

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

Ale Gicqueau, Clinovo, Sunnyvale, CA

Marc Desgrousilliers, 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 fasten 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, the mapping file, is quite straightforward for anyone with a technical background but is also accessible to a business analyst with no programming experience. The run-time transformation engine may be called by a SAS program, a UNIX shell or a DOS command and integrate 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.

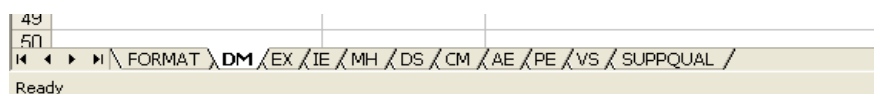


Figure 1. The different mapping file tabs

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”.

	A	B	C	D	E	F	G
1	format	from	tovalue				
2	\$sev						
3		1	MILD				
4		2	MODERATE				
5		3	SEVERE				
6		4	LIFE-THREATENING OR DISABLING				
7		5	DEATH-RELATED				
8	\$yn						
9		YES	Y				
10		NO	N				
11		UNK	U				
12	\$action_taken						
13		NONE	DOSE NOT CHANGED				
14		DRUG PERMANENTLY DISCONTINUED	DRUG WITHDRAWN				
15		DOSE INCREASED	DOSE INCREASED				
16		DOSE HELD	DRUG INTERRUPTED				
17		DOSE DECREASED	DOSE REDUCED				
18		INFUSION INTERRUPTED	DRUG INTERRUPTED				
19		INFUSION RATE DECREASED	NA				
20	\$inrange						
21		N	NORMAL				
22		L	LOW				
23		H	HIGH				
24	\$ie_desc						
25		I-7	Granulocyte count of >= 1500µL, platelet count of >= 10				
26		I-8A	Serum bilirubin less or equal to the upper limit of normal				
27		I-11	Fasting total serum cholesterol 220 mg/dL with choleste				
28		I-12	At least 4 weeks since last chemotherapy, investigation				
29	\$disp_event						
30		SUBJECT DISCONTINUED EARLY	DISCONTINUED EARLY				
31		COMPLETION	COMPLETED STUDY				
32							

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.

Dataset	Merge Key	CDISC variable	Expression	Comments
hxdehxde	patnum	MHTERM	hxdes	
		MHCAT	hxtyp	
		MHPRESP	%CONVERTIF(_if_variable=hxtyp,_if_value=TARGETED,_then_value=Y,_else_value=N)	
		MHDTC	dcmdt	
		MHSTDTC	hxdt	
		MHDY	%STUDYDAY(_date=hxdt)	
hxco	patnum	USUBJID	%GET_USUBJID()	
srdesrde	pt	MHTERM	srdes	
		MHCAT	"SURGERY / PROCEDURE"	
		MHDTC	%FORMAT(_variable=dcmdt,_format=yymmdd10)	
		MHSTDTC	srdt	
		MHDY	%STUDYDAY(_date=srdt)	
srco	pt	USUBJID	%GET_USUBJID()	
all(stack)		STUDYID	study	
		DOMAIN	&domain	
		MHSEQ	%SEQUENCE()	

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 already defined SDTM variables. Therefore, the SUPPQUAL variables metadata must be fully defined. This includes the domain name, the variable name, label, type, length, source, etc...

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.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Domain	VariableName	VariableLabel	Type	Len	Codes	Origin	Role	Comments	CORE	Drop	SUPP	CO	VL	EVAL	DATE	REF
DM	XBIRTHDC	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	AECTX	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 Prescribed	Num	8	CRF	Result Qualifier	shh3925g	PERM	QNAM							
EX	TXAS2	Total No. of 125 mg Capsules Prescribed	Num	8	CRF	Result Qualifier	shh3925g	PERM	QNAM							
EX	TXAS3	Total No. of 270 mg Capsules Prescribed	Num	8	CRF	Result Qualifier	shh3925g	PERM	QNAM							
EX	TXAS4	Total No. of 150 mg Capsules Prescribed	Num	8	CRF	Result Qualifier	shh3925g	PERM	QNAM							
EX	ITXAS1	Initial Assigned Total Daily Dose	Char	200	CRF	Result Qualifier	shh3925g	PERM	QNAM							
EX	ITXAS2	Modified 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.

