

# Step by Step to PROC CDISC and More

Sy Truong, Meta-Xceed, Inc, Milpitas, CA

---

## ABSTRACT

There are many crucial steps that are required during the implementation of the CDISC data models. This paper presents a step by step procedure that is based on real world implementations. Rather than stumbling through trial and error, these steps are based on lessons learned from implementations of CDISC SDTM version 3.1. This paper will cover technical challenges along with methodologies and processes. Some of the topics covered include:

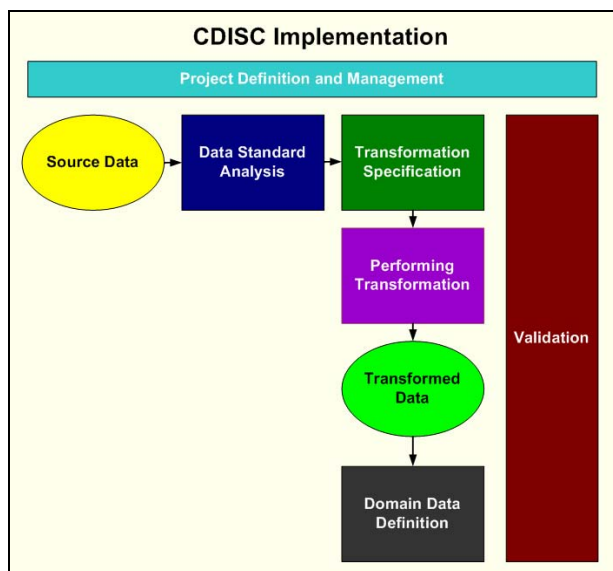
- Project Definition, Plan and Management
- Data Standard Analysis and Review
- Data Transformation Specification and Definition
- Performing Data Transformation to Standards
- Review and Validation of Transformations and Standards
- Domain Documentation for DEFINE.PDF and DEFINE.XML

The regulatory requirements are going to include CDISC in the near future and the benefits are obvious. It is therefore wise and prudent to implement with techniques and processes refined from lessons learned from real life implementations.

## INTRODUCTION

CDISC standards have been in development for many years. There have been structural changes to the recommended standards going forward from version 2 to 3. It is still an evolving process but it has reached a point of critical mass that organizations are recognizing the benefits of taking the proposed standard data model out of the theoretical and putting it into real life applications. The complexity of clinical data coupled with technologies involved can make implementation of a new standard challenging. This paper will explore the pitfalls and present methodologies and technologies that would make the transformation of nonstandard data into CDISC efficient and accurate.

There are some tasks within the process that can be applied asynchronously, but the majority of the steps are dependent on each other and therefore follow a sequence. The process is described below:



It is important to have a clear vision of the processes for the project before you start. This provides the ability to resource and plan for all the processes. This is an important step since the projects can push deadlines and break budgets due to the resource intensive nature of this effort. The organization and planning for this undertaking can become an essential first step towards an effective implementation.

## PROJECT MANAGEMENT

Before any data is transformed or any programs are developed, a project manager needs to clearly define the project for CDISC implementation. This is an essential step which will clarify the vision for the entire team and will galvanize the organization into committing to this endeavor. The project definition and plan works on multiple levels from providing a practical understanding of the steps required to creating a consensus building team effort. This can avoid the potential political battles which may arise among distinct departments within an organization.

**STEP 1: DEFINE SCOPE** – The project scope should be clearly stated in a document. This does not have to be long and can be as short as one paragraph. The purpose of this is to clearly define the boundaries of the project since without this definition, the project has tendencies towards scope creep. It can therefore potentially eat up your entire resource budget. Some of the parameters to be considered for the scope of the project include:

- **Pilot** – For an initial project, it is a good idea to pilot this on one or two studies before implementing this broadly. The specific study should be selected based on the number of datasets and number of rows of each data.
- **Roll Out** – This could be scoped as a limited roll out of a new standard or a global implementation for the entire organization. This also requires quantifying details such as how many studies are involved and which group will be affected. Not only does this identify resources in the area of programming and validation, but it also determines the training required.
- **Standard Audience** – The scope should clearly identify the user groups who will be affected by this standard. It can be limited to the SAS programming and Biostatistics group, or it can have implications for data managers, publishing, regulatory, and electronic submission groups.
- **Validation** – The formality of the validation is dictated by the risk analysis which needs to be clearly defined separately. The scope would define if the project would include validation, or if this would be part of a separate task.
- **Documentation** – The data definition documentation is commonly generated as part of an electronic submission. It is a task that is implemented with a CDISC implementation. The scope would identify if the data definition is part of the project or considered another project all together.
- **Establishing Standards** – The project may be used to establish a future set of standards that will be implemented with this new standard. The scope should identify if it is within the scope to establish global standards or just meant as a project specific implementation.

