept code identifier and a definition. DEs, or operational variables, consist of a unique pairing of a DEC and a Value Domain (VD). A VD is the domain of possible values for a DE which include data types, formats, and constraints. A DE is formed when a DEC takes on a specific representation or VD. Display 1 illustrates the ISO 11179 model.

Display 1. Overview Model for ISO/IEC 11179 Metadata Registry



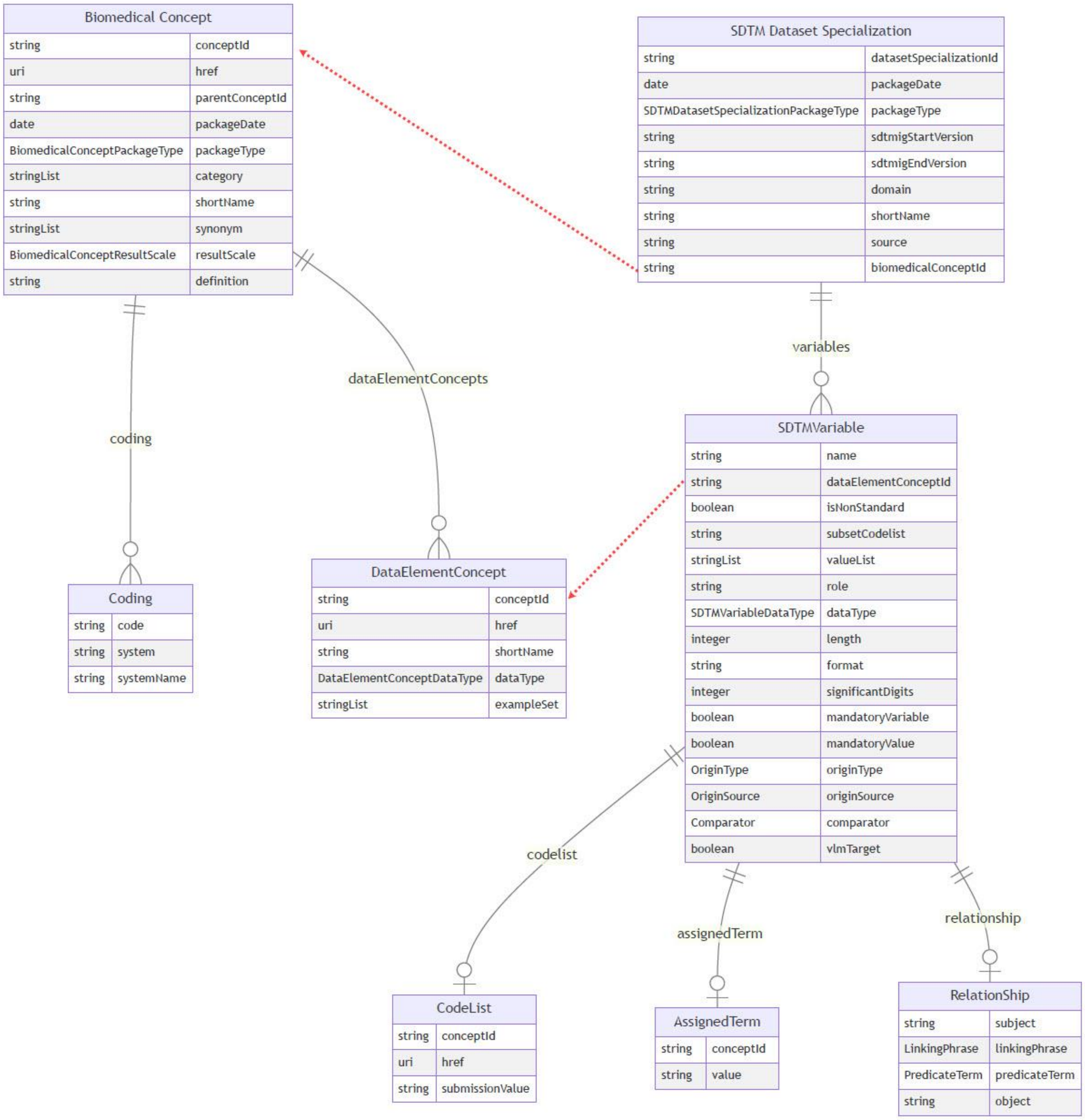
The pragmatic CDISC Biomedical Concepts approach, as shown in Display 2 and Display 3, helps implementers by:

- Adding explicit relationships to the existing standards such that they exist as explicit relationships at the variable and dataset level. For example, a results variable may have a HAS_UNITS relationship to the units variable. The missing relationships in the standards belong as part of the standard itself, not added through an additional concept layer.
- Adding explicit operational metadata to the existing standards such that they are defined for each variable. A variable definition should include all the operational metadata needed to create a Define-XML document. So, each variable should have a specific data type (e.g., integer, decimal, datetime), significant digits, and length defined. Where appropriate, the appropriate codelist subset should be associated with the variable.
- Associating example values with variables to highlight how a variable might be represented.

The model and schema describing the CDISC Biomedical Concepts and SDTM Dataset Specializations can be found on GitHub (https://github.com/cdisc-org/COSMoS).

Display 2 shows the class diagrams for the Biomedical Concepts and the SDTM Dataset Specializations. The red lines show how an SDTM Dataset Specialization has a reference to a Biomedical Concept and SDTM variables have references to Data Element Concepts.

Display 2. Class diagrams for the Biomedical Concepts and the SDTM Dataset Specializations



Biomedical Concept	
string	conceptId
uri	href
string	parentConceptId
date	packageDate
BiomedicalConceptPackageType	packageType
stringList	category
string	shortName
stringList	synonym
BiomedicalConceptResultScale	resultScale
string	definition

SDTM Dataset Specialization	
string	datasetSpecializationId
date	packageDate
SDTMDatasetSpecializationPackageType	packageType
string	sdtmigStartVersion
string	sdtmigEndVersion
string	domain
string	shortName
string	source
string	biomedicalConceptId

SDTMVariable	
string	name
string	dataElementConceptId
boolean	isNonStandard
string	subsetCodelist
stringList	valueList
string	role
SDTMVariableDataType	dataType
integer	length
string	format
integer	significantDigits
boolean	mandatoryVariable
boolean	mandatoryValue
OriginType	originType
OriginSource	originSource
Comparator	comparator
boolean	vlmTarget

Coding	
string	code
string	system
string	systemName

DataElementConcept	
string	conceptId
uri	href
string	shortName
DataElementConceptDataType	dataType
stringList	exampleSet

CodeList	
string	conceptId
uri	href
string	submissionValue

AssignedTerm	
string	conceptId
string	value

RelationShip	
string	subject
LinkingPhrase	linkingPhrase
PredicateTerm	predicateTerm
string	object

Display 3 shows a Biomedical Concept (Systolic Blood Pressure) and an associated SDTM Dataset Specialization (SYSBP).

Display 3. Biomedical Concept (Systolic Blood Pressure) and an associated SDTM Dataset Specialization (SYSBP)

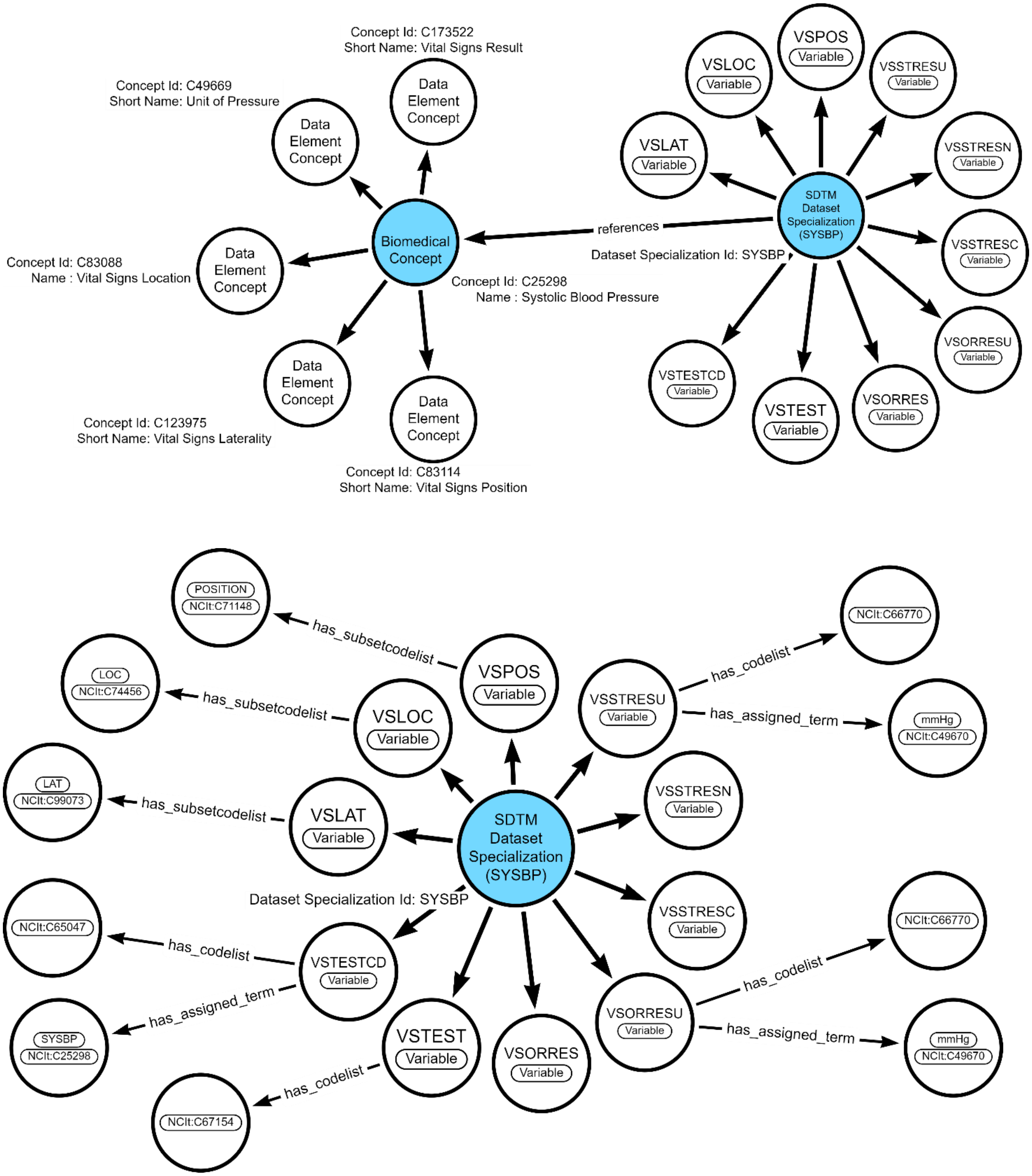


Table 1 lists some of the attributes that are part of a SDTM Dataset Specialization, together with their description.

Table 1. SDTM Dataset Specialization attributes

Attribute	Description
datasetSpecializationId	Identifier for SDTM Value Level Metadata group
domain	Domain for the SDTM specialization group
shortName	SDTM group short name which provides a user friendly and intuitive name for the datasetSpecializationId
source	SDTM VLM Source which categorizes VLM groups by topic variable
sdtmigStartVersion	The earliest SDTMIG version applicable to the SDTM dataset specialization
sdtmigEndVersion	The last SDTMIG version that is applicable to the SDTM dataset specialization
biomedicalConceptId	Biomedical Concept identifier

Every SDTM Dataset Specialization contains one or more variables. Table 2 lists some of the variable attributes and their description.

Table 2. SDTM Dataset Specialization Variable attributes

Attribute		Description
name		Variable included in the SDTM dataset specialization
dataElementConceptId		Biomedical Concept Data Element Concept identifier
codelist	conceptId	C-code for a codelist in NCIt
	href	Link to NCIt for the codelist
	submissionValue	CDISC submission value for the codelist
subsetCodelist		Subset codelist short name
valueList		List of SDTM submission values used if subset codelist is not applicable
assignedTerm	conceptId	C-code for assigned term in NCIt
	value	Submission value for assigned term in NCIt if it exists, or an assigned value which will be the default value
role		SDTM variable role
relationship	Subject	Subject in a variable relationship
	linkingPhrase	Variable relationship descriptive linking phrase
	predicateTerm	Short variable relationship linking phrase for programming
	object	Object in a variable relationship
datatype		Variable data type
length		Variable length
format		Variable display format
significantDigits		Variable significant digits
originType		Variable origin type (Assigned, Collected, Derived, Protocol, Predecessor)
originSource		Variable origin source (Investigator, Sponsor, Subject, Vendor)
comparator		Comparison operator for SDTM group variables included in VLM (EQ, IN)
vlmTarget		Target variable for VLM (true/false)

ACCESSING BIOMEDICAL CONCEPTS AND SDTM DATASET SPECIALIZATIONS

CDISC Biomedical Concepts and SDTM Dataset Specializations are published by CDISC as packages and can be accessed through a REST API in a comparable way as data standards metadata and controlled terminology can be accessed from the CDISC Library [4]. At the time of the writing of this paper (April 2023), three packages with Biomedical Concepts and SDTM Dataset Specializations were released (2022-10-26, 2023-02-13, 2023-03-31). A package is incremental and contains new or updated content.

