PharmaSUG 2019 - Paper DS-318

How to use SUPPQUAL for specifying natural key variables in define.xml?

Sergiy Sirichenko, Pinnacle 21

ABSTRACT

Define.xml must identify natural keys for each dataset to specify uniqueness for records and sort order. Sometimes standard SDTM/SEND variables are not enough to completely describe the structure of collected study data. In this presentation, we will show examples and recommend appropriate strategies for using SUPPQUAL variables or other common alternatives in the natural key. We will also provide guidance on how to document SUPPQUAL natural keys in define.xml.

INTRODUCTION

Reviewers depend on high quality metadata to understand the content of submitted study data. A list of key variables is one of the most important yet often overlooked parts of define.xml file. Standard SDTM or SEND variables sometimes lack sufficient detail to fully describe the structure of collected data. While SUPPQUAL datasets provide an opportunity to utilize non-standard variables, there is a lack of technical guidance on how to reference key variables from SUPPQUAL datasets.

KEY VARIABLES

CDISC Define-XML standard is limited to describing XML structure, but it does not provide enough implementation guidance like SDTM Implementation Guide (IG). PhUSE working groups tried to fix this deficiency by developing and publishing "Define-XML Version 2.0 Completion Guidelines" [1]. However, information about populating keys variables still remains insufficient.

Surprisingly, the best knowledge sources for key variables are SDTM and SEND Implementation Guides rather than Define-XML standard.

According to SDTM/SEND IGs, data definition file should describe each dataset by providing dataset's *natural key* structure.

"A natural key is a set of data (one or more columns of an entity) that uniquely identifies that entity and distinguishes it from any other row in the table. The advantage of natural keys is that they exist already; one does not need to introduce a new, "unnatural" value to the data schema." (SDTM IG 3.3 section 3.2.1.1)

Natural keys describe the physical structure of datasets and should be consistent with the description of dataset structure provided separately in define.xml file. Natural keys represent additional and more detailed information which is also machine-readable and can be used for automation.

For example, a structure of lb.xpt dataset may be defined generically as "One record per analyte per visit per subject", while key variables specify "analyte" as a combination of values in LBCAT, LBMETHOD and LBTESTCD variables (Display 1).

Tabulation Datasets for Study CDISC01 (SDTM-IG 3.1.2)

Dataset	Description	Structure	Keys
LB	Laboratory Tests Results	One record per analyte per visit per subject	STUDYID, USUBJID, LBCAT, LBMETHOD, LBTESTCD, LBDTC, VISITNUM

Display 1. Example of LB domain Structure and Keys from CDISC Define-XML 2.0 package

In most cases, key variables also represent a sorting order of records in a dataset.

By contrast, a *surrogate key* is a single-part, artificially established identifier for a record. For example, --SEQ variable. Surrogate keys are great for linking records across datasets such as from SUPPQUAL back to their parent domains or when using the RELREC dataset. However, surrogate key variables should not be used as keys for domains. The TS domain is one notable exception.

SUPPQUAL DATASETS

SDTM model has a special method to utilize non-standard variables using Supplemental Qualifier (SUPPQUAL) datasets. QNAM and QLABEL variables in SUPPQUAL datasets represent name and label of new non-standard variable, while QVAL stores its value. Linkage to parent domains is provided by RDOMAIN, USUBJID, IDVAR and IDVARVAL variables. In most cases, a surrogate key --SEQ is used to identify a parent record (e.g., IDVAR='AESEQ'). For more details about Supplemental Qualifier datasets, please see SDTM Model and Implementation Guide documentation.

SPECIFICATION OF KEY VARIABLES IN DEFINE.XML

In Define-XML specification, *ItemRef* and *ItemDef* elements describe variables. *ItemRef* provides information about variable in relation to a dataset, while *ItemDef* includes all additional information about variable. For example, *ItemRef* element specifies an order of variable within a dataset (*OrderNumber* attribute) or if a value of variable must be populated for all records (*Mandatory* attribute). *ItemDef* element specifies variable name, label, origin, codelist, etc.