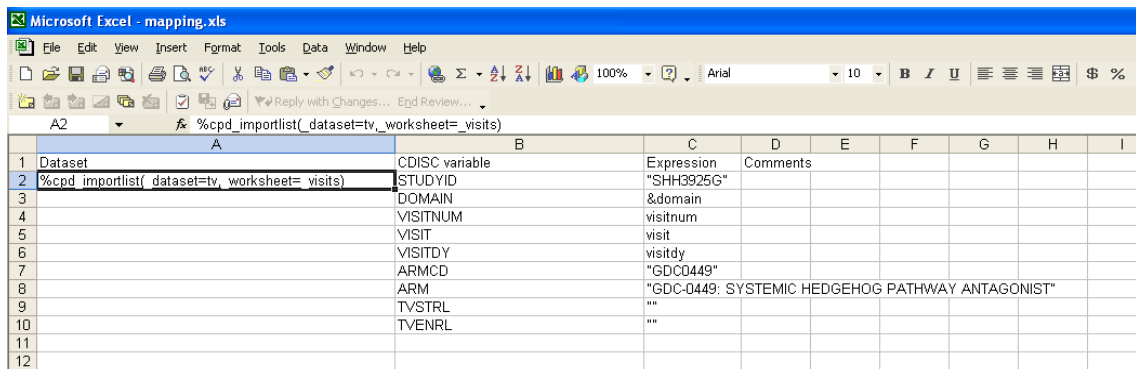
Each domain tab contains instructions to create the dataset named by the tab, for example, . DM. The source datasets specified in the tab may be merged and/or 'set' (aggregated) together. The SDTM variables listed in the CDISC variable column are assigned values as specified in the Expression column.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

DATASET COLUMN

The Dataset column indicates which source dataset to use. Before CDISC Architect begins to operate 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 which 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.



1	Dataset	CDISC variable	Expression	Comments	F	G	H	I
2	%cpd_importlist(dataset=tv, worksheet= visits)	STUDYID	"SHH3925G"					
3		DOMAIN	&domain					
4		VISITNUM	visitnum					
5		VISIT	visit					
6		VISITDY	visitdy					
7		ARMCD	"GDC0449"					
8		ARM	"GDC-0449: SYSTEMIC HEDGEHOG PATHWAY ANTAGONIST"					
9		TVSTRL	""					
10		TVENRL	""					
11								
12								

Figure 5. Example of dataset creation by a macro

MERGE KEY COLUMN

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.

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. CDISC Architect can use macros that have been already 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*.

The screenshot shows a Microsoft Excel spreadsheet titled "mapping.xls". The spreadsheet contains a table with three columns: A (Dataset), B (CDISC variable), and C (Expression). The table lists various CDISC variables and their corresponding expressions for different source datasets.

	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

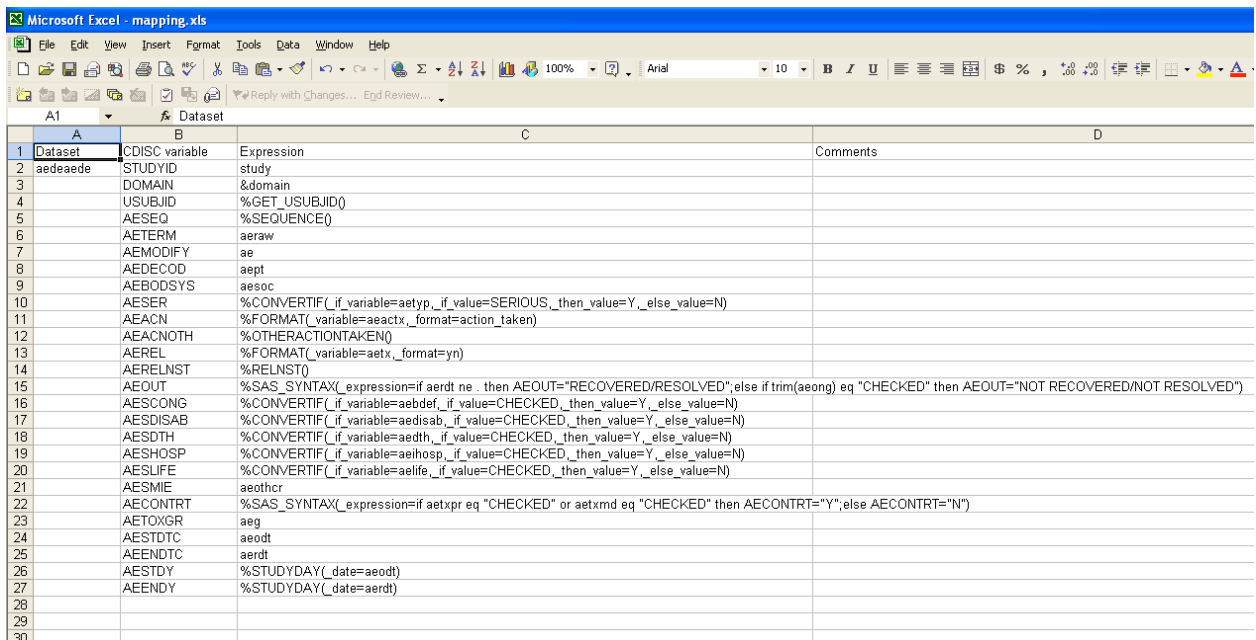
The JOIN column is optional and is usually placed on the right of the 'Merge Key' column. By default, data sets are merged using inner join. An inner join is the most common join operation and can be regarded as the default join-type. Inner join creates a new result table by combining column values of two tables based upon the join-predicate. The query compares each row of the first table with each row of the second table to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. The result of the join can be defined as the outcome of first taking the [Cartesian product](#) (or cross-join) of all records in the tables (combining every record in table A with every record in table B)—then return all records which satisfy the join predicate. Inner joins are specified using the "I" keyword next to the corresponding merge key.