Table 3 gives a list of the API endpoints that are currently available. More API endpoints are being developed by CDISC. To be able to use the API it is required that the user has an API key [5].

Table 3. REST API Endpoints to access Biomedical Concepts and SDTM Dataset Specializations

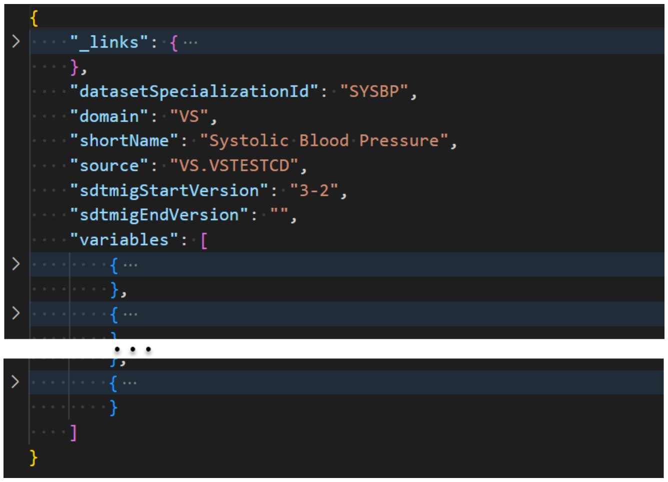
API Endpoint / Description
`/mdr/bc/packages` Get Biomedical Concept Package List
`/mdr/bc/packages/{package}/biomedicalconcepts` Get Biomedical Concept List
`/mdr/bc/packages/{package}/biomedicalconcepts/{biomedicalconcept}` Get Biomedical Concept
`/mdr/specializations/sdtm/packages` Get SDTM Dataset Specialization Package List
`/mdr/specializations/sdtm/packages/{package}/datasetspecializations` Get SDTM Dataset Specialization List
`/mdr/specializations/sdtm/packages/{package}/datasetspecializations/{datasetspecialization}` Get SDTM Dataset Specialization

The API request returns JSON content by default. Display 4 shows part of the JSON response from the API request to get the Diastolic Blood Pressure:

```
/mdr/specializations/sdtm/packages/2022-10-26/datasetspecializations/SYSBP
```

The response in Display 4 shows some SDTM Data Specialization metadata, like the datasetSpecializationId ("SYSBP"), the domain ("VS"), and an array of variables.

Display 4. Example SDTM Dataset Specialization JSON Response Fragments from an API Request

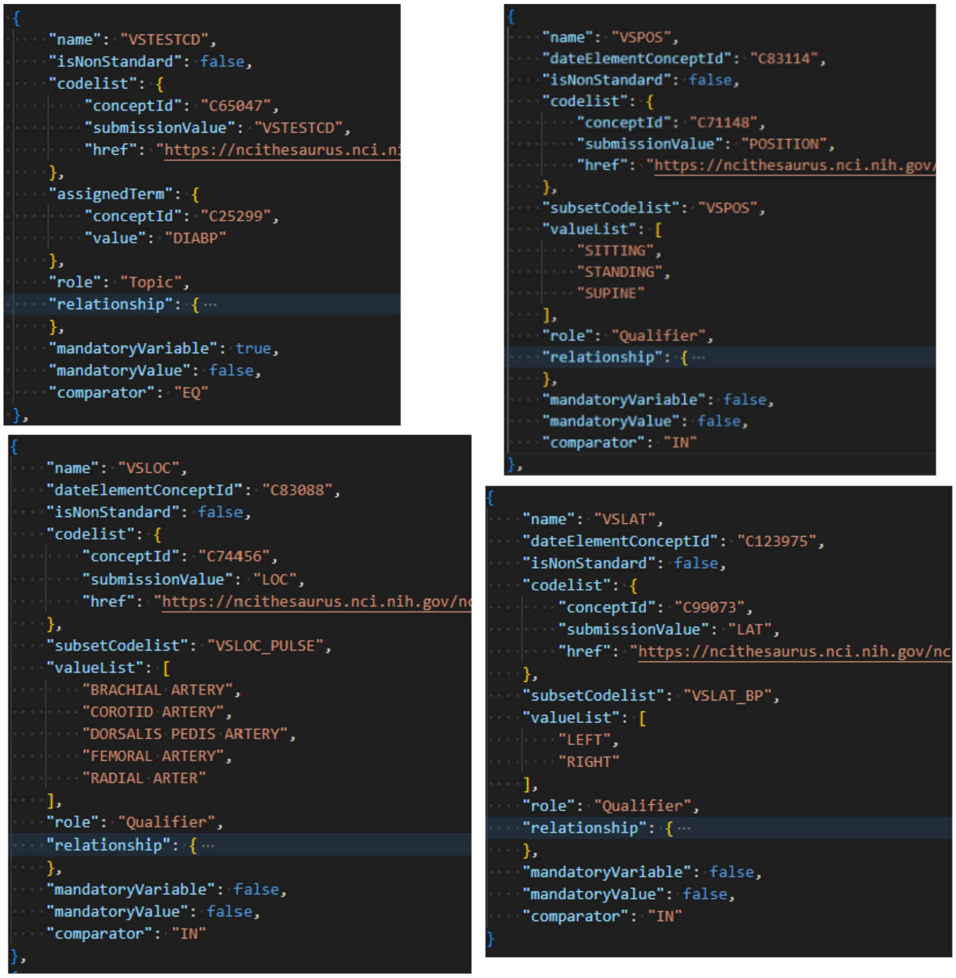


```
{
    "_links": {…
    },
    "datasetSpecializationId": "SYSBP",
    "domain": "VS",
    "shortName": "Systolic Blood Pressure",
    "source": "VS.VSTESTCD",
    "sdtmigStartVersion": "3-2",
    "sdtmigEndVersion": "",
    "variables": [
        {…
        },
        {…
        },
        • • •
        {…
        }
    ]
}
```

Display 5 shows a selection of the variables in the response: VSTESTCD, VSPOS, VSLOC, VSLAT. These variables have in common that they all have a **comparator** property equal to either "IN" or "EQ".

Also, they all have an **assigned term** (VSTESTCD) or a **codelist** (VSPOS, VSLOC and VSLAT).

Display 5 Example SDTM Dataset Specialization JSON Response Fragments from an API Request



```
{
    "name": "VSTESTCD",
    "isNonStandard": false,
    "codelist": {
        "conceptId": "C65047",
        "submissionValue": "VSTESTCD",
        "href": "https://ncithesaurus.nci.ni
    },
    "assignedTerm": {
        "conceptId": "C25299",
        "value": "DIABP"
    },
    "role": "Topic",
    "relationship": {⋯
    },
    "mandatoryVariable": true,
    "mandatoryValue": false,
    "comparator": "EQ"
},
```

```
{
    "name": "VSPOS",
    "dateElementConceptId": "C83114",
    "isNonStandard": false,
    "codelist": {
        "conceptId": "C71148",
        "submissionValue": "POSITION",
        "href": "https://ncithesaurus.nci.nih.gov/
    },
    "subsetCodelist": "VSPOS",
    "valueList": [
        "SITTING",
        "STANDING",
        "SUPINE"
    ],
    "role": "Qualifier",
    "relationship": {⋯
    },
    "mandatoryVariable": false,
    "mandatoryValue": false,
    "comparator": "IN"
},
```

```
{
    "name": "VSLOC",
    "dateElementConceptId": "C83088",
    "isNonStandard": false,
    "codelist": {
        "conceptId": "C74456",
        "submissionValue": "LOC",
        "href": "https://ncithesaurus.nci.nih.gov/nc
    },
    "subsetCodelist": "VSLOC_PULSE",
    "valueList": [
        "BRACHIAL ARTERY",
        "COROTID ARTERY",
        "DORSALIS PEDIS ARTERY",
        "FEMORAL ARTERY",
        "RADIAL ARTER"
    ],
    "role": "Qualifier",
    "relationship": {⋯
    },
    "mandatoryVariable": false,
    "mandatoryValue": false,
    "comparator": "IN"
},
```

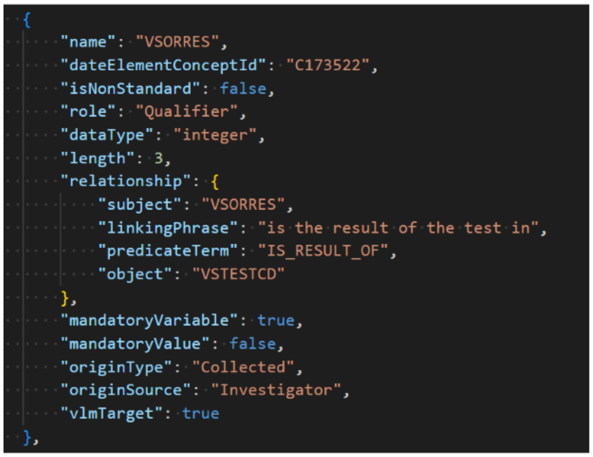
```
{
    "name": "VSLAT",
    "dateElementConceptId": "C123975",
    "isNonStandard": false,
    "codelist": {
        "conceptId": "C99073",
        "submissionValue": "LAT",
        "href": "https://ncithesaurus.nci.nih.gov/nc
    },
    "subsetCodelist": "VSLAT_BP",
    "valueList": [
        "LEFT",
        "RIGHT"
    ],
    "role": "Qualifier",
    "relationship": {⋯
    },
    "mandatoryVariable": false,
    "mandatoryValue": false,
    "comparator": "IN"
}
```

Since the variables VSTESTCD, VSPOS, VSLOC, and VSLAT have a comparator (last property in each screenshot), they will become conditions in a (composite) WhereClause in Value Level Metadata in a Define-XML document:

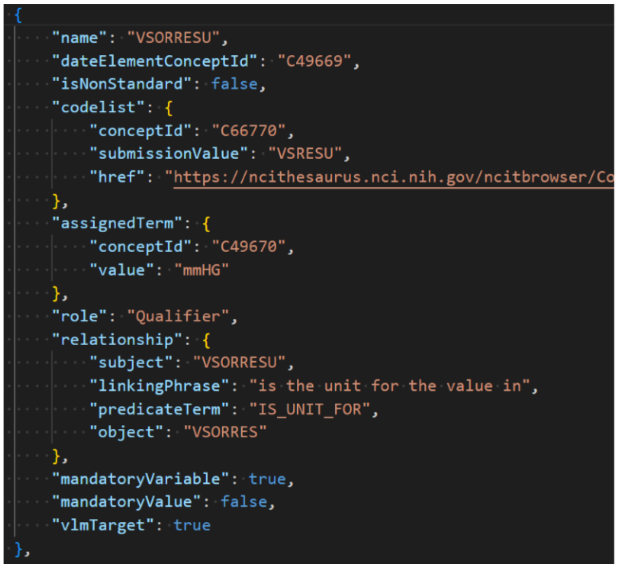
VSTESTCD EQ "SYSBP"	VSPOS IN ("SITTING","STANDING","SUPINE")	VSLOC IN ("BRACHIAL ARTERY","CAROTID ARTERY","DORSALIS PEDIS ARTERY","FEMORAL ARTERY","RADIAL ARTERY")	VSLAT IN ("LEFT","RIGHT")

Display 6 shows the VSORRES and VSORRESU variables in the JSON response. These variables both have the **vlmTarget** attribute with a value of “true”. This means that these variables are described by Value Level Metadata. Other variable attributes in the response, like length, datatype, originType, originSource, codelist, and assignedTerm describe the variable as part of the Value Level Metadata under the specific condition defined by the WhereClause.

Display 6. Example SDTM Dataset Specialization JSON Response Fragments from an API Request



```
{
    "name": "VSORRES",
    "dataElementConceptId": "C173522",
    "isNonStandard": false,
    "role": "Qualifier",
    "dataType": "integer",
    "length": 3,
    "relationship": {
        "subject": "VSORRES",
        "linkingPhrase": "is the result of the test in",
        "predicateTerm": "IS_RESULT_OF",
        "object": "VSTESTCD"
    },
    "mandatoryVariable": true,
    "mandatoryValue": false,
    "originType": "Collected",
    "originSource": "Investigator",
    "vlmTarget": true
},
```



```
{
    "name": "VSORRESU",
    "dataElementConceptId": "C49669",
    "isNonStandard": false,
    "codelist": {
        "conceptId": "C66770",
        "submissionValue": "VSRESU",
        "href": "https://ncithesaurus.nci.nih.gov/ncitbrowser/Co
    },
    "assignedTerm": {
        "conceptId": "C49670",
        "value": "mmHG"
    },
    "role": "Qualifier",
    "relationship": {
        "subject": "VSORRESU",
        "linkingPhrase": "is the unit for the value in",
        "predicateTerm": "IS_UNIT_FOR",
        "object": "VSORRES"
    },
    "mandatoryVariable": true,
    "mandatoryValue": false,
    "vlmTarget": true
},
```

REST API REQUESTS IN SAS

REST API requests are widely supported by software languages. PROC HTTP is a powerful SAS procedure to issue Hypertext Transfer Protocol (HTTP) requests [6] [7] [8]. The procedure includes a DEBUG statement, response status macro variables, and the ability to specify a time-out period for requests.