The scope document is a form of a requirement document which will help you identify the goals for this project. It can also be used as a communication tool to other managers and team members to set the appropriate level of expectations.

**STEP 2: PROJECT PLAN** – Once the tasks have been clearly documented, the list of tasks will be expanded into a project plan. The project plan is an extension of the task list including more of the following types of information:

- *Project Tasks* – Tasks are grouped by function. This is usually determined by the skills required to perform the task. This can correlate to individuals involved or whole departments. Groups of tasks can also be determined by the chronological order in which they are to be performed. If a series of steps require that they be done one after another, they should be grouped.
- *Tasks Assignments* – Once the tasks have been grouped by function, they are assigned to a department, manager or an individual. The logistics of this depends on the SOPs of your organization. This however needs to be clearly defined for planning and budgeting purposes.
- *Schedules of Tasks* – A time line is drafted noting at a high level when important deliverables or milestones are met. The titles of the tasks are the same as the title for the group of tasks. This will allow users to link back to the list of tasks to understand the details from the calendar. The schedule is also shown in calendar format for ease of planning.

A subset and sample of the project plan is shown here:

## Study ABC1234 CDISC Transformation Project Plan

### Overview

This project plan will detail some of the tasks involved in transforming the source data of study ABC1234 into CDISC SDTM in preparation for electronic submission. The proposed time lines are intended as goals which can be adjusted to reflect project priorities.

### Project Tasks

The following tasks are organized into groups of tasks which have some dependency. They are therefore organized in chronological order.

1. Data Review
  1. Evaluate variable attributes differences within internal data of ABC1234
  2. Evaluate variable attributes between ABC1234 as compared to ACME Standards
  3. Evaluate ABC1234 differences and similarities with CDISC SDS v3.1
  4. Evaluate potential matches of ABC1234 variable names and labels against CDISC SDS v3.1
  5. Initial evaluation of ABC1234 against CDISC evaluation
  6. Generate metadata documentation of the original source data from ABC1234
  
2. Data Transformation Specifications
  1. Perform a thorough review of all data and associated attributes against CDISC SDS v3.1. Identify all recommended transformation requirements. This is documented in a transformation requirement specification.
  2. Create transformation models based on the transformation specifications for each data domain.
  3. Have transformation reviewed for feedback.
  4. Update the specification to reflect feedback from review

### Task Assignments

Project Tasks	Project Manager	Team Managers
Data Review	James Brown, Director of Data Management	James Brown Billy Joel Joe Jackson
Data Transformation Specification	Janet Jackson, Manager of Biometry	Elton John Mariah Carey Eric Clapton

### Schedule of Tasks

August 2005

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		<u>12</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>
	Data Review					
<u>7</u>	<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>
<u>14</u>	<u>15</u>	<u>16</u>	<u>17</u>	<u>18</u>	<u>19</u>	<u>20</u>
<u>21</u>	<u>22</u>	<u>23</u>	<u>24</u>	<u>25</u>	<u>26</u>	<u>27</u>
	Data Transformation Specifications				Final review of Data Transformation	
<u>28</u>	<u>29</u>	<u>30</u>	<u>31</u>	-	-	-

**STEP 3: VALIDATION** – Validation is an essential step towards maintaining accuracy and integrity throughout the process. It can be determined to be outside the scope of some projects since it is resource intensive. The following lists some of the tasks that are performed as it pertains to validation.

- *Risk Assessment* – An evaluation of each task or groups of tasks. This will evaluate and determine the level of validation effort to be performed.
- *Test Plan* – This will document the testing approach and methodologies used during the validation testing. It describes how the testing will be performed and how deviations are collected and resolved. It will also include test scripts used during testing.
- *Summary Results* – This will document all the findings as a result from the testing. It quantifies the number of deviations and documents how they are to be fixed.

The following example shows a form that is used to collect the tasks and associated risks.

