An Excel Framework to Convert Clinical Data to CDISC SDTM Leveraging SAS Technology

Sophie McCallum, Clinovo, Sunnyvale CA

Stephen Chan, Clinovo, Sunnyvale, CA

ABSTRACT

CDISC SDTM data is the standard format requested by the FDA for clinical trial data submission. SAS® is often used as an Extract, Transform, Load (ETL) tool to manually convert SAS extracts from a clinical database to SDTM format.

While this is a reasonable approach, it can quickly become tedious, error-prone, and time-consuming. In addition, the code is difficult to maintain.

As an alternative, Clinovo has developed an Excel framework that maps any clinical database to CDISC SDTM. All mapping definitions and rules written in Excel are then dynamically converted into a SAS program that can perform the SDTM transformation & validation with minimum programming through a series of SAS macros.

Our framework is highly extensible and we can rely on a function library of SAS macros for mapping standard data elements. Additionally, this framework supports natively CDISC controlled terminology.

This presentation will be very helpful for SAS programmers interested in:

- Learning new SAS programming skills

- Using Microsoft Excel to dynamically generate SAS code and improve code re-usability

- Converting clinical data to the CDISC SDTM standard with minimum programming

- Helping their organization build an effective CDISC toolkit

- Promoting CDISC benefits within their organization

INTRODUCTION

CDISC Architect is a SAS application that converts clinical data into CDISC SDTM with minimum programming. Designed for the pharmaceutical and biotechnology industry, CDISC Architect was designed to expedite and improve the quality of FDA submission. By automatically and systematically translating data into CDISC SDTM, the standard format for regulatory submission, companies are better prepared to improve their time to market.

The CDISC Architect mapping rules are stored in an Excel spreadsheet, which is parsed and translated into SAS code in real-time by a SAS-based transformation engine. The Excel spreadsheet acts as a set of mapping specifications. The configuration of the Excel spreadsheet - i.e. the mapping file - is straightforward for anyone with a technical background but is also accessible to a business analyst with no programming experience. The transformation engine may be called by a SAS program, a UNIX shell or a DOS command and integrates easily with any internal biometrics programming processes.

THE MAPPING FILE

GENERAL INFORMATION

The mapping file is an Excel file in XML format named *"mapping.xls"*. The mapping file contains several sheets: the FORMAT tab, the domain tabs (DM, EX, IE etc) and the SUPPQUAL domains. A description of these tabs is given below.
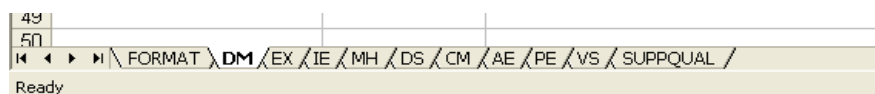
Figure 1. The different mapping file tabs

FORMAT TAB

All SAS formats can be used in the mapping file. However, you can also define custom formats and specify them in the FORMAT tab (Figure 2).

The FORMAT tab contains 3 columns:

- **format** – Defines the format name.

- **from** – Defines the entry value that you want to apply the format to.

- **tovalue** – Defines the value that will replace the entry value.

As an example, in the Figure 2, the first format is *$sev*. If you apply this format to a variable, the value "*1*" will be replaced by "*MILD*".

Figure 2. FORMAT tab

DOMAIN TABS

Each SDTM domain that will be mapped has to have its own tab (Figure 3). The name of the tab defines the SDTM domain dataset that is created (Data Management, Medical History etc).

Each domain tab contains 6 columns:

- **Dataset** – Specifies the source datasets that will be operated on to create the domain dataset as defined by the name of the tab.

- **Merge Key** (optional) – Defines the variables that will be used to merge the datasets that are specified in the Dataset column.

- **Join** (optional) – Specifies whether an inner or an outer join should be employed in merging the datasets with a merge key.

- **CDISC variable** – Specifies the CDISC variables that will be created.

- **Expression** – Provides the detail on the assignment statement of the SDTM variable in the CDISC variable column.