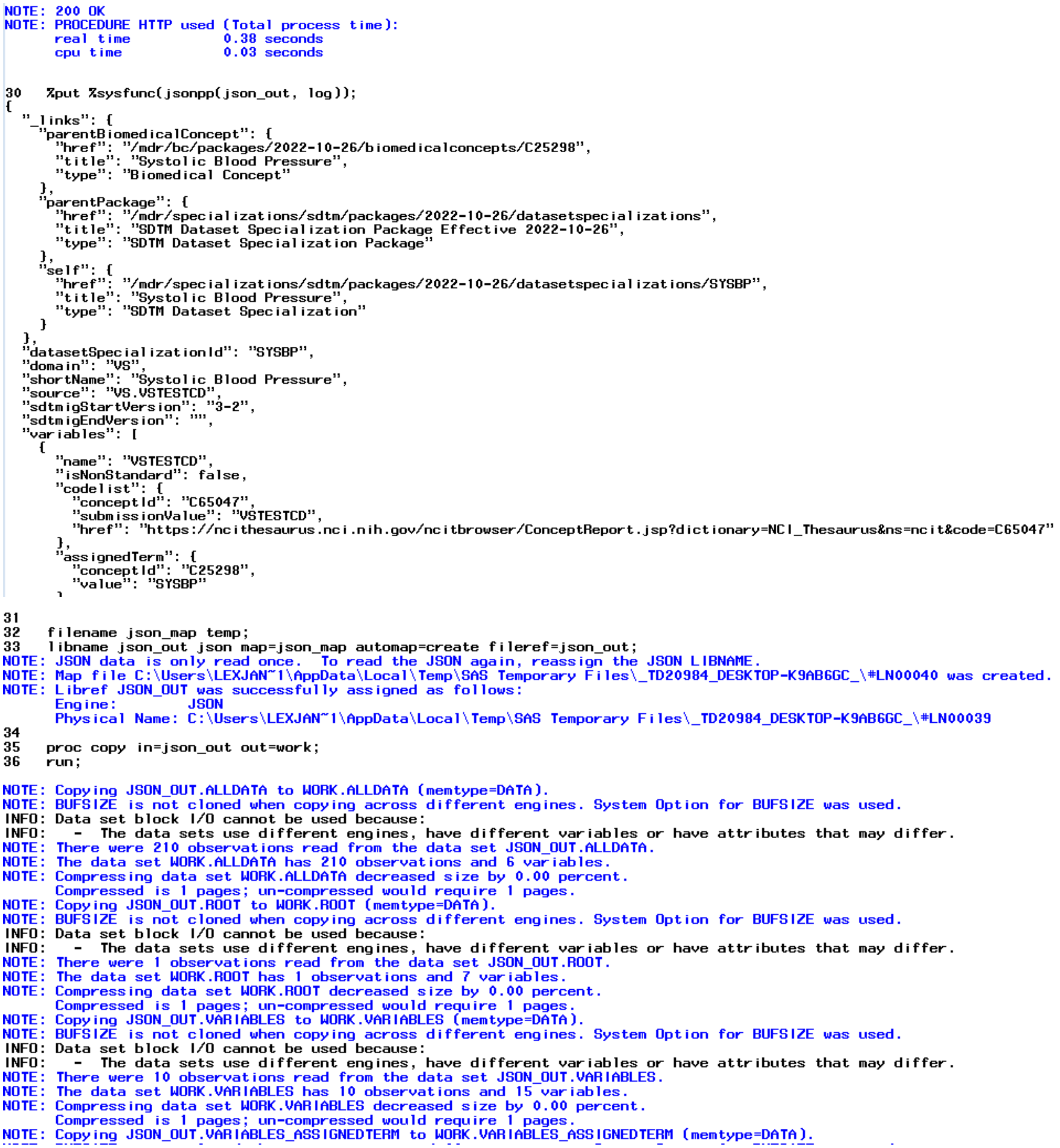
Below is an example of requesting the SYSBP SDTM Dataset Specialization from the 2022-10-26 package. It is expected that the macro variable &ApiKey contains your personal API key that you created at the CDISC Library API Management Developer Portal [5].

```
%let ApiKey=<your_personal_api_key>;
%let baseURL=https://library.cdisc.org/api;

filename json_out temp;
proc http
  method = 'GET'
  url="&baseURL/mdr/specializations/sdtm/packages/2022-10-26/datasetspecializations/SYSBP"
  out=json_out;
  headers
    "api-key" = "&ApiKey"
    "Accept" = "application/json";
run;
%put %sysfunc(jsonpp(json_out, log));

filename json_map temp;
libname json_out json map=json_map automap=create fileref=json_out;

proc copy in = json_out out = work;
run;
```



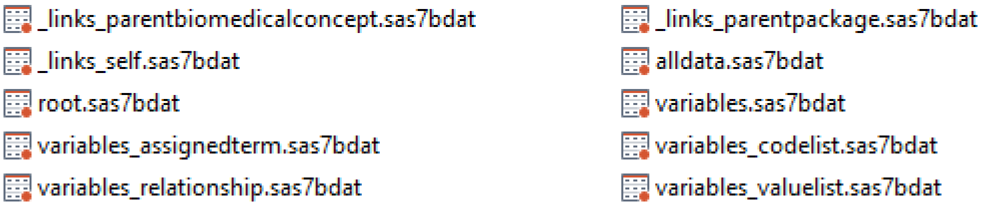
```
NOTE: 200 OK
NOTE: PROCEDURE HTTP used (Total process time):
      real time           0.38 seconds
      cpu time            0.03 seconds


30   %put %sysfunc(jsonpp(json_out, log));
{
  "_links": {
    "parentBiomedicalConcept": {
      "href": "/mdr/bc/packages/2022-10-26/biomedicalconcepts/C25298",
      "title": "Systolic Blood Pressure",
      "type": "Biomedical Concept"
    },
    "parentPackage": {
      "href": "/mdr/specializations/sdtm/packages/2022-10-26/datasetspecializations",
      "title": "SDTM Dataset Specialization Package Effective 2022-10-26",
      "type": "SDTM Dataset Specialization Package"
    },
    "self": {
      "href": "/mdr/specializations/sdtm/packages/2022-10-26/datasetspecializations/SYSBP",
      "title": "Systolic Blood Pressure",
      "type": "SDTM Dataset Specialization"
    }
  },
  "datasetSpecializationId": "SYSBP",
  "domain": "VS",
  "shortName": "Systolic Blood Pressure",
  "source": "VS.VSTESTCD",
  "sdtmigStartVersion": "3-2",
  "sdtmigEndVersion": "",
  "variables": [
    {
      "name": "VSTESTCD",
      "isNonStandard": false,
      "codelist": {
        "conceptId": "C65047",
        "submissionValue": "VSTESTCD",
        "href": "https://ncithesaurus.nci.nih.gov/ncitbrowser/ConceptReport.jsp?dictionary=NCI_Thesaurus&ns=ncit&code=C65047"
      },
      "assignedTerm": {
        "conceptId": "C25298",
        "value": "SYSBP"
      }

31
32   filename json_map temp;
33   libname json_out json map=json_map automap=create fileref=json_out;
NOTE: JSON data is only read once.  To read the JSON again, reassign the JSON LIBNAME.
NOTE: Map file C:\Users\LEXJAN~1\AppData\Local\Temp\SAS Temporary Files\_TD20984_DESKTOP-K9AB6GC_\#LN00040 was created.
NOTE: Libref JSON_OUT was successfully assigned as follows:
      Engine:        JSON
      Physical Name: C:\Users\LEXJAN~1\AppData\Local\Temp\SAS Temporary Files\_TD20984_DESKTOP-K9AB6GC_\#LN00039
34
35   proc copy in=json_out out=work;
36   run;

NOTE: Copying JSON_OUT.ALLDATA to WORK.ALLDATA (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines. System Option for BUFSIZE was used.
INFO: Data set block I/O cannot be used because:
INFO:   -  The data sets use different engines, have different variables or have attributes that may differ.
NOTE: There were 210 observations read from the data set JSON_OUT.ALLDATA.
NOTE: The data set WORK.ALLDATA has 210 observations and 6 variables.
NOTE: Compressing data set WORK.ALLDATA decreased size by 0.00 percent.
      Compressed is 1 pages; un-compressed would require 1 pages.
NOTE: Copying JSON_OUT.ROOT to WORK.ROOT (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines. System Option for BUFSIZE was used.
INFO: Data set block I/O cannot be used because:
INFO:   -  The data sets use different engines, have different variables or have attributes that may differ.
NOTE: There were 1 observations read from the data set JSON_OUT.ROOT.
NOTE: The data set WORK.ROOT has 1 observations and 7 variables.
NOTE: Compressing data set WORK.ROOT decreased size by 0.00 percent.
      Compressed is 1 pages; un-compressed would require 1 pages.
NOTE: Copying JSON_OUT.VARIABLES to WORK.VARIABLES (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines. System Option for BUFSIZE was used.
INFO: Data set block I/O cannot be used because:
INFO:   -  The data sets use different engines, have different variables or have attributes that may differ.
NOTE: There were 10 observations read from the data set JSON_OUT.VARIABLES.
NOTE: The data set WORK.VARIABLES has 10 observations and 15 variables.
NOTE: Compressing data set WORK.VARIABLES decreased size by 0.00 percent.
      Compressed is 1 pages; un-compressed would require 1 pages.
NOTE: Copying JSON_OUT.VARIABLES_ASSIGNEDTERM to WORK.VARIABLES_ASSIGNEDTERM (memtype=DATA).
```

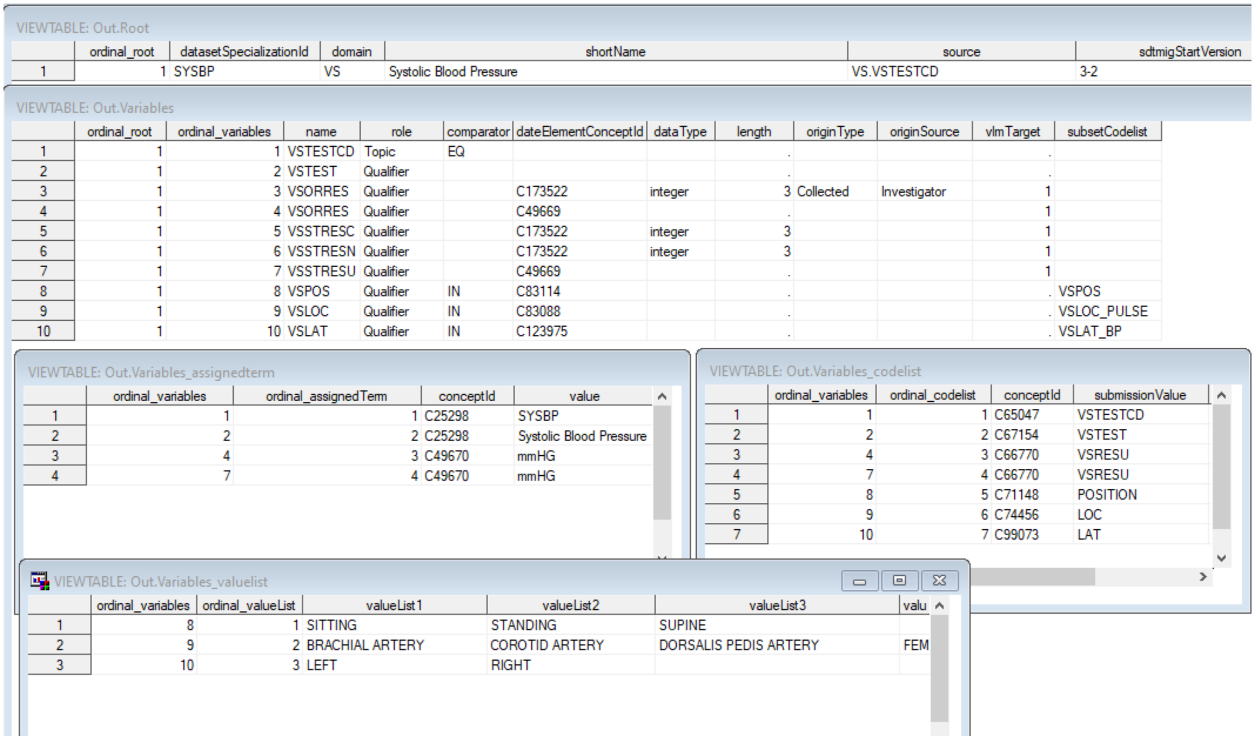
This code results in 10 SAS datasets, some of which need to be merged to get the expected SAS dataset with SDTM Dataset Specialization metadata.



```
_links_parentbiomedicalconcept.sas7bdat    _links_parentpackage.sas7bdat
_links_self.sas7bdat                       alldata.sas7bdat
root.sas7bdat                              variables.sas7bdat
variables_assignedterm.sas7bdat            variables_codelist.sas7bdat
variables_relationship.sas7bdat            variables_valuelist.sas7bdat
```

Display 7 shows the resulting root, variables, variables_codelists, variables_valuelist and variables_assigned_term datasets.