An outer join does not require each record in the two joined tables to have a matching record. The joined table retains each record—even if no other matching record exists. Outer joins subdivide further into left outer joins, right outer joins, and full outer joins, depending on which table(s) one retains the rows from (left, right, or both). Outer joins are specified using the "I" keyword next to the corresponding merge key.

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.



The screenshot shows a Microsoft Excel spreadsheet titled 'mapping.xls'. The spreadsheet has four columns: A (Dataset), B (CDISC variable), C (Expression), and D (Comments). The data is as follows:

A	B	C	D
1 Dataset	CDISC variable	Expression	Comments
2 aedeae	STUDYID	study	
3	DOMAIN	&domain	
4	USUBJID	%GET_USUBJID()	
5	AESEQ	%SEQUENCE()	
6	AETERM	aeraw	
7	AEMODIFY	ae	
8	AEDECOD	aept	
9	AEBODSYS	aesoc	
10	AESER	%CONVERTIF(_if_variable=aetyp,_if_value=SERIOUS,_then_value=Y,_else_value=N)	
11	AEACN	%FORMAT(_variable=aeactx,_format=action_taken)	
12	AEACNOTH	%OTHERACTIONTAKEN()	
13	AEREL	%FORMAT(_variable=aetx,_format=yn)	
14	AERELNST	%RELNST()	
15	AEOUT	%SAS_SYNTAX(_expression=if aerdt ne . then AEOUT="RECOVERED/RESOLVED",else if trim(aeong) eq "CHECKED" then AEOUT="NOT RECOVERED/NOT RESOLVED")	
16	AESCONG	%CONVERTIF(_if_variable=aebdef,_if_value=CHECKED,_then_value=Y,_else_value=N)	
17	AESDISAB	%CONVERTIF(_if_variable=aedisab,_if_value=CHECKED,_then_value=Y,_else_value=N)	
18	AESDTH	%CONVERTIF(_if_variable=aedth,_if_value=CHECKED,_then_value=Y,_else_value=N)	
19	AESHOSP	%CONVERTIF(_if_variable=aeihosp,_if_value=CHECKED,_then_value=Y,_else_value=N)	
20	AESLIFE	%CONVERTIF(_if_variable=aelife,_if_value=CHECKED,_then_value=Y,_else_value=N)	
21	AESMIE	aeothcr	
22	AECONTRT	%SAS_SYNTAX(_expression=if aetxpr eq "CHECKED" or aetxmd eq "CHECKED" then AECONTRT="Y",else AECONTRT="N")	
23	AETOXGR	aeg	
24	AESTDTC	aeodt	
25	AEENDTC	aerdt	
26	AESTDY	%STUDYDAY(_date=aeodt)	
27	AEENDY	%STUDYDAY(_date=aerdt)	
28			
29			
30			

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

For example, in Figure 7, the CDISC Architect interprets the AE tab as creating a dataset AE in a single data step from the *aedeae* dataset. The dataset *aedeae* 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 aedeae;
  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 use the term “*all(stack)*”. CDISC Architect interprets the tab as creating a dataset VS by stacking multiple datasets created from the *vsdebsde* dataset.