Risk Assessment Title	<i>Risk Assessment of analysis files for sample study.</i>	
Identify the task where the programs reside which contributes to the risk. ⇒	<b>Task Name and Location</b>	<i>Interim Analysis on my server</i>
Identify the groups of programs to see which categories they appear in. ⇒	<input checked="" type="checkbox"/> <b>Analysis Files (20)</b> <input type="checkbox"/> <b>Listings (5)</b> <input type="checkbox"/> <b>Summary Tables (10)</b> <input type="checkbox"/> <b>Graphs (10)</b> <input type="checkbox"/> <b>Edit Checks (5)</b> <input type="checkbox"/> <b>Other:</b>	<i>This is a subset of the analysis files Just as an example.</i>
From the group of studies identified, classify the types of programs. ⇒	Score: 20 <input checked="" type="checkbox"/> <b>Single Use Program in One Study (5)</b> <input type="checkbox"/> <b>Single Use Program (10)</b> <input type="checkbox"/> <b>Multi Use Program in One Study* (20)</b> <input type="checkbox"/> <b>Multi Use Utility or Macro in Multiple Studies* (30)</b> <input type="checkbox"/> <b>Multi Use Utility or Macro in All Studies * (40)</b> <input type="checkbox"/> <b>Other</b>	<i>This is a single use program and it is going to be used in this study only.</i>
What is the likelihood that the program would produce errors or incorrect results? ⇒ <ul style="list-style-type: none"> <li>• Are the specifications not clearly defined?</li> <li>• Does the program use custom logic versus SAS PROCs or standard macros?</li> </ul>	Score: 5 <input type="checkbox"/> <b>Error Likelihood Detection (0-20)</b>	<i>Since there are some derivations and hard coded values in this code, I will give it some degree of likelihood.</i>
Score: 10		

The test plan can vary depending on the amount of details and level of formality as determined by the risk assessment. The following example shows you a subset of a more formal test plan. This can be abbreviated to handle transformation tasks that are deemed to be of lower risk.

### Test Scripts

The format of the test scripts can also vary depending on the formality of your testing. It is important to have each test case contain a unique identifier such as a test case number. This is what a tester and reviewer use to track the test and its associated deviations.

<b>System Name and Version:</b>	Wonder Drug ABC1234 CDISC	<b>Functional Area/Module:</b>	Standardize E3200 Data
<b>Test Script Number:</b>	1 (Requirement 4.1)		
<b>Overall Test Objective:</b>	Verify the variable attributes of the existing source data of Wonder Drug ABC1234		
<b>Specific Test Condition:</b>	Tester has read access to input data.		
<b>Test Program Run Location:</b>	Test Area		
<b>Test Program Name(s):</b>	difftest_avf.sas		
<b>Test Script Prerequisites:</b>	None		

Step	Instruction	Expected Result	Actual Result	Initials/Date
1	Right mouse click on test script program and select batch submit.	Script file is executed.		
2	Evaluate log file for errors.	No errors are found.		
3	Evaluate output files to verify that the attributes results match with the report that is performed using %difftest as part the summary report.	Output is verified against output.		
<b>Recovery:</b>	Resubmit the program.	<b>Signature/Date</b>		
<b>Final Expected Result:</b>	Verify the variable attributes of the existing source data of Wonder Drug ABC1234.	<b>Actual Result:</b> <input type="checkbox"/> Pass <input type="checkbox"/> Fail		
<b>Comments:</b>		<b>Reviewed By:</b>		

The format presented in the test plan such as the summary report can follow the same format. The examples of this paper only show a subset of the entire test plan and are intended to give you a conceptual understanding so that you can apply the same concepts to all the other parts of the documentation.

**STEP 4: TRANSFORMATION SPECIFICATION** – The specification to the transformation towards CDISC standards is a detailed road map that will be referenced and used by all team members during the transformation implementation and review. There can be different technologies used to perform this task. The following example utilizes tools including MS Excel and Transdata™. Dataset transformation is a process in which a set of source datasets and its variables are changed to meet new standard requirements. The following list describes some of the changes that can occur:

1. **Dataset Name** - SAS dataset names must be updated to match SDS standards, which require them to be no more than 8 characters in length.
2. **Dataset Label** - The descriptive labels of SAS datasets must be modified to conform to SDS standards.
3. **Variable Name** - Each variable within a SAS dataset has a unique name. Variable names can be the same across different datasets, but if they share the same name, they are generally expected to possess the same attributes. Variable names are no more than 8 characters in length.
4. **Variable Label** - Each variable has an associated label that describes the variable in more detail. Labels are no more than 40 characters in length.
5. **Variable Type** - A variable's type can be either character or numeric.
6. **Variable Length** - A character variable can vary in length from 1 to 200 characters.
7. **Format** - Variable format will be updated.
8. **Yesno** - If the value of the variable is "yes", it will produce a new row with the newly assigned value of the label.
9. **Vertical** - Multiple variables can be assigned to one variable that will produce a row if it has a value.
10. **Combine** - Combine values of multiple source variables into one destination variable.
11. **Drop** - The variable from the source dataset will be dropped when creating the destination data.
12. **Same** - The same variable with all of the same attributes will be kept in the destination data.
13. **Value Change** - This can have either a recoding of values or a formula change. This will change the actual values of the variable.

There may be other types of transformations, but these are the common transformation types that are usually documented in the specification. The transformation specification is stored in an Excel spreadsheet and is organized by tabs. The first tab named "Tables" contains a list of all the source tables. The subsequent tabs contain the transformation specifications for each dataset as specified in the initial tables tab.