- **Comments** – It is merely for documentation purpose and is not used by CDISC Architect.

A	B	C	D	E
Dataset	Merge Key	Join	CDISC variable	Expression
txdetxde(where=(txdt2 eq .))	txdt1	O	STUDYID	study
			DOMAIN	&domain
			EXTRT	"GDC-0449"
			*_txad1	%CONVERTIF(_if_variable=missing(txad1),_if_value=1,_then_value=0,_else_value=^txad1)
			*_txad2	%CONVERTIF(_if_variable=missing(txad2),_if_value=1,_then_value=0,_else_value=^txad2)
			*_txad3	%CONVERTIF(_if_variable=missing(txad3),_if_value=1,_then_value=0,_else_value=^txad3)
			*_txad4	%CONVERTIF(_if_variable=missing(txad4),_if_value=1,_then_value=0,_else_value=^txad4)
			EXDOSE	_txad1*25+_txad2*125+_txad3*270+_txad4*150
			EXDOSU	"mg"
			EXDOSFRM	"CAPSULE, GELATIN COATED"
			EXDOSTOT	_txad1*25+_txad2*125+_txad3*270+_txad4*150
			EXROUTE	"ORAL"
			EXSTDTC	%DATEFLAG(_date=txdt1,_flag=txdtf1,_time=txadtm)
			EXENDTC	%DATEFLAG(_date=txdt1,_flag=txdtf1,_time=txadtm)
			EXSTDY	%STUDYDAY(_date=txdt1,_flag=txdtf1)
			EXENDY	%STUDYDAY(_date=txdt1,_flag=txdtf1)
			EPOCH	"TREATMENT PHASE"
			EXDOSFRQ	"Q24H"
txdetxde(where=(txdt1 ne .) rename=(txdt2=txdt1) keep=othrs othsp txrsae study invsite patnum txdt2)	txdt1	O	USUBJID	%GET_USUBJID()
txas3 txas4 ttxas1 ttxas2 txgrp)			USUBJID	%GET_USUBJID()
all			EXSEQ	%SEQUENCE()
			EXADJ	%DOSADJ(_adverse_event=txrsae,_other=othrs)
			VISITNUM	%VISITLOOKUP(_variable=visitnum,_key=visit txdt1)

Figure 3. Medical History domain tab

1.4 SUPPQUAL TAB

The Supplemental Qualifiers (SUPPQUAL) dataset is used to capture non-standard variables and their association to parent records in domains. It also allows capturing values for variables not presently included in the general observation class models. Because the CDISC SDTM does not allow the addition of new variables, it is necessary to represent the metadata and data for each non-standard variable/value combination in the SUPPQUAL dataset.

The SUPPQUAL tab defines the variables to be created that cannot be mapped to defined SDTM variables. Therefore, the SUPPQUAL variables metadata must be fully defined. This includes the domain name, the variable name, label, type, length, source …

Once SUPPQUAL variables have been defined and created from the definitions in the SUPPQUAL, they are computed directly within the DOMAIN tab. SUPPQUAL variables are differentiated from standard CDISC variables by adding the prefix '~' in the DOMAIN definitions.