The list of key variables is dataset-level information. Therefore, it is specified in *ItemRef* element by *KeySequence* attribute. Here is an example where STUDYID, USUBJID are declared as keys in Demographics domain:

```
  <!-- Dataset Definition (VS) -->
  <ItemGroupDef OID="IG.DM"
      Domain="DM"
      Name="DM"
      Repeating="No"
      IsReferenceData="No"
      SASDatasetName="DM"
      Purpose="Tabulation"
      def:Structure="One record per subject"
      def:Class="SPECIAL PURPOSE"
      def:ArchiveLocationID="LF.DM">
      <Description>
            <TranslatedText xml:lang="en">Demographics</TranslatedText>
      </Description>
      <ItemRef ItemOID="IT.DM.STUDYID" OrderNumber="1" Mandatory="Yes"
            KeySequence="1"/>
      <ItemRef ItemOID="IT.DM.DOMAIN" OrderNumber="2" Mandatory="Yes"/>
      <ItemRef ItemOID="IT.DM.USUBJID" OrderNumber="3" Mandatory="Yes"
            KeySequence="2" MethodOID="MT.DM.USUBJID"/>
      <ItemRef ItemOID="IT.DM.SUBJID" OrderNumber="4" Mandatory="Yes"/>
      <ItemRef ItemOID="IT.DM.RFSTDTC" OrderNumber="5" Mandatory="No"
            MethodOID="MT.DM.RFSTDTC"/>
      <ItemRef ItemOID="IT.DM.RFENDTC" OrderNumber="6" Mandatory="No"
            MethodOID="MT.DM.RFENDTC"/>
...
```

Information about non-standard variables in SUPPQUAL datasets is specified using Value Level Metadata (VLM) in define.xml file. Similar to how regular variables are listed for each domain

(*ItemGroupDef* element), new non-standard variables stored in SUPPQUAL datasets are listed in value level list (*def:ValueListDef* element). Their additional properies are specified in *ItemDef* element similar to regular variables.

Therefore, providing a reference to key variables in SUPPQUAL datasets is quite simple. You need to assign *KeySequence* attribute to specific value level item in value level list. Here is an example of non-standard key variable LBCLSIG from SUPPLB dataset for LB domain:

```
<def:ValueListDef OID="VL.SUPPLB.QVAL">
      <ItemRef ItemOID="IT.SUPPLB.QVAL.LBCLSIG"
            OrderNumber="1"
            Mandatory="No"
            MethodOID="MT.CLSIG"
            KeySequence="8">
      <def:WhereClauseRef WhereClauseOID="WC.SIPPLB.QNAM.LBCLSIG"/>
</ItemRef>
```

PRACTIAL RECOMMENDATIONS FOR TECHNICAL IMPLEMENTATION

Low utilization of non-standard variables leads to little business demand for supporting this functionality. For this reason, most current analysis tools and processes do not support or utilize key variable from SUPPQUAL datasets.

While the process of adding *KeySequence* attribute for value level item to specify keys from SUPPQUAL dataset looks very simple, there are several issues you need to understand and manage. The 3 major issues are:

1. Most Define.xml tools do not support keys from SUPPQUAL datasets
2. Outdated stylesheets do not show keys from SUPPQUAL datasets
3. New stylesheet from PhUSE utilizes specific attribute *Name* as display

ISSUE #1. MOST EXISTING DEFINE.XML TOOLS DO NOT SUPPORT KEYS FROM SUPPQUAL DATASETS

For example, while creating define.xml file with Pinnacle 21 Community 3.0 [2], the only option for users is to add the keys from SUPPQUAL datasets manually. To do this, you need to finalize your define.xml specs using the provided template and convert them into define.xml file. Then, you need to edit your file using any XML or text editor like XML Spy, Notepad+ (recommended option), or Notepad.

Open your define.xml file with your editor. Search for your SUPPQUAL variable in associated value list. You can look across all your value lists (search for "<def:ValueListRef"). Another option would be to search for name of the new non-standard variable name which is usually included in *ItemOID* value. For example, if you search for "REPNUM" text, you may find something like:

```
<ItemRef ItemOID="IT.SUPPVS.QVAL.REPNUM" …
```

or

```
<ItemRef ItemOID="IT.SUPPVS.QVAL.SUPPVS.QNAM.EQ.VSREPNUM-1091fc31"…
```

Add *KeySequence="NNN"* to *ItemRef* element, where *NNN* is the key sequence number:

```
<ItemRef ItemOID="IT.SUPPVS.QVAL.SUPPVS.QNAM.EQ.VSREPNUM-1091fc31"
      OrderNumber="1" Mandatory="No" MethodOID="MT.SUPPVS.QNAM.VSREPNUM"
      KeySequence="6">
      <def:WhereClauseRef WhereClauseOID="WC.SUPPAE.QNAM.EQ.TRTEMFL"/>
</ItemRef>
```

Save your define.xml and open in a browser to test your changes (Display 2).