### Tables Tab

The Tables tab contains the list of source datasets along with descriptions of how each one transforms into the standard data model. It also records associated data structures such as Relational Records and Supplemental Qualifiers.

#### ABC1234 Data Transformation

Source Data	CDISC Data Name	SDTM 3.1 Label	Related Records	Supplemental Qualifiers
Ae	AE	Adverse Events	AE	AE, CM, EX, DS
Ccancer	DC	Disease Characteristics		DC
Conduct	DV	Protocol Deviations		DV
Death	DS	Disposition	DS	DS
Demog	DM	Demographics Domain Model		DM, EX, DC
Discon	DS	Disposition		DS
Elig	IE	Inclusion/Exclusion Exceptions		MH
Lcea	LB	Laboratory Test Results		LB

This will list all the source datasets from the original study. There is not always a one to one transformation. That is, there may be many source datasets used to create one transformed CDISC data. This will act as an index of all the data and how they relate to each other. The relationship is not limited to relationship between source and destination data, but also how it relates to CDISC domain such as in the case of "Relational Records" and "Supplemental Qualifiers". These related data structures are used within SDTM to include data that contains values which do not fit perfectly into existing domains.

### Transformation Model Tab

Each source dataset will have a separate corresponding spreadsheet detailing the transformation. The following is an example of an adverse event transformation model tab.

#### Adverse Event Data Transformation from Study ABC1243 to CDISC SDTM 3.1

Variable	Label	Transformation Type	Update To	Domain
PATNUM	Subject ID (Num)	name label length	usubjid label="Unique Subject Identifier" length=\$15	
STUDY	Clinical Study	name label length	studyid label="Study Identifier" length=\$15	
ADCONATT	Con Med Attribution and Name	name label length combine	aereinst label="Relationship to Non-Study Treatment" length=\$140	CM
ADCTC	Adverse Event CTC	name label length	aeterm label="Reported Term for the Adverse Event" length=\$150	AE
ADCTCCAT	Organ System CTC	name label length	aebodsys label="Body System or Organ Class" length=\$30	AE
ADCTCOS	Other Specify CTC	Drop		AE