Display 7. Datasets created by the SAS JSON engine



VIEWTABLE: Out.Root

	ordinal_root	datasetSpecializationId	domain	shortName	source	sdtmigStartVersion
1	1	SYSBP	VS	Systolic Blood Pressure	VS.VSTESTCD	3-2

VIEWTABLE: Out.Variables

	ordinal_root	ordinal_variables	name	role	comparator	dateElementConceptId	dataType	length	originType	originSource	vlmTarget	subsetCodelist
1	1	1	VSTESTCD	Topic	EQ			.			.	
2	1	2	VSTEST	Qualifier				.			.	
3	1	3	VSORRES	Qualifier		C173522	integer	3	Collected	Investigator	1	
4	1	4	VSORRES	Qualifier		C49669		.			1	
5	1	5	VSSTRESC	Qualifier		C173522	integer	3			1	
6	1	6	VSSTRESN	Qualifier		C173522	integer	3			1	
7	1	7	VSSTRESU	Qualifier		C49669		.			1	
8	1	8	VSPOS	Qualifier	IN	C83114		.			.	VSPOS
9	1	9	VSLOC	Qualifier	IN	C83088		.			.	VSLOC_PULSE
10	1	10	VSLAT	Qualifier	IN	C123975		.			.	VSLAT_BP

VIEWTABLE: Out.Variables_assignedterm

	ordinal_variables	ordinal_assignedTerm	conceptId	value
1	1	1	C25298	SYSBP
2	2	2	C25298	Systolic Blood Pressure
3	4	3	C49670	mmHG
4	7	4	C49670	mmHG

VIEWTABLE: Out.Variables_codelist

	ordinal_variables	ordinal_codelist	conceptId	submissionValue
1	1	1	C65047	VSTESTCD
2	2	2	C67154	VSTEST
3	4	3	C66770	VSRESU
4	7	4	C66770	VSRESU
5	8	5	C71148	POSITION
6	9	6	C74456	LOC
7	10	7	C99073	LAT

VIEWTABLE: Out.Variables_valuelist

	ordinal_variables	ordinal_valueList	valueList1	valueList2	valueList3	valu
1	8	1	SITTING	STANDING	SUPINE	
2	9	2	BRACHIAL ARTERY	COROTID ARTERY	DORSALIS PEDIS ARTERY	FEM
3	10	3	LEFT	RIGHT		

The code above uses the SAS JSON engine libname, which was introduced in SAS 9.4TS1M4 [10]. Using SAS can be cumbersome when dealing with complex JSON files as it requires the merging of potentially a large number of datasets and dealing with the management of JSON MAP files to correct decisions that the SAS JSON automapper makes in terms of variable types and variable lengths. Earlier papers by the author have used PROC LUA to parse JSON files in SAS [4][11][12]. The JSON for SDTM Dataset Specializations is not too complex. So, for this paper I decided to use the native SAS JSON engine libname.

As mentioned, there are challenges when combining the various datasets with SDTM Dataset Specialization endpoints into a single dataset:

- When the JSON map is automatically generated (automap=create) the datasets, variables and length of variables depend on the endpoint. For example, Dataset Specializations will not always have a valuelist attribute, so there may not be a variables_valuelist dataset.
- Since the valuelist attribute is really a JSON array, the number of variables valueList1, valueList2, … is not known beforehand

The code, which can be found in the repository[2], manages these challenges by creating a template for the expected dataset with all variables and their lengths, checking whether datasets exist and pre-processing the variables_valuelist dataset before merging.

The following SAS code shows how to combine an unknown number of valuelist1, valuelist2, … variables into a _valuelist variable using an array:

[2] The GitHub repository that supports this paper is located at: https://github.com/lexjansen/sas-papers

```
data work.variables_valuelist(drop=valueList:);
  set work.variables_valuelist;
  length _valueList $ 2048;
  array valueList_{*} $ 1024 valueList:;
  _valueList = catx(";", OF valueList_{*});
run;
```

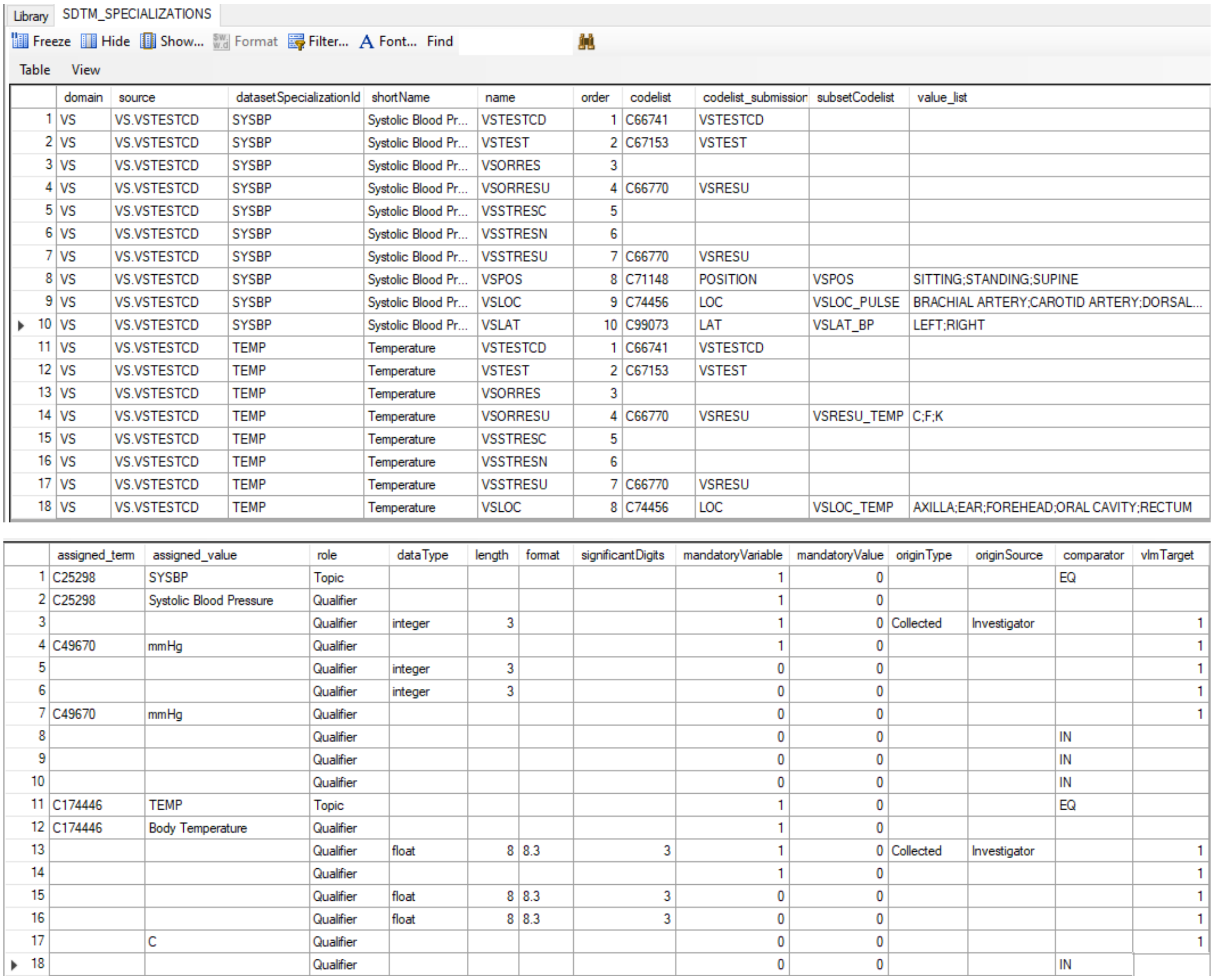
Tables that were extracted from JSON can be joined with the following SAS code:

```
proc sql;
  create table out.sdtm_specializations
    as select
        root.datasetSpecializationId
      , root.domain
      , root.source
      , root.shortName
      , var.name
      , var.ordinal_variables as order
      , var.isNonStandard
      , varcl.conceptId as codelist
      , varcl.submissionValue as codelist_submission_value
      , varvl._valueList as value_list
      , varat.conceptId as assigned_term
      , varat.value as assigned_value
      , var.subsetCodelist
      , var.role
      , var.dataType
      , var.length
      , var.format
      , var.significantDigits
      , var.mandatoryVariable
      , var.mandatoryValue
      , var.originType
      , var.originSource
      , var.comparator
      , var.vlmTarget
    from
      work.root root
      join work.variables var
    on (var.ordinal_root=root.ordinal_root)
      left join work.variables_codelist varcl
    on (varcl.ordinal_variables=var.ordinal_variables)
      left join work.variables_assignedterm varat
    on (varat.ordinal_variables=var.ordinal_variables)
      left join work.variables_valuelist varvl
    on (varvl.ordinal_variables=var.ordinal_variables)
    order by datasetSpecializationId, order
    ;
  ;
quit;
```

Display 8 shows part of the merged dataset with metadata for all SDTM Dataset Specializations, showing the Systolic Blood Pressure (SYSBP) and Temperature (TEMP) specializations.

Using this dataset, we can create Value Level Metadata and codelist metadata, which can then be used as building blocks to create a Define-XML document.

Display 8. Dataset with metadata for all SDTM Dataset Specializations



	domain	source	datasetSpecializationId	shortName	name	order	codelist	codelist_submission	subsetCodelist	value_list
1	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSTESTCD	1	C66741	VSTESTCD		
2	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSTEST	2	C67153	VSTEST		
3	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSORRES	3				
4	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSORRESU	4	C66770	VSRESU		
5	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSSTRESC	5				
6	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSSTRESN	6				
7	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSSTRESU	7	C66770	VSRESU		
8	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSPOS	8	C71148	POSITION	VSPOS	SITTING;STANDING;SUPINE
9	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSLOC	9	C74456	LOC	VSLOC_PULSE	BRACHIAL ARTERY;CAROTID ARTERY;DORSAL...
▶ 10	VS	VS.VSTESTCD	SYSBP	Systolic Blood Pr...	VSLAT	10	C99073	LAT	VSLAT_BP	LEFT;RIGHT
11	VS	VS.VSTESTCD	TEMP	Temperature	VSTESTCD	1	C66741	VSTESTCD		
12	VS	VS.VSTESTCD	TEMP	Temperature	VSTEST	2	C67153	VSTEST		
13	VS	VS.VSTESTCD	TEMP	Temperature	VSORRES	3				
14	VS	VS.VSTESTCD	TEMP	Temperature	VSORRESU	4	C66770	VSRESU	VSRESU_TEMP	C;F;K
15	VS	VS.VSTESTCD	TEMP	Temperature	VSSTRESC	5				
16	VS	VS.VSTESTCD	TEMP	Temperature	VSSTRESN	6				
17	VS	VS.VSTESTCD	TEMP	Temperature	VSSTRESU	7	C66770	VSRESU		
18	VS	VS.VSTESTCD	TEMP	Temperature	VSLOC	8	C74456	LOC	VSLOC_TEMP	AXILLA;EAR;FOREHEAD;ORAL CAVITY;RECTUM

	assigned_term	assigned_value	role	dataType	length	format	significantDigits	mandatoryVariable	mandatoryValue	originType	originSource	comparator	vlmTarget
1	C25298	SYSBP	Topic					1	0			EQ	
2	C25298	Systolic Blood Pressure	Qualifier					1	0				
3			Qualifier	integer	3			1	0	Collected	Investigator		1
4	C49670	mmHg	Qualifier					1	0				1
5			Qualifier	integer	3			0	0				1
6			Qualifier	integer	3			0	0				1
7	C49670	mmHg	Qualifier					0	0				1
8			Qualifier					0	0			IN	
9			Qualifier					0	0			IN	
10			Qualifier					0	0			IN	
11	C174446	TEMP	Topic					1	0			EQ	
12	C174446	Body Temperature	Qualifier					1	0				
13			Qualifier	float	8	8.3	3	1	0	Collected	Investigator		1
14			Qualifier					1	0				1
15			Qualifier	float	8	8.3	3	0	0				1
16			Qualifier	float	8	8.3	3	0	0				1
17		C	Qualifier					0	0				1
▶ 18			Qualifier					0	0			IN	

SAS OPENCST, FORMERLY KNOWN AS SAS CLINICAL STANDARDS TOOLKIT

The SAS Clinical Standards Toolkit (CST) was published by SAS in 2009 and supports clinical research activities by providing a framework of SAS macros based functionality to help ensure that standards are applied to clinical data and metadata. The SAS Clinical Standards Toolkit was designed as a modular system, able to adapt to new standards and versions of those standards [13][14]. CST focuses on the registration and use of standards defined by CDISC. SAS stopped development of CST in 2017. The last version supported by SAS, CST 1.7.2, primarily supported the following capabilities related to CDISC standards:

- Creating/reading CRT-DDS 1.0 (Define-XML v1.0)
- Creating/reading Define-XML v2.0 (including Analysis Results Metadata)[3]
- Creating/reading Dataset-XML
- Creating/reading ODM v1.3.0 and ODM v1.3.1
- Creating/reading CT-XML
- Registration of CDASH, SDTM, SEND and ADaM standards metadata

[3] The references section includes a paper about the Define-XML v2.0 implementation in CST [16].