Dataset	Description	Class	Structure	Keys	Location
VS	Vital Signs	FINDINGS	One record per vital sign measurement per time point per visit per subject	STUDYID, USUBJID, VSTESTCD, VISITNUM, VSTPTNUM, QNAM.VSREPNUM	vs.xpt

Display 2. Example of reference to VSREPNUM variable from SUPPVS dataset as Key for VS domain

If it does not work as expected, then most likely it's due to issue #2 and/or issue #3m below.

ISSUE # 2. OLD STYLESHEETS DO NOT SUPPORT KEYS FROM SUPPQUAL

Older stylesheets for define.xml file do not support key variables from SUPPQUAL datasets. For example, the most commonly used stylesheet is one provided in CDISC Define-XML 2.0 standard documentation. It does not show Key Sequence info in value level items (Display 3).

Dataset	Description	Class	Structure	Keys	Location
VS	Vital Signs	FINDINGS	One record per vital sign measurement per time point per visit per subject	STUDYID, USUBJID, VSTESTCD, VISITNUM, VSTPTNUM	vs.xpt

Display 3. Example of missing reference to VSREPNUM variable from SUPPAE dataset as Key for VS domain while using original stylesheet provided in Define-XML 2.0 standard package

One recently completed project by PhUSE was the development of an updated stylesheet for Define-XML 2.0. In 2018, PhUSE finalized and published it on the PhUSE wiki [3]. New stylesheet has many nice features including support for Key Variables from SUPPQUAL dataset. See package documentation for details. The industry has just started adopting the new stylesheet from PhUSE. For example, Pinnacle 21 Community 3.0 includes the new stylesheet, while previous versions of Pinnacle 21 Community (2.2.0 and older) utilize original stylesheet from documentation package of CDISC Define-XML 2.0 standard.

If you are still using an old stylesheet, download and employ the new stylesheet from PhUSE instead [3].

Note that name of the stylesheet file may be different compared to your existing stylesheet file. There are two options: 1) rename a new stylesheet according to file name of your current stylesheet or 2) edit define.xml with a reference to the new file.

Here is an example of option #2. The following code shows original reference to stylesheet on the second row of exported define.xml file from Pinnacle 21 Community 2.2.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="define2-0-0.xsl"?>
```

A file name of new stylesheet from PhUSE is “define2-0.xsl”. Edit an existing reference in your define.xml file accordingly:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="define2-0.xsl"?>
```

Test your define.xml file with the new stylesheet by opening in a browser. There is a high probability that you will encounter issue #3.

ISSUE # 3. STYLESHEET USES ATTRIBUTE NAME TO DISPLAY NEW VARIABLE FROM SUPPQUAL

Instead of a clear reference to the new variable from SUPPQUAL dataset based on QNAM value like “*QNAM.VSREPNUM*”, you may receive a long and confusing text like “*QNAM.QVAL.SUPPVS.QNAM.EQ.VSREPNUM-1091fc31*” or “*SUPPVS.QNAM.EQ.ce8eb9c250eaa8deec1334a597a941cea6741b51*” (Display 4)

Dataset	Description	Class	Structure	Keys	Location
VS	Vital Signs	FINDINGS	One record per vital sign measurement per time point per visit per subject	STUDYID, USUBJID, VSTESTCD, VISITNUM, VSTPTNUM, QNAM. SUPPVS.QNAM.EQ.ce8eb9c250eaa8deec1334a597a941cea6741b51	vs.xpt

Display 4. Example of confusing long reference to VSREPNUM variable from SUPPVS dataset

This reference is generated based on attribute *Name* from element *ItemDef*.

In Define-XML 2.0 standard, *ItemDef* element provides details about variable or value level item. Its attributes are the same for both variable and value level item. However, practical usage of some elements for value level items is sometimes quite challenging. For example, it is quite clear what should be a *Name* for a variable. In some cases, defining *Name* for value level items is also straightforward. For example, for common value level items of QVAL variable in SUPPQUAL datasets, *Name* attribute can be assigned based on a value of QNAM variable. The define.xml stylesheet assumes this approach to generate a reference in a browser. Unfortunately, it is not always true.