Domain	VariableName	VariableLabel	Type	Len	Codes	Origin	Role	Comments	CORE	Drop_	SUPP	CO	VL	EVAL	DATE	REF
DM	XBRTHDTC	Modified Birth Date	Char	40		CRF	Result Qualifier	shh3925g	PERM		QNAM					
DM	_DEM	Change Flagging record	Char	500		MACR	Result Qualifier	shh3925g	PERM		QNAM					
DM	_ELCO	Change Flagging record	Char	500		MACR	Result Qualifier	shh3925g	PERM		QNAM					
DM	INDALK	Race	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
DM	ASIAN	Race	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
DM	BLACK	Race	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
DM	ISLAND	Race	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
DM	WHITE	Race	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
DM	RACENA	Race	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
VS	_VSDEVSDE	Change Flagging record	Char	500		MACR	Result Qualifier	shh3925g	PERM		QNAM					
AE	AEACTX	Action Taken with Study Treatment	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AETXNO	Other action taken	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AETXMD	Other action taken	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AETXPR	Other action taken	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AETXDC	Other action taken	Char	48		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AEHDT	Hospitalization Administration Date	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	PG	Page Number	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	LN	AE Line Number on CRF	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	PGSEQ	Page Sequence	Char	2		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	DTHDT	Death Date	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	DTHAUT	Autopsy Performed	Char	7		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	_AEDEAEDE	Change Flagging record	Char	500		MACR	Result Qualifier	shh3925g	PERM		QNAM					
AE	AENONE	No Other Suspected Adverse Event	Char	20		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AESDX	Disease Under Study	Char	20		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AECDX	Concurrent Illness	Char	20		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AECMD	Concomitant Medication (Non-Chemo)	Char	20		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AEPROC	Procedure	Char	20		CRF	Result Qualifier	shh3925g	PERM		QNAM					
AE	AEXP	Expedited Adverse Event	Char	20		CRF	Result Qualifier	shh3925g	PERM		QNAM					
IE	DVDES	Informed Consent Date	Char	200		CRF	Result Qualifier	shh3925g	PERM		QNAM					
IE	APRDT	Derivation Approval Date	Char	19		CRF	Result Qualifier	shh3925g	PERM		QNAM					
IE	APR	Derivation was approved	Char	2		CRF	Result Qualifier	shh3925g	PERM		QNAM					
IE	ICDT	Informed Consent Date	Char	19		CRF	Result Qualifier	shh3925g	PERM		QNAM					
IE	_ELDEELDE	Change Flagging record	Char	500		MACR	Result Qualifier	shh3925g	PERM		QNAM					
IE	_ELCO	Change Flagging record	Char	500		MACR	Result Qualifier	shh3925g	PERM		QNAM					
EX	TXGRP	Study Stage	Char	200		CRF	Result Qualifier	shh3925g	PERM		QNAM					
EX	TXAS1	Total No. of 25 mg Capsules Perscribed	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
EX	TXAS2	Total No. of 125 mg Capsules Perscribed	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
EX	TXAS3	Total No. of 270 mg Capsules Perscribed	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
EX	TXAS4	Total No. of 150 mg Capsules Perscribed	Num	8		CRF	Result Qualifier	shh3925g	PERM		QNAM					
EX	TTXAS1	Initial Assigned Total Daily Dose	Char	200		CRF	Result Qualifier	shh3925g	PERM		QNAM					
EX	TTXAS2	Modifed Assigned Total Daily Dose	Char	200		CRF	Result Qualifier	shh3925g	PERM		QNAM					

Figure 4. SUPPQUAL tab

CDISC ARCHITECT INTERPRETATION OF THE MAPPING FILE

GENERAL INFORMATION

The CDISC Architect interprets the mapping file as instructions to map source datasets to SDTM.
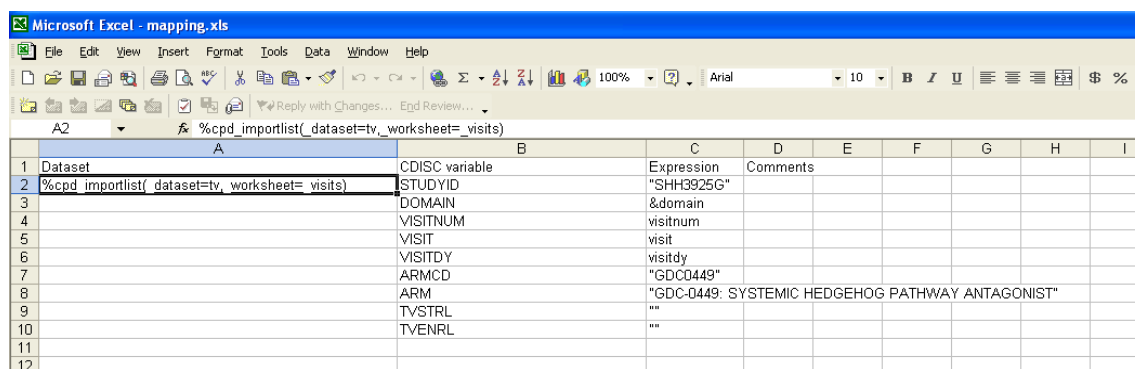
Each domain tab contains instructions to create the dataset named by the tab, for example, DM. The source datasets specified in the tab may be merged and/or 'set' (aggregated) together. The SDTM variables listed in the CDISC variable column are assigned values as specified in the Expression column. The Expression column may contain any valid SAS assignment or SAS macro from the function library that perform an assignment.