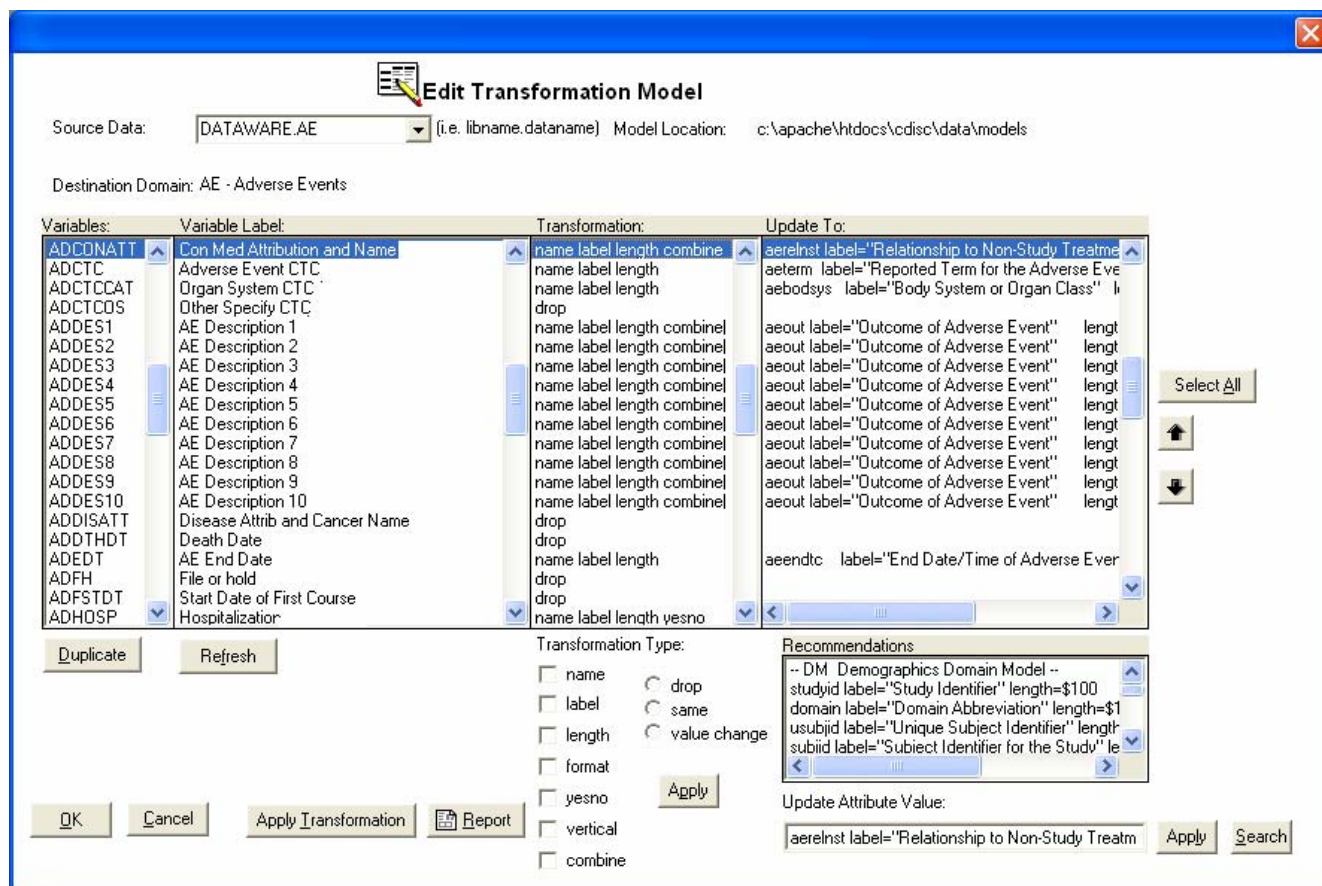
#### Key

Relational Records
Supplemental Qualifiers
Comments

In this example, the source variable ADCTCOS is moved to the supplemental qualifiers structure. Most of the transformations are pretty straightforward attribute changes. However, the transformation of type "combine" will concatenate multiple

source variables into one target variable. Most of these are going towards the AE domain except for the variable ADCONATT which is being transformed into the CM domain. This example illustrates how various details of data transformations can be expressed concisely with great detail in the form of a transformation specification.

**STEP 5: APPLYING TRANSFORMATION** – In an ideal world, the specification is completed one time and you would apply the transformation according to the specification. In the real world however, the specification goes through changes throughout the duration of the project. You would therefore need to make an executive decision at specified times to apply the transformation even when things are still changing. Because of the dynamic nature of the data, a tool such as Transdata can be very useful since the transformation specification needs to also be dynamic to keep up with the changing data. Changes to the transformation would also have implications of re-coding the transformation logic. This is where manually programming transformation can lead to constant updates and becomes a very resource intensive task. To automate this process, the same transformation specification is captured in a SAS dataset within Transdata and managed with the following screen.



All the source variables and associated labels can be managed and displayed on the left two columns. The type of transformations including the most commonly used ones are listed with check boxes and radio buttons for ease of selection and application. The new attributes of the target variables which were seen from the specification spreadsheet can also be captured here. Besides being able to edit these attributes, standard attributes from CDISC are also listed as recommendations. The advantages to managing the specifications in this manner as compared to Excel include:

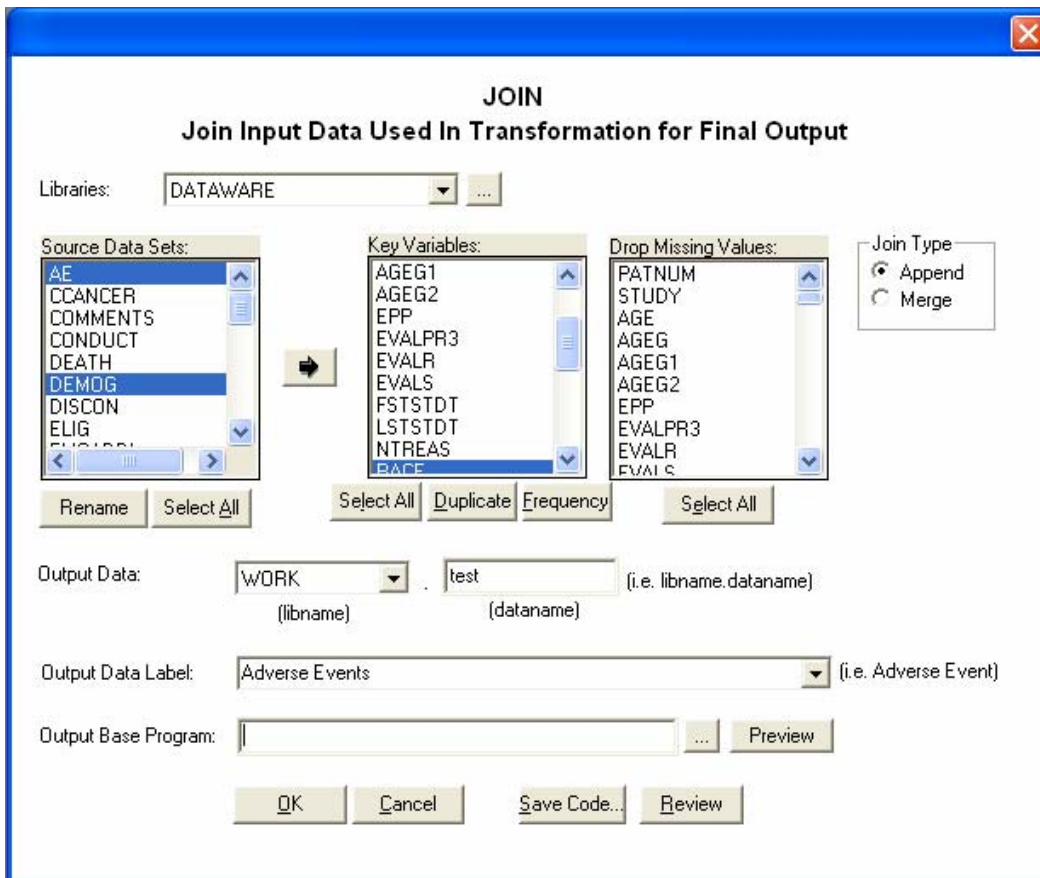
1. An audit trail is kept of all changes.
2. The selection choices of transformation type and target attributes make it easier to generate standardized transformation.
3. Transformation logic coding and algorithms can be generated directly from these definitions.
4. A refresh of the source variables can be applied against physical datasets to keep up with changing data.