Complexity arises immediately outside the case of value level items in SUPPQUAL datasets. For example, we may have value level metadata in LB domain with conditions based on LBTESTCD values. So, it may seem like LBTESTCD values are a good candidate for values of *Name* attribute for value level items. However, value level metadata in LB domain may not be limited to a single variable (e.g., LBORRES). The same *WhereClause* condition based on LBTESTCD values may be applied to value level items of different variables (e.g., LBORRES, LBORRESU, LBSTRESN, LBSTRESU). Therefore, using LBTESTCD values for *Name* attribute will produce duplicate names for different value level items.

Other examples are 1) multiple conditions in *WhereClause* with reference to different variables (e.g., LBTESTCD, LBCAT, LBMETHOD or VSTESTCD, DM.COUNTRY) or 2) other than “EQUAL” condition (e.g., “IN”, “NOT IN”, etc.).

Such potential diversity of value level metadata creates a challenge for population of meaningful names for value level items. Therefore, historically most define.xml tools used some algorithms to generate unique *Name* attributes for each value level item. It may be based on random hash value like “SUPPVS.QNAM.EQ.ce8eb9c250eaa8deec1334a597a941cea6741b51” or more short and meaningful value like “SUPPVS.QNAM.EQ.VSREPNUM”.

A stylesheet takes these values and attaches them to prefix “QNAM.” to generate a reference in datasets table.

Until existing algorithms in define.xml tools are adjusted, you can edit your define.xml manually.

Similar to steps in Issue #1, open your define.xml file with your editor. However, now you need *ItemDef* element instead of *ItemRef*. Use *ItemOID* value from your already edited value level item to search related *ItemDef* element. Similar to the example provided in Issue #1, you should use text "IT.SUPPVS.QVAL.SUPPVS.QNAM.EQ.VSREPNUM-1091fc31", which is used as ID in both *ItemRef* and *ItemDef* elements.

Most likely you will find something like:

```
<ItemDef OID="IT.SUPPVS.QVAL.SUPPVS.QNAM.EQ.VSREPNUM-1091fc31"
      Name="QVAL.SUPPVS.QNAM.EQ.VSREPNUM-1091fc31"
      DataType="interger"
      Length="2">
<def:Origin Type="Derived"/>
</ItemDef>
```

Modify Name attribute according to QNAM value in *WhereClause* as a name of new variable:

```
<ItemDef OID="IT.SUPPVS.QVAL.SUPPVS.QNAM.EQ.VSREPNUM-1091fc31"
      Name="VSREPNUM"
      DataType="interger"
      Length="2">
<def:Origin Type="Derived"/>
</ItemDef>
```

Save your define.xml and open in a browser to test your changes (Display 5)

Dataset	Description	Class	Structure	Keys	Location
VS	Vital Signs	FINDINGS	One record per vital sign measurement per time point per visit per subject	STUDYID, USUBJID, VSTESTCD, VISITNUM, VSTPTNUM, QNAM.VSREPNUM	vs.xpt

Display 5. Example of reference to VSREPNUM variable from SUPPVS dataset as Key for VS domain

WHEN TO USE KEYS FROM SUPPQUAL DATASETS

SDTM mapping of collected study data is quite a challenging process, because it could be done in many ways depending on client's standards, integration needs, ease of analysis, personal preference, etc. Sometimes the same information may be mapped in different domains, different variables, or stored in SUPPQUAL dataset.

Here are common examples for utilizing key variables from SUPPQUAL datasets.

OUTDATED VERSION OF SDTM

New versions of SDTM introduced new standard variables. Recent SDTM 1.7 model (SDTM-IG 3.3) has 30 new variables. Many of them can be utilized as key variables. For example, --REPNUM (Repetition Number) may represent a common case when multiple measurements are taken during assessment of Vital Signs. However, if study data utilized a previous version of SDTM, then the only option to use the new --REPNUM variable is by storing it in SUPPQUAL dataset.

Other historical examples of important information commonly utilized as key variables, but not a dedicated standard variable in old versions of SDTM are --LAT (Laterality) or --DIR (Directionality) introduced in SDTM 3.1.3 and not available in SDTM 3.1.2.

CDISC TAUG WITH PROVISIONAL VARIABLES

CDISC has already developed more than 30 Therapeutic Area User Guides (TAUG). Most of them include new *provisional* variables which are planned to be promoted to standard variables in new versions of SDTM. Some of them were already accepted as standard variables, some were not.

If you plan to utilize CDISC TAUGs [4] and their *provisional* variables, the only option is to submit these *non-standard* variables in SUPPQUAL datasets.