In 2022 SAS released SAS Clinical Standards Toolkit under an Open Source license as SAS Clinical Standards Toolkit (openCST). The CDISC Open Source Alliance (COSA) has added openCST to its repository of Open Source projects [15]. This Open Source release is a direct port of the last production release 1.7.2 with minor modifications to adapt to new deployment architecture and has product documentation, installation instructions and details for contribution.

In the rest of this paper when we talk about SAS Clinical Standards Toolkit, we mean the Open Source version openCST.

Since releasing SAS Clinical Standards Toolkit under an Open Source license, the following additions have been made:

- Updated Define-XML v2.0 stylesheet.
- Support for ODM v1.3.2.
- Added CT-XML 1.2.0, to be able to support the latest NCI Controlled Terminology.
- Added full support for Define-XML v2.1.

SUPPORTING DEFINE-XML V2.1 WITH SAS OPENCST

Each SAS Clinical Standards Toolkit standard provides a SAS representation of the published source guidelines or source specification. The SAS representation is designed to serve as a model or template of the source specification. This representation helps with the following points:

- It supplies an implementation of data models and standards that are based on SAS.
- It enables you to use SAS routines to assess how well any user-defined set of data and metadata conforms to the standard.
- It enables you to use SAS code to read and derive files in other formats (for example, XML).

Since a Define-XML file does not have a 2-dimensional data structure, it is not a trivial task to translate this hierarchical file to SAS dataset with rows and columns. SAS has defined a relational data model that represents a Define-XML file.

The source metadata SAS datasets in SAS Clinical Standards Toolkit for Define-XML v2.1 are like the source metadata SAS datasets for Define-XML v2.0 [16]. For Define-XML v2.1 a source_standards dataset was added.

For Define-XML v2.1 the following source metadata SAS datasets are defined in SAS Clinical Standards Toolkit:

- **source_study**

  Metadata about the study, such as study name, study description and protocol name.

- **source_standards**

  Metadata about the data standards and terminology standards used in the study.

- **source_tables**

  Table metadata, such as name, domain, description (label), class, structure, purpose, keys, data location, comments, and document references.

- **source_columns**

  Column metadata, such as name, description (label), order number, datatype, length, codelist, origin type, origin source, significant digits, display format, derivation (algorithm), comments and document references.

- **source_values**

  Value level metadata (VLM), that has a condition defined in the WHERECLAUSE column.

Example WHERECLAUSE values are:

- (LBTESTCD EQ "BILI") AND (LBCAT EQ "CHEMISTRY") AND (LBSPEC EQ "BLOOD")
- VSTESTCD EQ “HEIGHT”
- PARAMCD IN ("ACITM01","ACITM02","ACITM03")
  PARAMCD NOTIN ("ACTOT")

The column which the value level metadata is attached to, is defined by the TABLE and COLUMN columns. Apart from the WHERECLAUSE column, this dataset has the same kind of metadata as the **source_columns** dataset.

- **source_codelists**

  Metadata related to Controlled Terminology, such as name, description, datatype, valid values, decodes, rank, order number, reference to NCI code, and external terminologies.

- **source_documents**:

  Metadata related to referenced documents, such as annotated CRF, reviewer guides or other supplemental documents. Records in this dataset can be linked to source_study, source_standards, source_tables, source_columns, source_values, source_codelists, or source_analysisresults datasets. Page numbers and named destinations in PDF files can be defined in this dataset as well.

- **source_analysisresults**:

  Metadata related to analysis displays and results: display identifier, display name, display description, result identifier, result description, analysis purpose and reason, parameter column, analysis variables, analysis datasets, selection criteria (WhereClause), Selection criteria for the records subject to analysis, result programming code and context, result documentation.

There are five key macros that are provided with the SAS Clinical Standards Toolkit that support the CDISC Define-XML v2.1 standard.

1. The **define_sourcetodefine** macro creates the SAS tables that contain the SAS representation of the Define-XML v2.1 file from the study source metadata SAS datasets. This macro, using SDTM, SEND or ADaM table and column metadata as its source, populates a subset of the Define-XML v2.1 datasets.
2. The **define_write** macro creates the Define-XML file from the SAS representation of the CDISC Define-XML v2.1 file.
3. The **cstutilxmlvalidate** macro validates that the XML file is syntactically correct according to the XML schema that is associated with the Define-XML v2.1 standard.
4. The **define_read** macro creates the SAS representation of the CDISC Define-XML 2.1 file by importing a Define-XML file.
5. The **define_createsrcmetafromdefine** macro creates the eight study source metadata SAS datasets from the SAS representation of the Define-XML v2.1 file.

These macros are called by driver programs that are responsible for properly setting up each openCST process to perform a task. Sample driver programs are provided with the Define-XML v2.1 standard in openCST related to the creation and import of the Define-XML file.

Here is the purpose of each of these driver programs:

- The **create_definexml_from_source.sas** driver program sets up the required metadata and SASReferences dataset for the sample study. It runs the **define_sourcetodefine** macro to create the SAS representation of the CDISC Define-XML v2.1 file from the sample study source metadata datasets. Then it runs the **define_write** and **cstutilxmlvalidate** macros to create the Define-XML v2.1 file. It also validates the XML syntax.

- The **create_sourcemetadata_from_definexml.sas** driver program sets up the required metadata and SASReferences dataset for the sample study. It runs the **cstutilxmlvalidate** and **define_read** macros to validates the XML syntax and reads the Define-XML file into the SAS dataset representation of the CDISC Define-XML 2.1 file. Then it runs the **define_createsrcmetafromdefine** macro to create the study source metadata datasets.

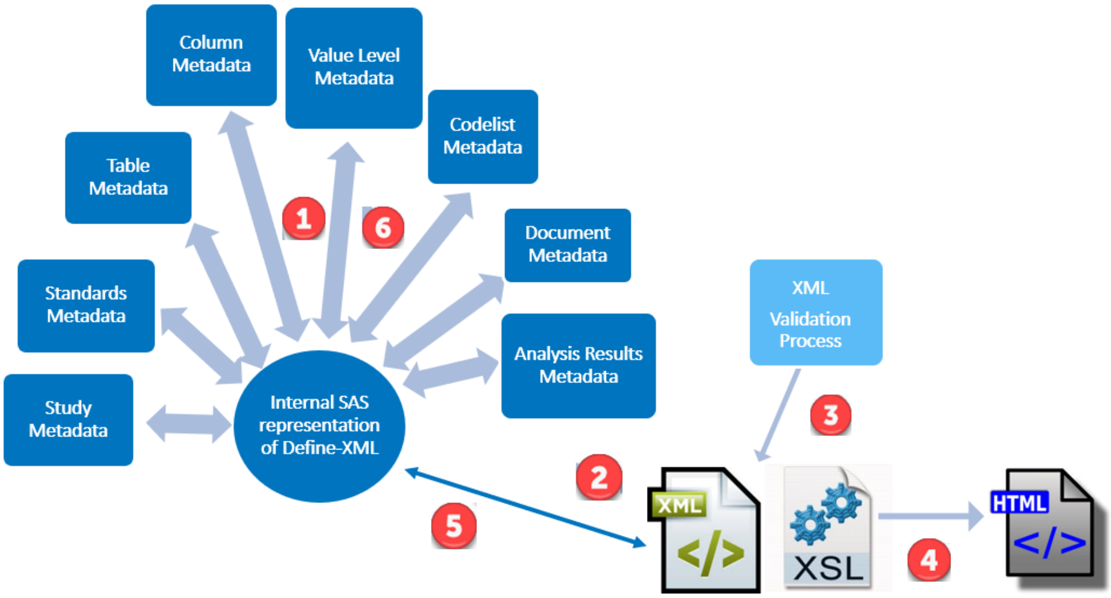
The sample implementation also includes other programs:

- **migrate_definexml_20_21.sas**
  Migrates study source metadata from the Define-XML v2.0 format to the Define-XML v2.1 format. The result may not be a complete representation of the study source metadata for Define-XML v2.1, since the Define-XML v2.0 study source metadata may not have everything that is required to create a complete and valid Define-XML v2.1 file. It does give a jump start for creating a Define-XML v2.1 fille when a Define-XML v2.0 file is available.
- **create_sourcemetadata_fromsaslib.sas**
  Creates initial study source metadata for Define-XML v2.1 from a library of SAS datasets. This is only an attempted approximation of study source metadata. No assumptions should be made that the result accurately and fully represents the study source metadata that is required to create a complete and valid Define-XML v2.1 file.
- **compare_metadata_sasdefine_xpt.sas**
  Compared the metadata from SAS XPT files with the SAS tables that contain the SAS representation of the Define-XML v2.1 file that describes the XPT files.
- **definexml_roundtrip_full_example.sas**
  Creates study source metadata from a Define-XML v2.1 file and uses that same metadata to create a Define-XML v2/1 file. The Define-XML file contains full coverage of the supported Define-XML v2.1 elements and attributes.

These driver programs are examples that are provided with the SAS Clinical Standards Toolkit. You can use these driver programs or create your own. The names of these driver programs are not important. However, the content is important and demonstrates how the various SAS Clinical Standards Toolkit framework macros are used to generate the required metadata files.

Display 9 illustrates the process for creating a Define-XML v2.1 in the SAS Clinical Standards Toolkit.

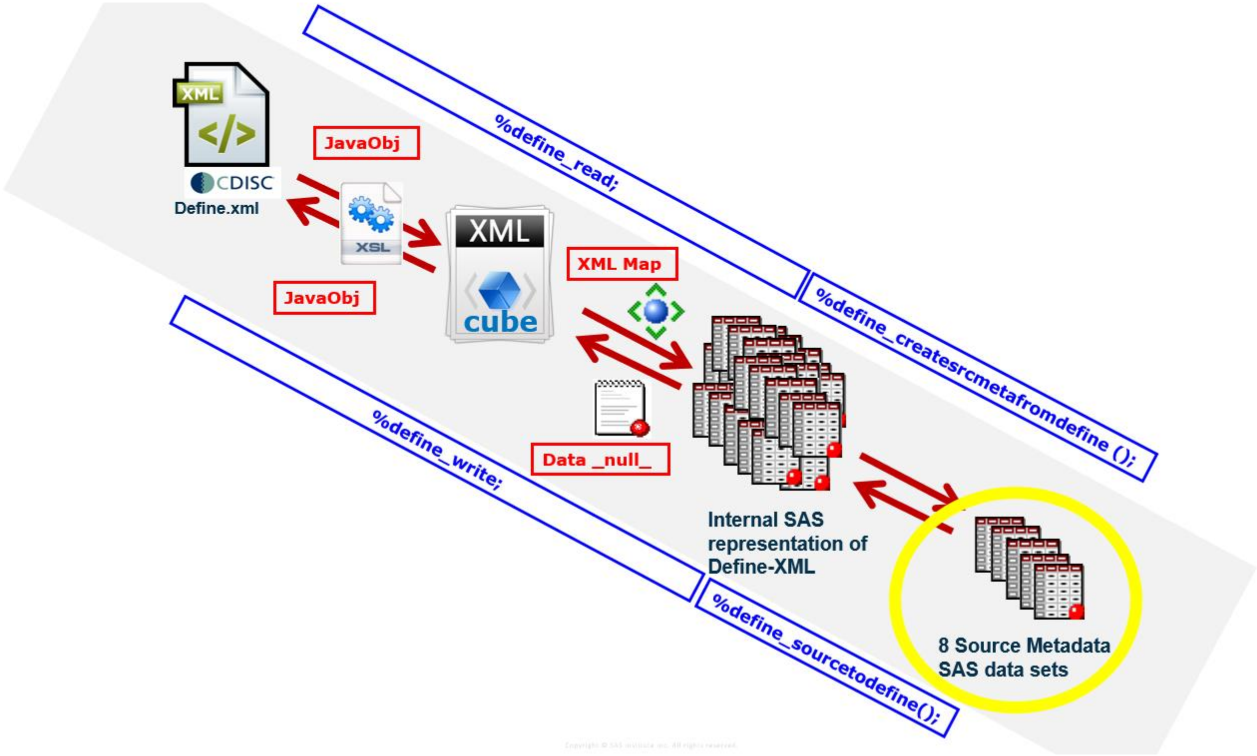
Display 9. The SAS macro process to create and read a Define-XML v2.1 document



The following steps are identified:

1. The **define_sourcetodefine** macro creates the tables for the SAS representation of the Define-XML v2.1 file from study source metadata datasets (source_*).
2. The **define_write** macro creates the Define-XML file from the SAS representation of the CDISC Define-XML v2.1 files.
3. The **cstutilxmlvalidate** macro validates the Define-XML file against the XML schema that is associated with the CDISC Define-XML v2.1 standard.
4. PROC XSL creates an HTML document from the Define-XML file and the XSL stylesheet that comes with the SAS Clinical Standards Toolkit.
5. The **define_read** macro creates the tables for the SAS representation of the Define-XML v2.1 files from a Define-XML file.
6. The **define_createsrcmetafromdefine** macro creates the study source metadata datasets (source_* datasets).