## Program Transformation

Once the transformation specification has been clearly defined and updated against the data, you would need to write the program that would perform this transformation. A sample program may look like:

```
*****;
* Program: trans_ae.sas
* Path: c:\temp\
* Description: Transform Adverse Events data
*              from DATAWARE.AE to STDMLIB.AE
* By: Sy Truong, 01/21/2006, 3:49:13 pm
*****;
libname DATAWARE "C:\APACHE\HTDOCS\CDISC\DATA";
libname STDMLIB "C:\APACHE\HTDOCS\CDISC\DATA\SDTM";
data STDMLIB.AE (label="Adverse Events"
                );
  set DATAWARE.AE;
  retain obs 1;
  *** Define new variable: aereinst that combined by old variables: adconatt adoagatt;
  attrib aereinst label="Relationship to Non-Study Treatment" length=$140;
  aereinst = trim(trim(adconatt) || ' ' || adoagatt);
  drop adconatt adoagatt;
  *** Define new variable: aeout that combined by old variables: addes1 addes2 addes3 addes4 addes5
  addes6 addes7 addes8 addes9 addes10;
  attrib aeout label="Outcome of Adverse Event" length=$1000;
  aeout = trim(trim(trim(trim(trim(trim(trim(trim(trim(addes1) || delimit_aeout0 || addes2) ||
delimit_aeout1 || addes3) || delimit_aeout2 || addes4) || delimit_aeout3 || addes5) ||
delimit_aeout4 || addes6) || delimit_aeout5 || addes7) || delimit_aeout6 || addes8) || delimit_aeout7
|| addes9) || delimit_aeout8 || addes10);
  drop delimit_aeout0 delimit_aeout1 delimit_aeout2 delimit_aeout3 delimit_aeout4 delimit_aeout5
delimit_aeout6 delimit_aeout7 delimit_aeout8 delimit_aeout9 addes1 addes2 addes3 addes4 addes5
addes6 addes7 addes8 addes9 addes10 ;
run;
```

This is only an example subset since normal transformation programs are much more complex and longer. Some involve multiple transformations into separate target datasets which are then later merged into a single final target dataset. Transdata is a code generator so all of the code shown above is automatically generated. In the event that the transformation requires multiple datasets to be merged, you can develop code manually by performing PROC SORT and MERGE, or you can use the following interface in Transdata.



The two most common types of joins are classified as “append” or “merge”. The append stacks the data on top of each other with the SAS code being something like:

```

data WORK.test (label = 'Adverse
Events');
set
  input1(in=input1)
  input2(in=input2)
;
by RACE;
run;

```

The other type of merge is when the two data are actually “merged” by a particular key. The code for this is more like:

```

data WORK.test (label = 'Adverse
Events');
merge
  input1(in=input1)
  input2(in=input2)
;
by RACE;
run;

```

The difference is that you would need to use the “MERGE” statement rather than the “SET” statement. In either case, Transdata will generate this code for you so that you do not have to perform the PROC SORT and merging data step yourself.

**STEP 6: SEQUENCE, ORDER AND LENGTHS** – Data value sequence along with variable order and lengths needs to also follow standards. CDISC does specify guidance for data sequences and variable order but is not as strict on variable lengths. In either case, you would need to have these applied consistently in order for it to be standardized.

### Sequence

Any dataset that contains more than one observation per subject requires a sequence variable. The sequence variable would then identify the order of the values for each subject. If your data does not contain this sequence variable, you would need to add it. Besides the subject ID, you would also need to identify a unique identifier variable that would distinguish between the observations within one subject. This can be another group type of identification variable or a form date.