A	B	C	D
16 vsdebsde(where=(resp ne .))	VSTESTCD	"RESP"	
17	VSTEST	"Respiratory Rate"	
18	VSORRES	put(resp_best12.)	
19	VSORRESU	"BREATHS/MIN"	
20	VSTSTRESC	put(resp_best12.)	
21	VSTSTRESN	resp	
22	VSTSTRESU	"BREATHS/MIN"	
23 vsdebsde(where=(bpd ne .))	VSTESTCD	"DIABP"	
24	VSTEST	"Diastolic Blood Pressure"	
25	VSORRES	put(bpd_best12.)	
26	VSORRESU	"mmHg"	
27	VSTSTRESC	put(bpd_best12.)	
28	VSTSTRESN	bpd	
29	VSTSTRESU	"mmHg"	
30 vsdebsde(where=(bps ne .))	VSTESTCD	"SYSBP"	
31	VSTEST	"Systolic Blood Pressure"	
32	VSORRES	put(bps_best9.)	
33	VSORRESU	"mmHg"	
34	VSTSTRESC	put(bps_best9.)	
35	VSTSTRESN	bps	
36	VSTSTRESU	"mmHg"	
37 vsdebsde(where=(ht ne .))	VSTESTCD	"HEIGHT"	
38	VSTEST	"Height"	
39	VSORRES	put(ht_best9.)	
40	VSORRESU	"cm"	
41	VSTSTRESC	put(ht_best9.)	
42	VSTSTRESN	ht	
43	VSTSTRESU	"cm"	
44 vsdebsde(where=(wt ne .))	VSTESTCD	"WEIGHT"	
45	VSTEST	"Weight"	
46	VSORRES	put(wt_best9.)	
47	VSORRESU	"kg"	
48	VSTSTRESC	put(wt_best9.)	
49	VSTSTRESN	wt	
50	VSTSTRESU	"kg"	
51 all(stack)	STUDYID	study	
52	DOMAIN	&domain	
53	VSSEQ	%SEQUENCE()	
54	VISITNUM	%VISITLOOKUP(_variable=visitnum,_key=visit vldt)	
55	VISIT	visit	
56	VISITDY	%VISITLOOKUP(_variable=visitdy,_key=visit vldt)	
57	VSBLFL	%CONVERTIF(_if_variable=visit,_if_value=SCREENING,_then_value=Y)	
58	VSDTC	vldt	
59	VSDY	%STUDYDAY(_date=vldt)	
60	VSSTAT	%CONVERTIF(_if_variable=vtrec,_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 vsdebsde(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, CDISC Architect will create the dataset eldeelde. Next, CDISC Architect will create the dataset elco. Once two datasets are created, CDISC Architect will merge the datasets together based on the merge key specified. CDISC Architect now encounters the “all”, which instructs 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;

```

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

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

CDISC Architect will create the datasets *hxdehxde* and *hxco*, and merge on PATNUM. Next, CDISC Architect will create the dataset *srdesrde* and *srco*, and merge on PT. Once two datasets are created, CDISC Architect will stack the two creating 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 was started in 1998. As a result, it is mature and offers a complete framework to manage clinical data in a standard 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 are now made easy, 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 and it speeds-up the review process and improve submission accuracy.

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 allow 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 in 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. SAS is also one of the leading ETL vendors however a full CDISC conversion will result quickly in thousand of lines of code difficult to maintain, understand and re-use.

STREAMLINE THE DATA MAPPING & CONVERSION

Our frameworks 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 converted automatically 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, and update the pre-built Edit Checks or Reports automatically.

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 transformations. By streamlining the CDISC data transformation processes, all promises for code re-usability, data reporting, cross-study analyses, improved FDA submissions are made reality.

CONTACT INFORMATION

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

Name: Ale Gicqueau

Enterprise: Clinovo

Address: 1208 East Arques Avenue

City, State ZIP: Sunnyvale CA 94085

Work Phone: 408 773 6250

E-mail: ale@clinovo.com

Web: www.clinovo.com

Name: Marc Desgrousilliers

Enterprise: Clinovo

Address: 1208 East Arques Avenue

City, State ZIP: Sunnyvale CA 94085

Work Phone: 408 773 6253

E-mail: marc@clinovo.com

Web: www.clinovo.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.