Display 10. SAS macros to support reading/writing Define-XML v2.1


XML
</>
CDISC
Define.xml
JavaObj
XSL
JavaObj
XML
cube
XML Map
%define_read;
%define_createsrcmetafromdefine ();
Data _null_
%define_write;
Internal SAS representation of Define-XML
8 Source Metadata SAS data sets
%define_sourcetodefine();

Display 10 illustrates how the openCST macros support the creation and import of a Define-XML v2.1 document.

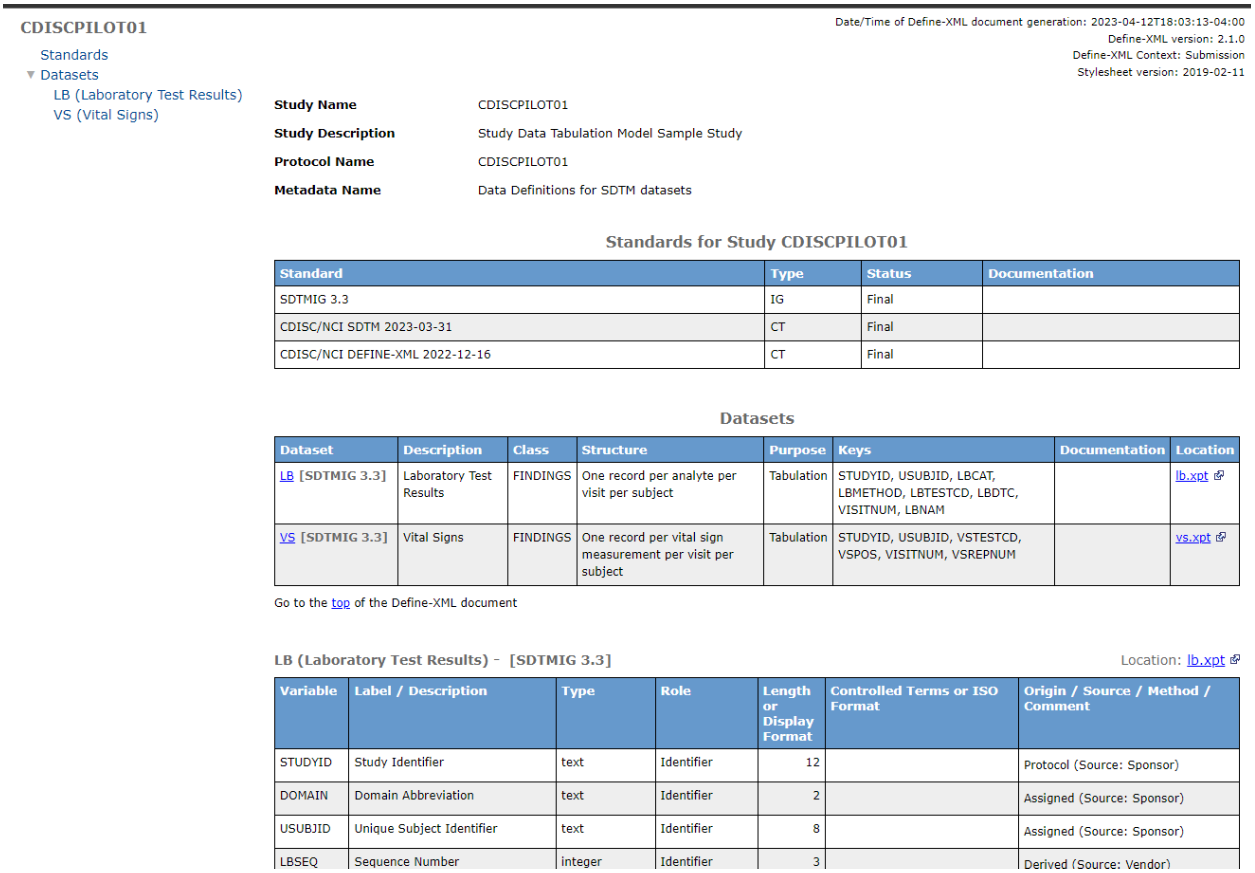
As mentioned, the Clinical Standard Toolkit uses XML schema validation to validate the resulting Define-XML v2.1 file against the XML schema. The Define-XML v2.1 file still needs to be validated against the conformance rules for Define-XML v2.1 [17]. Currently the Clinical Standard Toolkit does not support validation of a Define-XML v2.1 file

CREATING DEFINE-XML V2.1 FROM CDISC SDTM SPECIALIZATIONS

This section shows how one can use the SAS Clinical Standards Toolkit, PROC HTTP and basic SAS code to extract CDISC SDTM Dataset Specializations and NCI Controlled Terminology from the CDISC Library to add Value Level Metadata to limited domain metadata. This demonstrates then that CDISC SDTM Dataset Specializations and NCI Controlled Terminology provide building blocks that can be used to create Value Level Metadata for Define-XML v2.1. A similar process would also work for Define-XML v2.0.

For this exercise I used the Define-XML v2.1 document that was published with the SDTM Metadata Submission Guidelines v2.0 [18]. This Define-XML v2.1 document was stripped down to only contain the study metadata, standards metadata, and domain metadata for the LB (Laboratory Test Results) and VS (Vital Signs) domains. The resulting basic Define-XML v2.1 file did not contain any terminology information, or even comments or methods. A partial view of this basic Define-XML v2.1 file can be seen in Display 11.

Display 11. Basic Define-XML v2.1 document with study metadata, standards metadata, and basic table and column metadata for the LB and VS domains.



CDISCPILOT01

Standards
▼ Datasets
LB (Laboratory Test Results)
VS (Vital Signs)

Date/Time of Define-XML document generation: 2023-04-12T18:03:13-04:00
Define-XML version: 2.1.0
Define-XML Context: Submission
Stylesheet version: 2019-02-11

Study Name	CDISCPILOT01
Study Description	Study Data Tabulation Model Sample Study
Protocol Name	CDISCPILOT01
Metadata Name	Data Definitions for SDTM datasets

Standards for Study CDISCPILOT01

Standard	Type	Status	Documentation
SDTMIG 3.3	IG	Final	
CDISC/NCI SDTM 2023-03-31	CT	Final	
CDISC/NCI DEFINE-XML 2022-12-16	CT	Final	

Datasets

Dataset	Description	Class	Structure	Purpose	Keys	Documentation	Location
LB [SDTMIG 3.3]	Laboratory Test Results	FINDINGS	One record per analyte per visit per subject	Tabulation	STUDYID, USUBJID, LBCAT, LBMETHOD, LBTESTCD, LBDTC, VISITNUM, LBNAM		lb.xpt
VS [SDTMIG 3.3]	Vital Signs	FINDINGS	One record per vital sign measurement per visit per subject	Tabulation	STUDYID, USUBJID, VSTESTCD, VSPOS, VISITNUM, VSREPNUM		vs.xpt

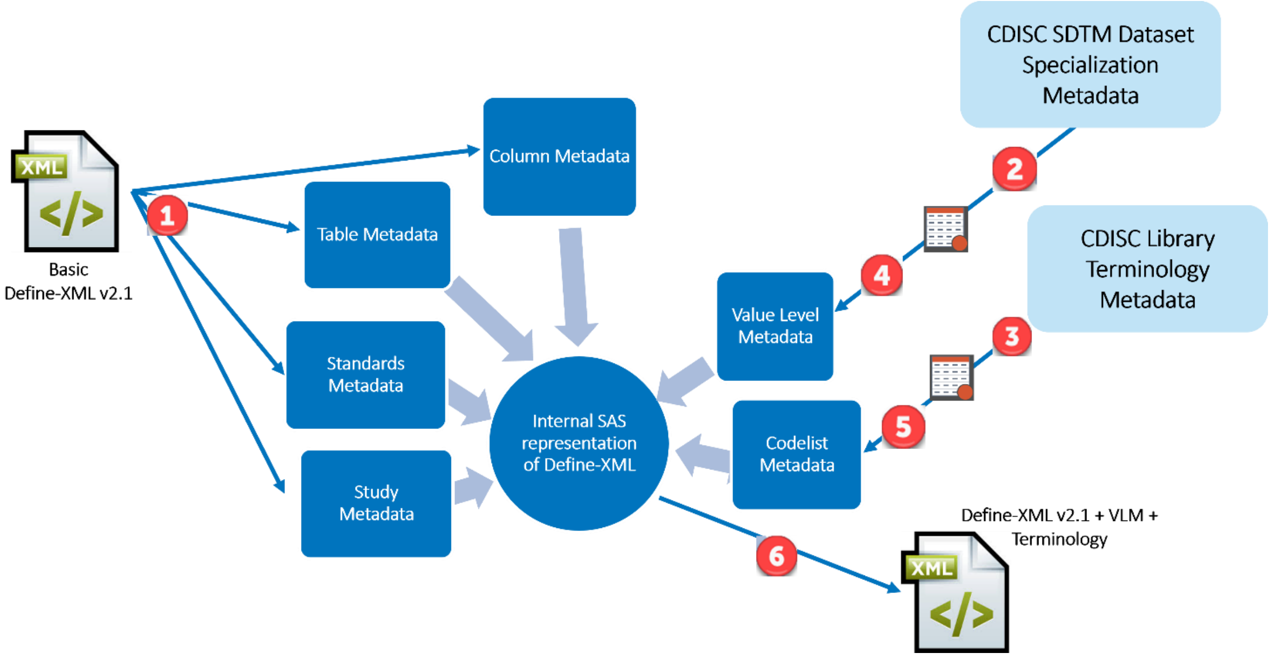
Go to the top of the Define-XML document

LB (Laboratory Test Results) - [SDTMIG 3.3]
Location: lb.xpt

Variable	Label / Description	Type	Role	Length or Display Format	Controlled Terms or ISO Format	Origin / Source / Method / Comment
STUDYID	Study Identifier	text	Identifier	12		Protocol (Source: Sponsor)
DOMAIN	Domain Abbreviation	text	Identifier	2		Assigned (Source: Sponsor)
USUBJID	Unique Subject Identifier	text	Identifier	8		Assigned (Source: Sponsor)
LBSEQ	Sequence Number	integer	Identifier	3		Derived (Source: Vendor)

Display 12 shows the process to get from a Define-XML v2.1 file with basic LB and VS domain metadata to a Define-XML v2.1 file with complete SDTM Dataset Specialization Value Level Metadata and Controlled Terminology metadata for the LB and VS domains.

Display 12. The SAS process to create a Define-XML v2.1 document with VLM from SDTM Dataset Specializations and Terminology



CDISC SDTM Dataset Specialization Metadata
XML
Basic Define-XML v2.1
Column Metadata
Table Metadata
Standards Metadata
Study Metadata
Internal SAS representation of Define-XML
Value Level Metadata
Codelist Metadata
CDISC Library Terminology Metadata
1
2
3
4
5
6
Define-XML v2.1 + VLM + Terminology
XML

The process in Display 12 uses six SAS program, each with a specific task.

1. **01_import_definexml.sas**
   Imports a basic Define-XML v2.1 document (Display 11) that contains study metadata, standards metadata, and basic LB and VS table and column metadata. The result of this import are study source metadata tables (source_study, source_standards, source_tables, and source_columns). It also creates the remaining source_* tables with zero records (source_values, source_codelists, and source_documents).
   The program uses the standard openCST Define-XML v2.1 import process.

2. **02_request_api_sdtm_latest.sas**
   Uses PROC HTTP to get the latest version of all SDTM Dataset Specializations metadata from the CDISC Library as JSON files. It then creates a SAS dataset that has the complete metadata for all the latest versions of the SDTM Dataset Specializations. This dataset has the same structure as the dataset in Display 8.

3. **03_request_api_ct.sas**
   Uses PROC HTTP to get the SDTM Terminology metadata from the CDISC Library as a JSON file. The version of the Terminology is determined by the standards metadata that was in the imported basic Define-XML v2.1 file in step 1. The program then converts the JSON file to a SAS dataset

4. **04_create_vlm_from_sdtm_specializations.sas**
   Creates the **source_values** dataset from the SDTM Dataset Specializations dataset that was created in step 2. Records with comparator = "EQ" or comparator = "IN" will be used to create WhereClause metadata. Records with vlmTarget = 1 will be used to create records in the source_values dataset with 'virtual variable' metadata. This will include creating references for both codelists and subset codelists.