DATASET COLUMN

The Dataset column indicates which source dataset to use. Before the CDISC Architect begins operating on the datasets listed in the Dataset column, it checks if the "*all*" or "*all*(*stack*)" is listed as a dataset. "*all*" instructs CDISC Architect to perform the variable computation only after all previous datasets have been merged. "*all*(*stack*)" instructs CDISC Architect that all previous datasets will be stacked (SAS command SET) together, and no merging will be done unless a merge key specifies which variable to use.

The dataset column may use a macro that creates a dataset to be operated on by CDISC Architect.

For instance, in Figure 5, the *%cpd_importlist* macro creates a dataset TV. This feature allows the programmer to perform complex data operations on data using SAS that CDISC Architect mapping interface is not able to support. The *_dataset* parameter is used to create the corresponding dataset in the *tempdata* directory of CDISC Architect for troubleshooting. This mechanism is used for very complex mapping and should be minimized to increase the readability of the mapping rules.

Figure 5. Example of dataset creation by a macro

MERGE KEY COLUMN

The CDISC Architect checks the merge key column to see if all merge key variables are the same for the listed datasets in the Dataset column. If merge keys are different, the CDISC Architect will only merge datasets with identical merge keys.

The CDISC Architect logic works for only up to two levels of merge. In the first pass, CDISC Architect will merge all datasets separately that share the same merge key. This also means that a mapping file with a single merge key will be incorrect and will not validate. If two levels of merge are not sufficient for complex mapping, then it is advised to use a macro to be called from the dataset column.
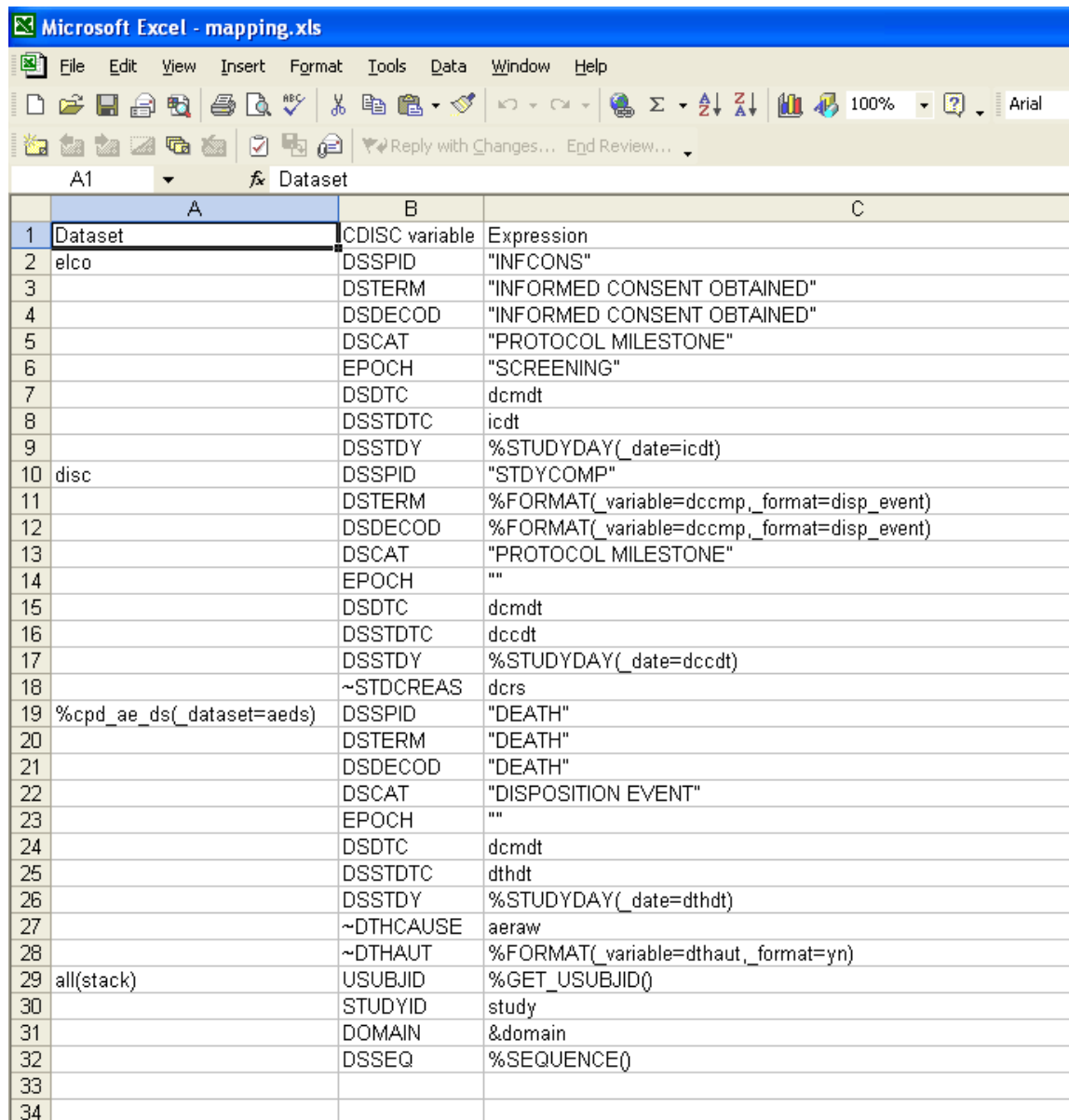
CDISC VARIABLE COLUMN

This column indicates the name of the CDISC variable to be created. If the CDISC variable is prefixed by the '~' character it means that the variable is defined in SUPPQUAL.

The symbol '*' defines a temporary variable to be created within the dataset group. The temporary variable is then used to create other CDISC variables that are created after the temporary variable.

5

EXPRESSION COLUMN

The Expression column can contain a:

- **String** – For example in Figure 8, the value for the CDISC variable DSSPID will be the constant string "*INFCONS*".

- **Dataset variable** – For example, the value for the CDISC variable DSDTC will be the value of the variable *dcmdt* from the source dataset *elco*.

- **Call** – For example, a call to a macro from the function library. The CDISC Architect can use macros that have already been programmed for a specific purpose. For instance, the CDISC variable DSTERM for the source dataset *disc* has for value the result of the call of the macro %FORMAT. This macro has for parameter the variable *dccmp* from the *disc* dataset and the format *disp_event*.