STUDY-SPECIFIC NON-STANDARD VARIABLES

CDISC SDTM model has experienced impressive growth. For example, the number of standard variables which can be used in Findings domains has expanded from 63 to 91 since the first version. Nevertheless, there is always a need for new types of information to be promoted into standards.

For example, SDTM has variable --LAT which means Laterality (e.g., “left” or “right”) within a subject body like “Left Leg” and “Right Leg”. In pre-clinical studies there is an additional concept of Laterality within an organ rather than within a subject body like “Left side of Liver” vs. “Right side of Liver”. Standard --LAT variable is not applicable for such cases. Use of new non-standard variable is expected.

Eventually, similar variables may be introduced as standard variables by SDTM model. Until it happens, the only option is to store them in SUPPQUAL datasets.

200-CHAR LIMITATION OF SAS® XPORT FORMAT

There is also an interesting case due to a 200-character limitation of variable length in SAS® XPORT v5 format. Some collected terms may be too long to fit into a single standard variable which is a key in a domain. For example, DVTERM (Protocol Deviation Term) is collected as free-text and may be really long. Therefore, collected value is split across standard DVTERM variable and additional new variables in SUPPDV dataset. Formally, Protocol Deviation Term info as a key variable should be represented by all its component variables like “DVTERM, QNAM.DVTERM1, QNAM.DVTERM2, QNAM.DVTERM3”.

WHEN TO NOT USE SUPPQUAL DATASET FOR KEYS

SDTM MODEL VS. PRACTICAL USE OF STUDY DATA

There are some standard SDTM variables which allow flexibility in their usage and sometimes may be utilized instead of SUPPQUAL datasets.

--SPID variable is a Sponsor-Defined Identifier. SDTM IG acknowledges that despite being Identifier, --SPID variable may also have meaning across domains.

For example, historically many users utilized --SPID variable to track repetitive measurements for Vital Signs assessments. Formally, it’s considered bad SDTM mapping practice because Repetitive Number is Record Qualifier information rather than Identifier. Therefore, CDISC team added new --REPNUM variable in SDTM-IG 3.3 to address this common challenge and prevent misuse of --SPID variable.

In practice, however, most users preferred to keep important information which may include domain keys in parent domain rather than in SUPPQUAL dataset. Though a formal violation of SDTM model, the benefits for keeping all core information together drives users toward this implementation.

CURRENT TOOLS SUPPORT FOR KEYS FROM SUPPQUAL DATASET

As we mentioned before, support for key variables from SUPPQUAL datasets is very limited. Also, we are not aware of any analysis tools or data warehousing systems which utilize domains keys in general.

Humans are still major consumers of this information. They need a good stylesheet for define.xml files which correctly displays machine-readable study metadata in a human-friendly way.

Today, communication of domain keys in define.xml file is limited to people and does not require machine-readable metadata for automation. The only exception is data validation process for duplicate records in datasets per study-specific definition in define.xml file.

CONCLUSION

The industry's increasing experience with SDTM and SEND implementation drives users to handle new challenges including the use of key variables from SUPPQUAL datasets. Recent development of a new define.xml stylesheet by PhUSE supports visual representation of key variables from SUPPQUAL datasets. While many define.xml tools are still catching up with its new functionality, users can easily edit their define.xml files to utilize key variables from SUPPQUAL datasets. While correct mapping of study data to SDTM model is important, users should be aware that today support for key variables from SUPPQUAL datasets is still limited.

REFERENCES

1. "PhUSE Define-XML Version 2.0 Completion Guidelines Version 1.0 Draft". 2018-11-28. Available at https://www.phuse.eu/documents//working-groups/deliverables/sphuse-define-xml-20-completion-guidelines-v10-draft-for-public-review-19881.pdf
2. Pinnacle 21 Community 3.0 Available at https://www.pinnacle21.com/downloads
3. "PhUSE Define-XML v2.0 Stylesheet Recommendations". 2018-12-07. Available at http://www.phusewiki.org/docs/WorkingGroups/XML/Define-XML%202.0%20Stylesheet%20Recommendations.zip
4. Michael Beers, "The Need for Therapeutic Area User Guide Implementation", PhUSE Connect US 2019. Available at https://www.lexjansen.com/phuse-us/2019/ds/DS07.pdf

CONTACT INFORMATION

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

Sergiy Sirichenko
Pinnacle 21 LLC
+1.570.817.6137
sergiy@pinnacle21.com