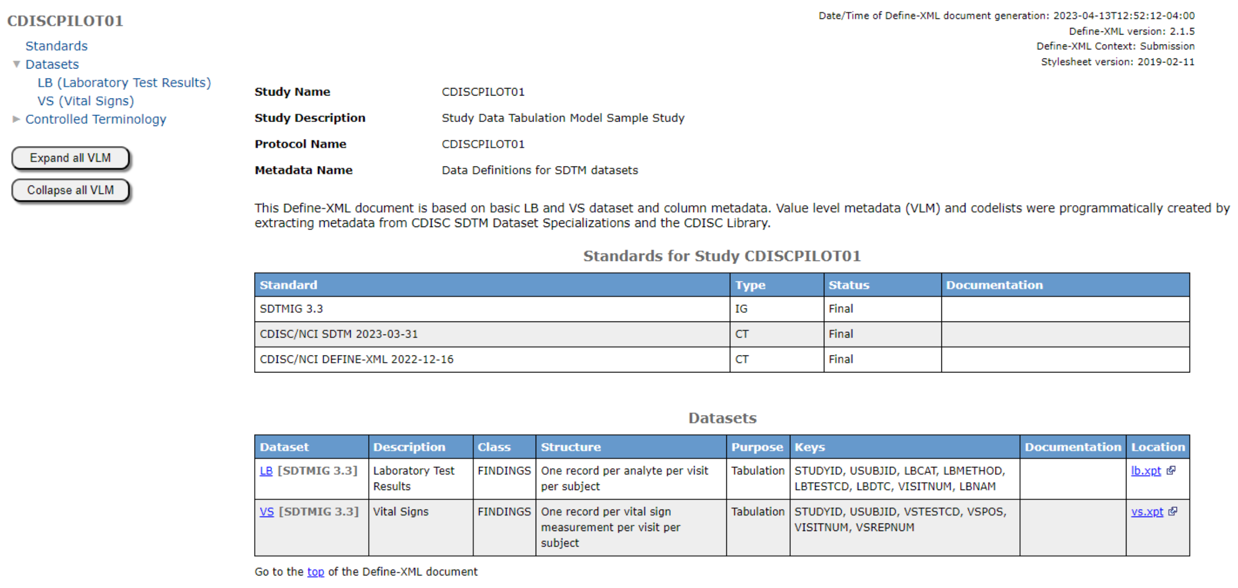
5. **05_create_ct_metadata.sas**
   Creates the **source_codelists** dataset from the SDTM Terminology metadata that was created in step 3. Only the codelists will be kept that are reference from the source_values dataset in step 4.

The program will also create subset codelists from valuelists that were part of the SDTM Dataset Specializations dataset that was created in step 2.

6. **06_create_definexml.sas**
Uses the source metadata tables that were created in the previous steps to create a Define-XML v2.1 file, including its HTML rendition.
The program uses the standard openCST Define-XML v2.1 creation process.

Display 13 shows screenshots of the HTML rendition of the 'enhanced' Define-XML v2.1 file.

Display 13. Define-XML v2.1 document with complete SDTM Dataset Specialization Value Level Metadata and Controlled Terminology metadata for the LB and VS domains.



Date/Time of Define-XML document generation: 2023-04-13T12:52:12-04:00
Define-XML version: 2.1.5
Define-XML Context: Submission
Stylesheet version: 2019-02-11

Study Name	CDISCPILOT01
Study Description	Study Data Tabulation Model Sample Study
Protocol Name	CDISCPILOT01
Metadata Name	Data Definitions for SDTM datasets

This Define-XML document is based on basic LB and VS dataset and column metadata. Value level metadata (VLM) and codelists were programmatically created by extracting metadata from CDISC SDTM Dataset Specializations and the CDISC Library.

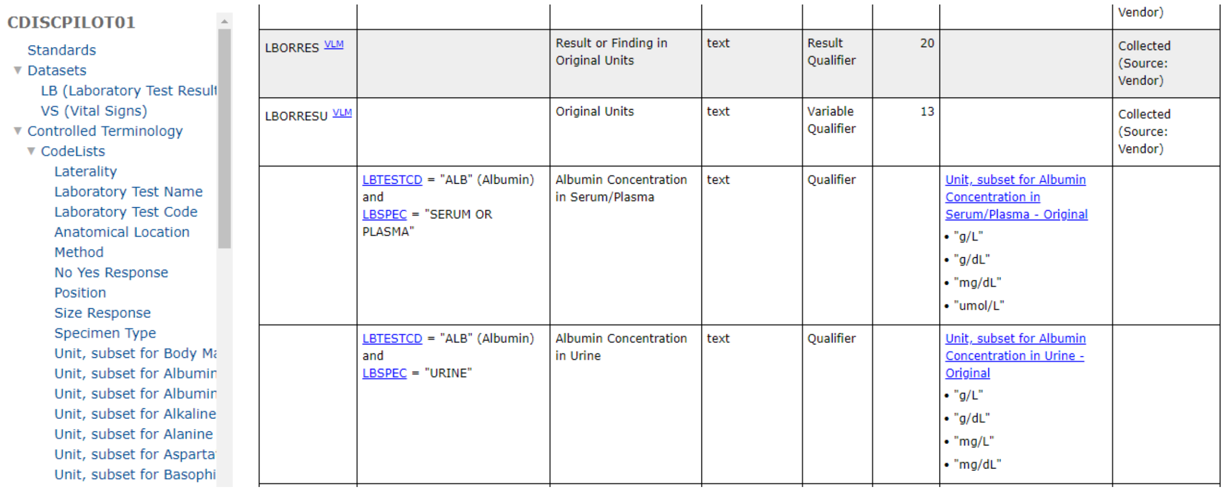
Standards for Study CDISCPILOT01

Standard	Type	Status	Documentation
SDTMIG 3.3	IG	Final	
CDISC/NCI SDTM 2023-03-31	CT	Final	
CDISC/NCI DEFINE-XML 2022-12-16	CT	Final	

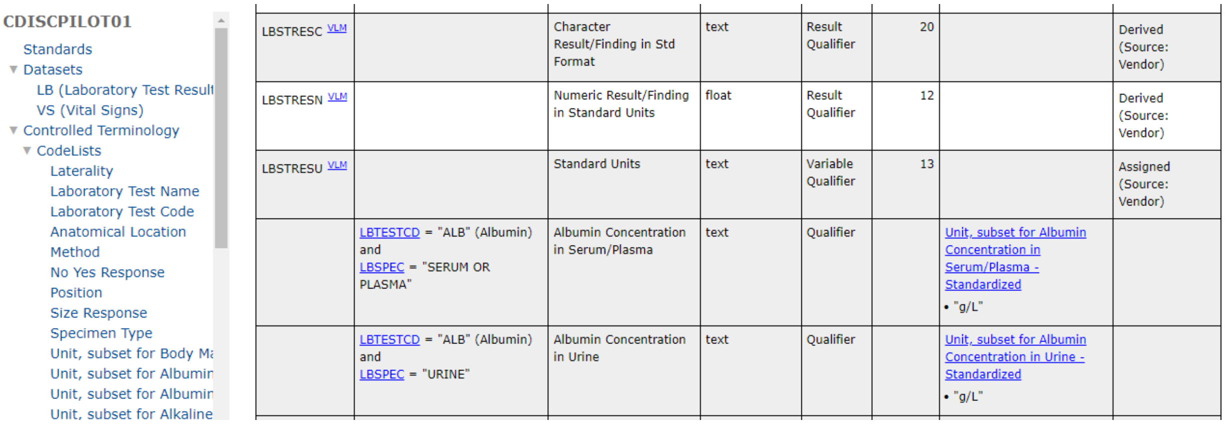
Datasets

Dataset	Description	Class	Structure	Purpose	Keys	Documentation	Location
LB [SDTMIG 3.3]	Laboratory Test Results	FINDINGS	One record per analyte per visit per subject	Tabulation	STUDYID, USUBJID, LBCAT, LBMETHOD, LBTESTCD, LBDTC, VISITNUM, LBNAM		lb.xpt
VS [SDTMIG 3.3]	Vital Signs	FINDINGS	One record per vital sign measurement per visit per subject	Tabulation	STUDYID, USUBJID, VSTESTCD, VSPOS, VISITNUM, VSREPNUM		vs.xpt

Go to the top of the Define-XML document



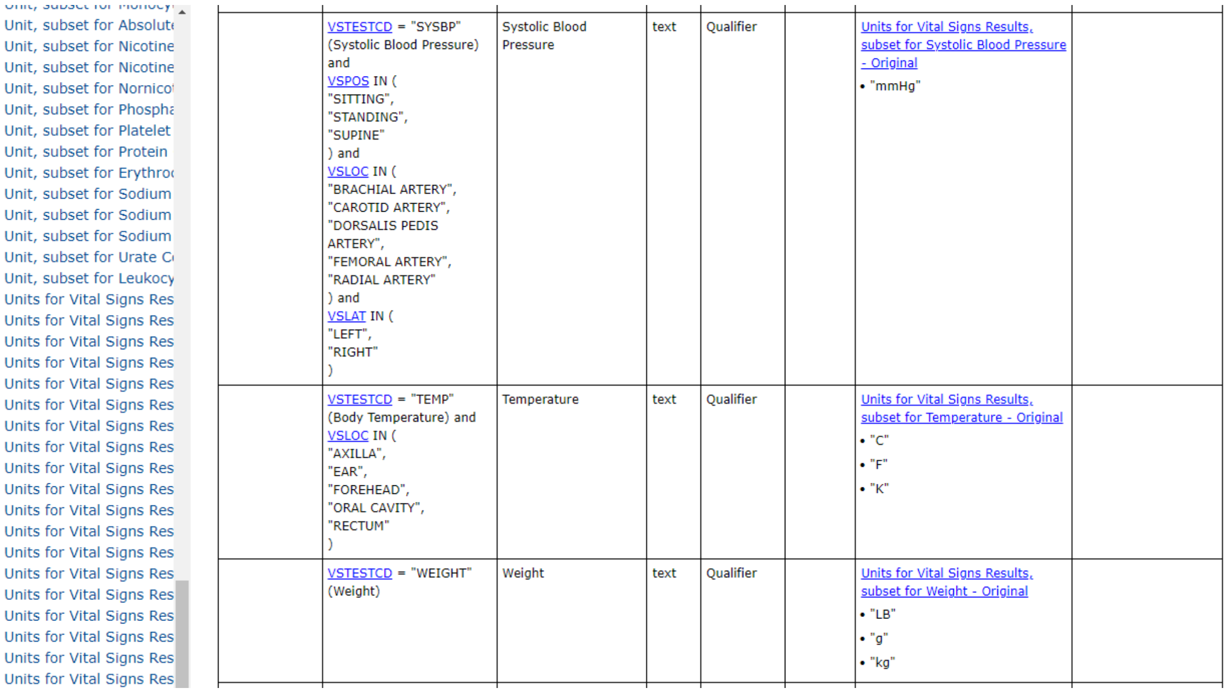
							Vendor)
LBORRES VLM		Result or Finding in Original Units	text	Result Qualifier	20		Collected (Source: Vendor)
LBORRESU VLM		Original Units	text	Variable Qualifier	13		Collected (Source: Vendor)
	LBTESTCD = "ALB" (Albumin) and LBSPEC = "SERUM OR PLASMA"	Albumin Concentration in Serum/Plasma	text	Qualifier		Unit, subset for Albumin Concentration in Serum/Plasma - Original • "g/L" • "g/dL" • "mg/dL" • "umol/L"	
	LBTESTCD = "ALB" (Albumin) and LBSPEC = "URINE"	Albumin Concentration in Urine	text	Qualifier		Unit, subset for Albumin Concentration in Urine - Original • "g/L" • "g/dL" • "mg/L" • "mg/dL"	



CDISCPILOT01

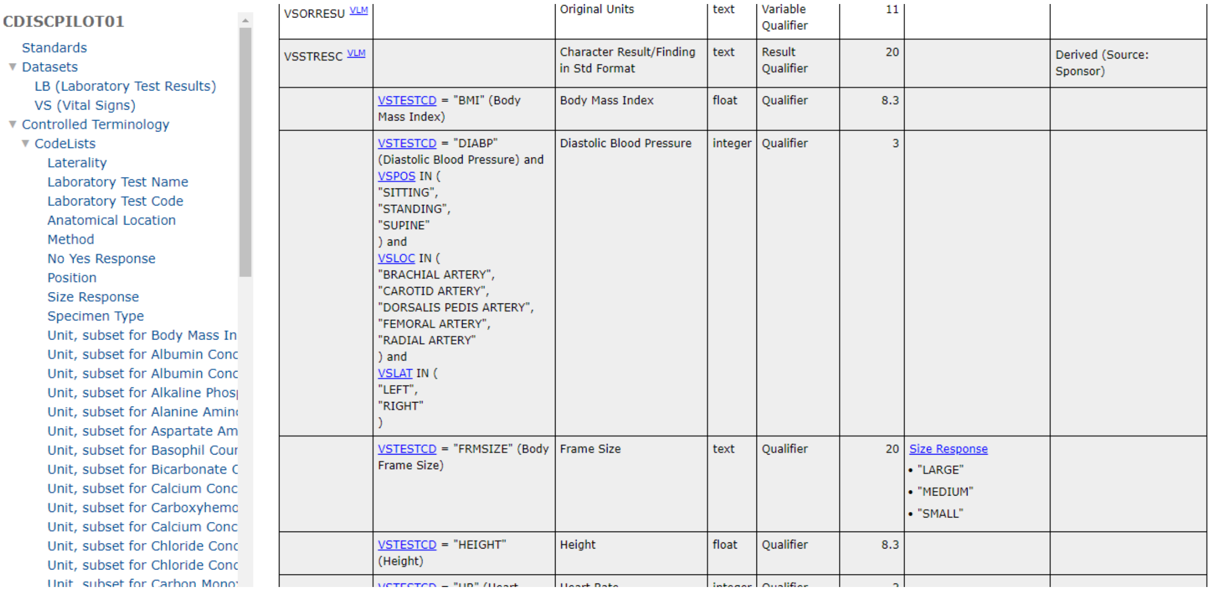
Standards
▼ Datasets
LB (Laboratory Test Result
VS (Vital Signs)
▼ Controlled Terminology
▼ CodeLists
Laterality
Laboratory Test Name
Laboratory Test Code
Anatomical Location
Method
No Yes Response
Position
Size Response
Specimen Type
Unit, subset for Body Ma
Unit, subset for Albumin
Unit, subset for Albumin
Unit, subset for Alkaline