CDISC Builder supplies a tool named ADDSEQ that would add this sequence based upon the choices you decide upon a specific dataset.

The ADDSEQ tool will then create a new sequence variable containing sequential values after it sorts the data by the subject ID and identification variable. In addition to creating the sequence variables, there is also a tool that tells you if the dataset requires a sequence variable or not. It essentially verifies if there is more than one observation per subject. This will then help prompt you to add sequence variables in case it is overlooked.

### Variable Order

The data that is delivered in CDISC format needs to be ordered in a standard manner. All the key fields need to be first in

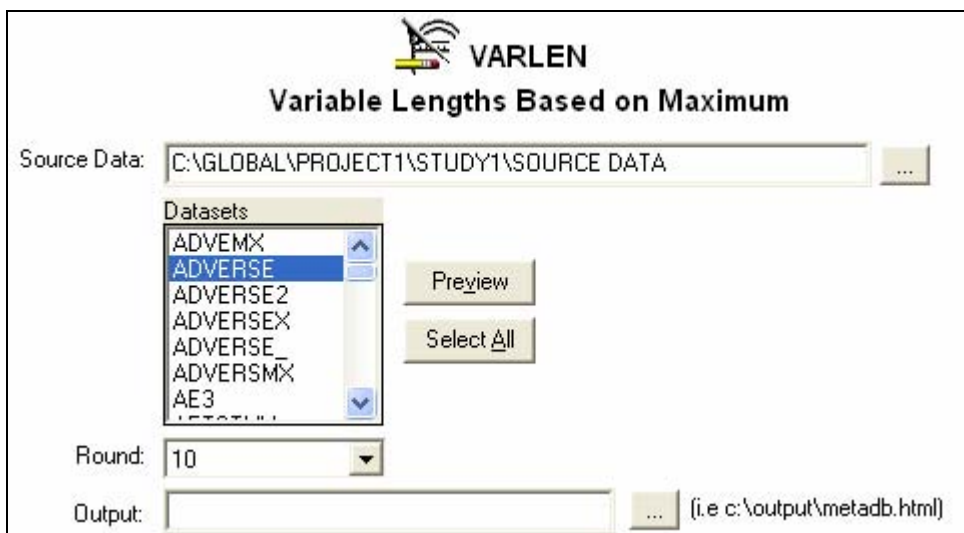
order followed by the rest of the variables. The rest of the variables are then shown in alphabetical order. SAS datasets has its variables stored in a specified order and it is not necessary in this standard order. CDISC Builder will re-order the variables with the keys appearing first followed by the rest of the variables. The rest can be optionally alphabetized or left in their original order. This task may appear mundane but can be very helpful for the reviewer who is navigating through many datasets.

### Variable Lengths

Variable lengths are not strictly specified by CDISC guidelines. It is still however important to have variable lengths follow a standard for consistency. This includes:

1. Consistent lengths between variables that are the same across different data domains
2. Optimal lengths set to handle the data

In order to accomplish the first rule of standards, if you were to assign a length of one variable such as USUBJID on one dataset, you would need to set the same length for all other variables that are the same across all datasets. The second rule suggests that if the contents of your variable for this field has the largest text value of 9 characters, it would probably be optimal to set the length to 10. It makes sense to round up to the nearest tenth to give it some buffer but not too much so that it would not be wasteful. Datasets can be very bloated and oversized for the value they carry if the second rule is not applied. The following tool named VARLEN from CDISC Builder assigns the length optimally.



In this example, the rounding option can be set to 10, 20 or none. It can therefore assign the exact maximum length which the data value contains if that is what is required. This would create the proper length statement so that your data will have the optimal lengths used for the values stored in that data.

**STEP 7: CDISC VERIFICATION** – After you have created your CDISC data structure, there are tools that you can use to verify what you have created. An example of this is PROC CDISC. This is available for both SAS version 8 and 9 as a download from the SAS website located at: [sas.com/download](http://sas.com/download) → Base SAS Software. PROC CDISC can be applied for both CDISC ODM and SDTM data models. The ODM model has many more options compared to SDTM since it has been around longer. This paper, however, will focus on the SDTM data model. In this case, you need to make the following decisions by specifying the following parameters:

1. **Input Data** – This is where you specify the SAS dataset in the SDTM structure to be verified.
2. **Model Version** – Both the ODM and SDTM have several versions since their release. In our example, the latest SDTM version is 3.1.
3. **Domain** – CDISC organizes its data into clinical domains. It is specified by an abbreviated two letter name such as “AE” for “Adverse Events” or, “IE” for “Inclusion / Exclusion”.
4. **Category** – Different domains are also placed into categories such as “Events” or “Interventions”.