	A	B	C
		fx Dataset	
	A	B	C
1	Dataset	CDISC variable	Expression
2	elco	DSSPID	"INFCONS"
3		DSTERM	"INFORMED CONSENT OBTAINED"
4		DSDECOD	"INFORMED CONSENT OBTAINED"
5		DSCAT	"PROTOCOL MILESTONE"
6		EPOCH	"SCREENING"
7		DSDTC	dcmdt
8		DSSTDTC	icdt
9		DSSTDY	%STUDYDAY(_date=icdt)
10	disc	DSSPID	"STDYCOMP"
11		DSTERM	%FORMAT(_variable=dccmp,_format=disp_event)
12		DSDECOD	%FORMAT(_variable=dccmp,_format=disp_event)
13		DSCAT	"PROTOCOL MILESTONE"
14		EPOCH	""
15		DSDTC	dcmdt
16		DSSTDTC	dccdt
17		DSSTDY	%STUDYDAY(_date=dccdt)
18		~STDCREAS	dcrs
19	%cpd_ae_ds(_dataset=aeds)	DSSPID	"DEATH"
20		DSTERM	"DEATH"
21		DSDECOD	"DEATH"
22		DSCAT	"DISPOSITION EVENT"
23		EPOCH	""
24		DSDTC	dcmdt
25		DSSTDTC	dthdt
26		DSSTDY	%STUDYDAY(_date=dthdt)
27		~DTHCAUSE	aeraw
28		~DTHAUT	%FORMAT(_variable=dthaut,_format=yn)
29	all(stack)	USUBJID	%GET_USUBJID()
30		STUDYID	study
31		DOMAIN	&domain
32		DSSEQ	%SEQUENCE()
33			
34			

Figure 6. Expression column for the domain DS

JOIN COLUMN

The JOIN column is optional and is usually placed on the right of the 'Merge Key' column. It can take the values 'I' (inner join) or 'O' (outer join). With this column, you can specify the type of join you want to use. If 'I' is indicated, the IN option is employed on the dataset. If 'O' is indicated, the IN option is not employed on the dataset. If the join column is not present then the IN option is employed on both datasets. There are four possibilities which are illustrated below:

```
data data3;           *(I,I);
            merge data1(in=indata1) data2(in=indata2);
            by var1;
            if indata1 and indata2;
run;

data data3;           *(O,O);
            merge data1 data2;
            by var1;
run;

data data3;           *(I,O);
            merge data1(in=indata1) data2;
            by var1;
            if indata1;
run;

data data3;*(O,I);
            merge data1 data2(in=indata2);
            by var1;
            if indata2;
run;
```

EXAMPLES OF DATASET MANIPULATION

CREATION FROM A SINGLE DATASET

If only one source dataset is available in the Dataset column, the domain will be created from a single dataset.

Figure 7. AE domain dataset creation from a single source dataset

For example, in Figure 7, the CDISC Architect creates a dataset AE in a single data step from the *aedeaede* dataset. The dataset *aedeaede* is set and each CDISC variable is assigned to the expression as specified in the tab. The code would look as follows:

```
data ae;

    set aedeaede;

    studyid=study;

    domain=&domain;

    usubjid=%concatenate(_variables=study invsite patnum);

    …

run;
```

CREATION BY STACKING MULTIPLE DATASETS FROM A SOURCE DATASET

In the figure below, we used several datasets in the Dataset column and used the term "*all*(*stack*)". The CDISC Architect creates a dataset VS by stacking multiple datasets created from the *vsdebsde* dataset.