LBSTRESC VLM		Character Result/Finding in Std Format	text	Result Qualifier	20		Derived (Source: Vendor)
LBSTRESN VLM		Numeric Result/Finding in Standard Units	float	Result Qualifier	12		Derived (Source: Vendor)
LBSTRESU VLM		Standard Units	text	Variable Qualifier	13		Assigned (Source: Vendor)
	LBTESTCD = "ALB" (Albumin) and LBSPEC = "SERUM OR PLASMA"	Albumin Concentration in Serum/Plasma	text	Qualifier		Unit, subset for Albumin Concentration in Serum/Plasma - Standardized • "g/L"	
	LBTESTCD = "ALB" (Albumin) and LBSPEC = "URINE"	Albumin Concentration in Urine	text	Qualifier		Unit, subset for Albumin Concentration in Urine - Standardized • "g/L"	



Unit, subset for Monocy
Unit, subset for Absolute
Unit, subset for Nicotine
Unit, subset for Nicotine
Unit, subset for Nornicot
Unit, subset for Phospha
Unit, subset for Platelet
Unit, subset for Protein
Unit, subset for Erythroc
Unit, subset for Sodium
Unit, subset for Sodium
Unit, subset for Sodium
Unit, subset for Urate C
Unit, subset for Leukocy
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res

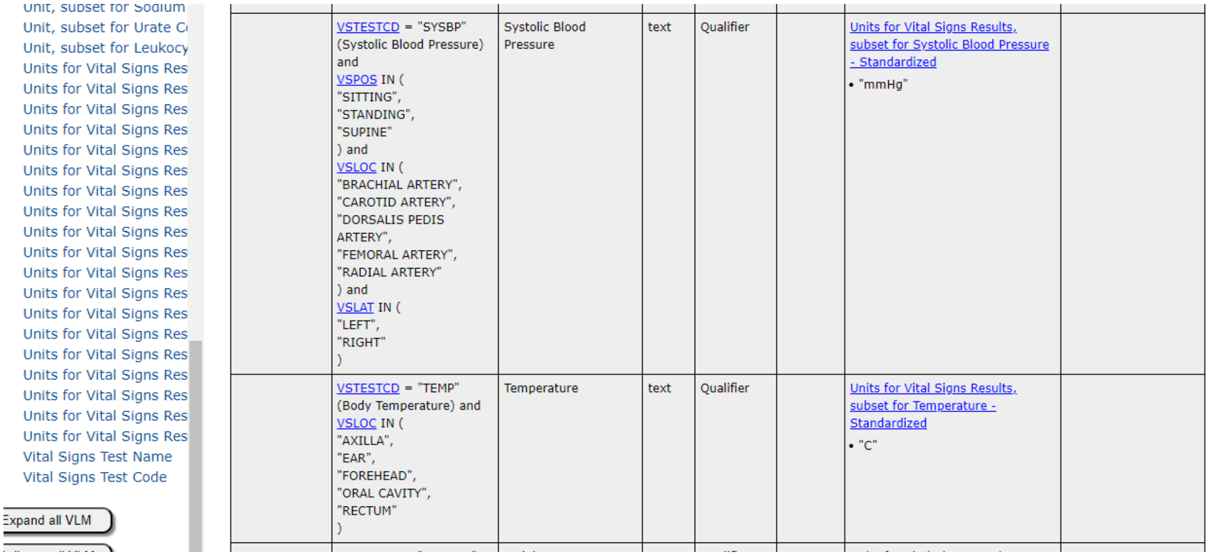
	VSTESTCD = "SYSBP" (Systolic Blood Pressure) and VSPOS IN ("SITTING", "STANDING", "SUPINE") and VSLOC IN ("BRACHIAL ARTERY", "CAROTID ARTERY", "DORSALIS PEDIS ARTERY", "FEMORAL ARTERY", "RADIAL ARTERY") and VSLAT IN ("LEFT", "RIGHT")	Systolic Blood Pressure	text	Qualifier		Units for Vital Signs Results, subset for Systolic Blood Pressure - Original • "mmHg"	
	VSTESTCD = "TEMP" (Body Temperature) and VSLOC IN ("AXILLA", "EAR", "FOREHEAD", "ORAL CAVITY", "RECTUM")	Temperature	text	Qualifier		Units for Vital Signs Results, subset for Temperature - Original • "C" • "F" • "K"	
	VSTESTCD = "WEIGHT" (Weight)	Weight	text	Qualifier		Units for Vital Signs Results, subset for Weight - Original • "LB" • "g" • "kg"	



CDISCPILOT01

Standards
▼ Datasets
LB (Laboratory Test Results)
VS (Vital Signs)
▼ Controlled Terminology
▼ CodeLists
Laterality
Laboratory Test Name
Laboratory Test Code
Anatomical Location
Method
No Yes Response
Position
Size Response
Specimen Type
Unit, subset for Body Mass In
Unit, subset for Albumin Conc
Unit, subset for Albumin Conc
Unit, subset for Alkaline Phos
Unit, subset for Alanine Amin
Unit, subset for Aspartate Am
Unit, subset for Basophil Cour
Unit, subset for Bicarbonate C
Unit, subset for Calcium Conc
Unit, subset for Carboxyhemo
Unit, subset for Calcium Conc
Unit, subset for Chloride Conc
Unit, subset for Chloride Conc
Unit, subset for Carbon Mono

VSORRESU VLM		Original Units	text	Variable Qualifier	11		
VSSTRESC VLM		Character Result/Finding in Std Format	text	Result Qualifier	20		Derived (Source: Sponsor)
	VSTESTCD = "BMI" (Body Mass Index)	Body Mass Index	float	Qualifier	8.3		
	VSTESTCD = "DIABP" (Diastolic Blood Pressure) and VSPOS IN ("SITTING", "STANDING", "SUPINE") and VSLOC IN ("BRACHIAL ARTERY", "CAROTID ARTERY", "DORSALIS PEDIS ARTERY", "FEMORAL ARTERY", "RADIAL ARTERY") and VSLAT IN ("LEFT", "RIGHT")	Diastolic Blood Pressure	integer	Qualifier	3		
	VSTESTCD = "FRMSIZE" (Body Frame Size)	Frame Size	text	Qualifier	20	Size Response • "LARGE" • "MEDIUM" • "SMALL"	
	VSTESTCD = "HEIGHT" (Height)	Height	float	Qualifier	8.3		
	VSTESTCD = "HR" (Heart	Heart Rate	integer	Qualifier	3		



Unit, subset for Sodium
Unit, subset for Urate C
Unit, subset for Leukocy
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Units for Vital Signs Res
Vital Signs Test Name
Vital Signs Test Code

Expand all VLM

	VSTESTCD = "SYSBP" (Systolic Blood Pressure) and VSPOS IN ("SITTING", "STANDING", "SUPINE") and VSLOC IN ("BRACHIAL ARTERY", "CAROTID ARTERY", "DORSALIS PEDIS ARTERY", "FEMORAL ARTERY", "RADIAL ARTERY") and VSLAT IN ("LEFT", "RIGHT")	Systolic Blood Pressure	text	Qualifier		Units for Vital Signs Results, subset for Systolic Blood Pressure - Standardized • "mmHg"	
	VSTESTCD = "TEMP" (Body Temperature) and VSLOC IN ("AXILLA", "EAR", "FOREHEAD", "ORAL CAVITY", "RECTUM")	Temperature	text	Qualifier		Units for Vital Signs Results, subset for Temperature - Standardized • "C"	

CONCLUSION

CDISC kicked off the Biomedical Concepts project in 2022, taking a pragmatic, iterative approach to create Biomedical Concepts (BCs) and representing them in the Foundational Standards as Dataset Specializations with Value Level Metadata definitions. This paper showed that the SDTM Dataset Specializations can be represented as Value Level Metadata definitions in Define-XML v2.1. These definitions contain detailed metadata, including Controlled Terminology subsets. The SDTM Dataset Specializations can be considered pre-configured building blocks, from which end-users can select and configure to build Value Level Metadata, which is an important part of the metadata for their Define-XML. The simplified implementation approach is easy to understand and allows for quick return on investment. SDTM dataset specializations are ready to be used as building blocks for Define-XML. This provides immediate benefits to SDTM programmers and opens the door to efficient programming and automation.

REFERENCES

[1] CDISC Biomedical Concepts. Available at https://www.cdisc.org/cdisc-biomedical-concepts

[2] Hume, Sam. (2020). CDISC 360: Using Biomedical Concept Metadata to Generate Case Report Forms and Dataset Definitions. Proceedings of PHUSE US Connect 2020.
Retrieved from https://www.lexjansen.com/phuse-us/2020/tt/TT06.pdf

[3] ISO/IEC. (2013). ISO/IEC 11179 Part 3: Registry metamodel and basic attributes. Available at https://en.wikipedia.org/wiki/ISO/IEC_11179

[4] Jansen, Lex. (2021). Extracting Data Standards Metadata and Controlled Terminology from the CDISC Library using SAS© with PROC LUA. Proceedings of PharmaSUG 2021.
Retrieved from https://www.lexjansen.com/pharmasug/2021/AD/PharmaSUG-2021-AD-168.pdf

[5] CDISC website. How-to-articles: Getting Started: Access to CDISC Library API using API Key Authentication. Available at
https://wiki.cdisc.org/display/LIBSUPRT/Getting+Started%3A+Access+to+CDISC+Library+API+using+API+Key+Authentication

[6] SAS® Institute Inc. (2017). “The HTTP Procedure” Base SAS Procedures Guide: Seventh Edition. Cary, NC: SAS® Institute Inc. Available at
https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/proc/n0bdg5vmrpyi7jn1pbgbje2atoov.htm

[7] Henry, Joseph (2020), REST Just Got Easy with SAS® and PROC HTTP.
Proceedings of SAS Global Forum 2020.
Retrieved from https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4426-2020.pdf

[9] Henry, Joseph (2019). The ABCs of the HTTP Procedure.
Proceedings of SAS Global Forum 2019.
Retrieved from https://support.sas.com/resources/papers/proceedings19/3232-2019.pdf

[10] SAS® Institute Inc . 2017. “LIBNAME Statement: JSON Engine” SAS® 9.4 Global Statements: Reference, Seventh Edition. Cary, NC: SAS® Institute Inc. Available at
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.2/lestmtsglobal/n1jfdetszx99ban1rl4zll6tej7j.htm

[11] Jansen, Lex (2021). Parsing JSON Files in SAS© Using PROC LUA.
Proceedings of PHUSE EU Connect 2021.
Retrieved from https://www.lexjansen.com/phuse/2021/ad/PRE_AD06.pdf

[12] Jansen, Lex (2022). Working with Dataset-JSON using SAS©. Proceedings of PharmaSUG 2022.
Retrieved from https://www.lexjansen.com/pharmasug/2022/AD/PharmaSUG-2022-AD-150.pdf

[13] SAS® Clinical Standards Toolkit
Available at the Internet Archive (accessed on April 19, 2023):
https://web.archive.org/web/20220127085021/support.sas.com/rnd/base/cdisc/cst/index.html

[14] Villiers, Pete (2009). A Toolkit for CDISC Implementation.
Proceedings of SAS Global Forum 2009.
Retrieved from https://support.sas.com/resources/papers/proceedings09/161-2009.pdf

[15] SAS Clinical Standards Toolkit (openCST) at the CDISC Open Source Alliance.
Available at https://cosa.cdisc.org/directory/open-sas-clinical-standards-toolkit

[16] Jansen, Lex (2017). Creating Define-XML version 2 including Analysis Results Metadata with the SAS® Clinical Standards Toolkit. Proceedings of PharmaSUG 2017.
Retrieved from https://www.lexjansen.com/pharmasug/2017/SS/PharmaSUG-2017-SS08.pdf

[17] CDISC (2021). Conformance Rules for Define-XML v2.1.
Available at https://www.cdisc.org/standards/foundational/define-xml/conformance-rules-define-xml-v2-1

[18] CDISC (2021). SDTM Metadata Submission Guidelines v2.0.
Available at https://www.cdisc.org/standards/foundational/sdtm/sdtm-metadata-submission-guidelines-v2-0

ACKNOWLEDGMENTS

The author would like to acknowledge the CDISC team for their work on the CDISC Biomedical Concepts, including Linda Lander, Anthony Chow, Sam Hume, Nic Haydel, Andrew Su, without whom this paper would not be possible.

CONTACT INFORMATION

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

Lex Jansen
Sr Director, Data Science Development, CDISC (contract through Lex Jansen Consulting LLC)
Email: lexjansen@gmail.com or ljansen@cdisc.org

All code used in this paper can be found at GitHub:

https://github.com/lexjansen/sas-papers/tree/master/pharmasug-2023