	A	B	C	D
16	vsdevsde(where=(resp ne .))	VSTESTCD	"RESP"	
17		VSTEST	"Respiratory Rate"	
18		VSORRES	put(resp,best12.)	
19		VSORRESU	"BREATHS/MIN"	
20		VSSTRESC	put(resp,best12.)	
21		VSSTRESN	resp	
22		VSSTRESU	"BREATHS/MIN"	
23	vsdevsde(where=(bpd ne .))	VSTESTCD	"DIABP"	
24		VSTEST	"Diastolic Blood Pressure"	
25		VSORRES	put(bpd,best12.)	
26		VSORRESU	"mmHg"	
27		VSSTRESC	put(bpd,best12.)	
28		VSSTRESN	bpd	
29		VSSTRESU	"mmHg"	
30	vsdevsde(where=(bps ne .))	VSTESTCD	"SYSBP"	
31		VSTEST	"Systolic Blood Pressure"	
32		VSORRES	put(bps,best9.)	
33		VSORRESU	"mmHg"	
34		VSSTRESC	put(bps,best9.)	
35		VSSTRESN	bps	
36		VSSTRESU	"mmHg"	
37	vsdevsde(where=(ht ne .))	VSTESTCD	"HEIGHT"	
38		VSTEST	"Height"	
39		VSORRES	put(ht,best9.)	
40		VSORRESU	"cm"	
41		VSSTRESC	put(ht,best9.)	
42		VSSTRESN	ht	
43		VSSTRESU	"cm"	
44	vsdevsde(where=(wt ne .))	VSTESTCD	"WEIGHT"	
45		VSTEST	"Weight"	
46		VSORRES	put(wt,best9.)	
47		VSORRESU	"kg"	
48		VSSTRESC	put(wt,best9.)	
49		VSSTRESN	wt	
50		VSSTRESU	"kg"	
51	all(stack)	STUDYID	study	
52		DOMAIN	&domain	
53		VSSEQ	%SEQUENCE()	
54		VISITNUM	%VISITLOOKUP(_variable=visitnum, _key=visit vtldt)	
55		VISIT	visit	
56		VISITDY	%VISITLOOKUP(_variable=visitdy, _key=visit vtldt)	
57		VSBLFL	%CONVERTIF(_if_variable=visit, _if_value=SCREENING, _then_value=Y)	
58		VSDTC	vtldt	
59		VSDY	%STUDYDAY(_date=vtldt)	
60		VSSTAT	%CONVERTIF(_if_variable=vtlrec, _if_value=NO, _then_value=NOT DONE, _else_value=)	
61		USUBJID	%GET_USUBJID()	
62				
63				
64				
65				

Figure 8. VS domain dataset creation from multiple datasets

In SAS, the code that performs this transformation would look as follows:

```
data temp1;

    set vsdevsde(where=(resp ne .));

    vstestcd="RESP":

    …

run;

data temp2;

    set vsdevsde(where=(bpd ne .));

    vstestcd="DIABP":

    …

run;

data VS;

    set temp1 temp2;

    …

run;
```

MERGING MULTIPLE DATASETS USING THE SAME MERGE KEY

If the merge key column is not defined, the source datasets will be merged by the default variable: the patient ID.

Furthermore, the CDISC Architect scans the Dataset column and discovers the special defined term "all" which instructs CDISC Architect that all datasets encountered will be merged. CDISC Architect creates each dataset in the order that they are specified in the Dataset column. Thus, the CDISC Architect will create the dataset eldeelde. Then, the CDISC Architect will create the dataset elco. Once the two datasets are created, the CDISC Architect will merge the datasets together based on the merge key specified. The CDISC Architect now encounters the "*all*", which instructs the CDISC Architect to create the CDISC variables. The equivalent code would look as follows:

```
data temp1;
    set eldeelde;
    studyid=study;
    domain=&domain;
    usubjid=%concatenate(_variables=study invsite patnum);
    …
run;

data temp2;
    set elco;
    studyid=study;
    domain=&domain;
    usubjid=%concatenate(_variables=study invsite patnum);
    …
run;

data temp3;
    merge temp1(in=intemp1) temp2(in=intemp2);
    by patnum;
    if intemp1 and intemp2;
run;

data IE;
    set temp3;
```

```
        ieseq=%sequence();
    run;
```

The CDISC Architect can do more than a single merge, for instance if three datasets are specified instead of two:

```
    data temp1;
    …
    run;

    data temp2;
    …
    run;

    data temp3;
    …
    run;

    data temp4;
    merge temp1(in=intemp1) temp2(in=intemp2) temp3(in=intemp3);
        by patnum;
        if intemp1 and intemp2 and intemp3;
    run;

    data IE;
        set temp4;
        ieseq=%sequence();
    run;
```

MERGING MULTIPLE DATASETS USING DIFFERENT MERGE KEYS

The CDISC Architect can create and merge each dataset in the order that they are specified in the Dataset column. Furthermore, the CDISC Architect scans the Dataset column and discovers the special defined term "*all*(*stack*)" which instructs the CDISC Architect that all merged datasets will be stacked by setting together.

The CDISC Architect will create the datasets *hxdehxde* and *hxco*, and merge on PATNUM. The, the CDISC Architect will create the dataset *srdesrde* and *srco*, and merge on PT. Once the two datasets are created, the CDISC Architect will stack the two creating an MH dataset:

```
    data temp1;
        set hxdehxde;
        …
    run;

    data temp2;
        set hxco;
        …
    run;

    data temp12;
        merge temp1(in=intemp1) temp2(in=intemp2);
        by patnum;
        if intemp1 and intemp2;
    run;

    data temp3;
        set srdesrde;
        …
    run;

    data temp4;
        set srco;
```

```
        …
    run;

    data temp34;
        merge temp3(in=intemp3) temp4(in=intemp4);
        by pt;
        if intemp3 and intemp4;
    run;

    data MH;
        set temp12 temp34;
        mhseq=%sequence();
        …
    run;
```

SOLUTION BENEFITS

THE NEED FOR A CDISC SDTM DATA MODEL

CDISC is a standard initiative for clinical data that started in 1998. As a result, it is mature and offers a complete framework to manage clinical data in a standardized and uniform way. Once clinical data is saved in a standard format, we can improve SAS code re-usability for the many programs used in data management and biostatistics: Edit Checks, Patient Profile, TLGs, and custom reports. In addition, cross study analyses become easier, and Clinical Summary of Safety (CSS) or Clinical Summary of Efficacy are readily available. Finally, CDISC is a standard that is strongly recommended by the FDA, thus complying to this format significantly speeds-up the review process.

WHY MAPPING IS NECESSARY

While many EDC vendors are attempting to support the CDASH standard in order to start capturing CDISC data directly from the system, it is not really feasible for the clinical database to fully comply with the company CDISC data model for a number of reasons:

- A clinical database like an EDC system needs to be flexible enough to capture any form of clinical data and cannot be tied to a particular data model.

- The CDISC model is subject to interpretation and allows for some flexibility, which means that every sponsor company will implement SDTM with some variation.

- SDTM structure is flat and non hierarchical

As a result, programmers have to convert the clinical database into a SDTM format. While this task is not very complex, it is extremely tedious due to the number of CDISC domains and variables. Also, many ETL (Extract, Transform, Load) programmers lack the CDISC domain expertise and this may result in delays and errors in the data conversion. Although SAS is one of the leading ETL vendors, a full CDISC conversion would quickly result in thousand of lines of code difficult to maintain, understand and re-use.

STREAMLINE THE DATA MAPPING & CONVERSION

Our framework allows CDISC domain experts with minimum programming experience to implement complex CDISC transformation as the data mapping rules are fully abstracted in an easy-to-read Excel spreadsheet. In fact, this spreadsheet can play the role of specifications. It is also source code as the spreadsheet is automatically converted through our SAS macro to SAS code to convert the clinical data into SDTM. This approach allows for quick updates of the mapping definitions and versioning of the mapping excel file. Data conversion may then take place any time new raw data extracts are produced through a simple SAS macro call.

CONCLUSION

We showed in this paper how we could use an Excel-based framework to document mapping rules, and use SAS to automatically convert clinical data to the CDISC SDTM format. It shows how SAS is an effective development platform for ETL transformation. By streamlining the CDISC data transformation process, all promises for code re-usability, data reporting, cross-study analyses, and improved FDA submission are made reality.

CONTACT INFORMATION

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

Name: Sophie McCallum
Enterprise: Clinovo
Address: 1208 East Arques Avenue
City, State ZIP: Sunnyvale CA 94085
Work Phone: 408-773-6258
E-mail: sophie@clinovo.com
Web: www.clinovo.com

Name: Stephen Chan
Enterprise: Clinovo
Address: 1208 East Arques Avenue
City, State ZIP: Sunnyvale CA 94085
Work Phone: 408-940-3941
E-mail: stephen@clinovo.com
Web: www.clinovo.com