You can specify all these decisions into a SAS program using PROC CDISC. An example would look like:

```
/*-----*
* Program: proc_cdisc.sas
* Path: c:\mypath
* Description: Verify input data for adverse events.
*-----*/
libname source 'C:\mypath';

proc cdisc model = sdtm;
  sdtm SDTMVersion = "3.1";
  domaindata      data = source.AE
                 domain = EX
                 category = Interventions;
run;
```

The results of the verification can be found in the SAS Log. The verification information produced is documented as WARNING or ERROR messages in the log. For example, the following values appear in the log:

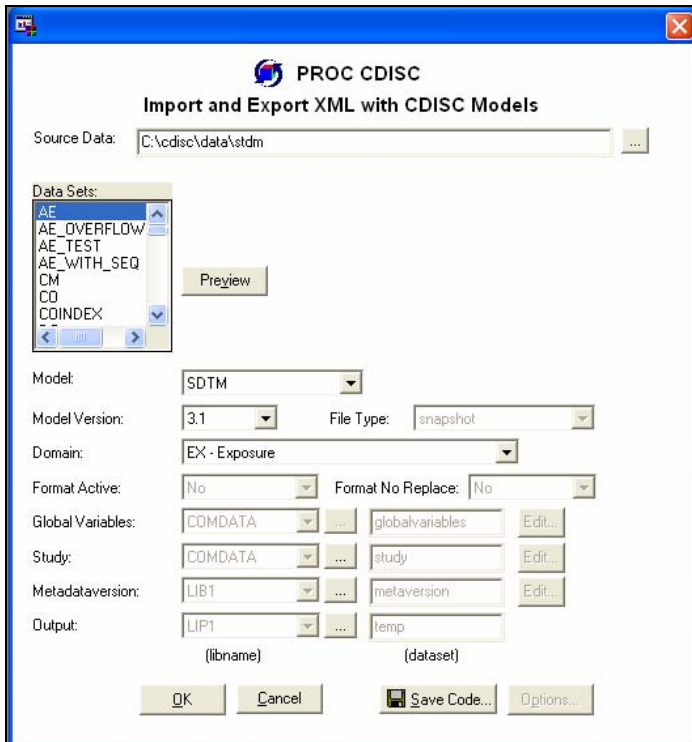
ERROR: Required parameters not contained on DOMAINDATA(Domain=EX) statement.

Required parameter EXSEQ not present.

Required parameter EXTRT not present.

Required parameter EXSTDTC not present.

This information can be useful since it is captured automatically according to the recommendations from the guidelines which you may have overlooked. If specifying parameters in a SAS procedure is too cumbersome, PROC CDISC is also available with a graphical user interface within CDISC Builder™. The graphical user interface of CDISC Builder™ assists you in selecting these parameters and making the right decisions. In this case, all the selections can be chosen from menus with pre-populated pull down menus.



Upon completion, you just need to click on the OK button to have the logic of PROC CDISC applied. There is also an option to save the PROC CDISC code based on your selection. This flexibility allows you to learn and use PROC CDISC without having to look up the syntax of each parameter.

**STEP 8: DATA DEFINITION DOCUMENTATION** – When you plan for a road trip, you need a map. This is analogous to understanding the data that is going to be part of an electronic submission. The reviewer requires a road map in order to understand what all the variables are and how they are derived. It is within the interest of all team members involved to have the most accurate and concise documentation pertaining to the data. This can help your team work internally while also speeding up the review process which can really make or break an electronic submission to the FDA.

#### Levels of Metadata

There are several steps towards documenting the data definition. Most of what is being done is documenting metadata which is information about the data that is to be included. There are several layers to the metadata. These include:

1. **General Information** – This pertains to information that affects the entire set of datasets that are to be included. It could be things such as the name of the study, the company name, or location of the data.
2. **Data Table** – This information is at the SAS dataset level. This includes things such as the dataset name and label.
3. **Variable** – This information pertains to attributes of the variables within a dataset. This includes such information as variable name, label and lengths.

The order in which the metadata is captured should follow the same order as the layers that are described.

#### Capture General Information

The following lists the types of information you need to be concerned about.

Metadata	Description
Company Name	This is the name of the organization that is submitting the data to the FDA.
Product Name	The name of the drug that is being submitted.
Protocol	The name of the study on which the analysis is being performed which includes this set of data.
Layout	The company name, product name, and protocol are all going to be displayed on the final documentation. The layout information will describe if it will be in the footnote or title and how it is aligned.

This high level metadata will be used in headers and footers on the final documentation.

#### Dataset Level Information

Some of the dataset level information can be captured through PROC CONTENTS but others need to be defined when you are documenting your data definition. Some of the information includes:

Metadata	Description
Data Library	Library name defines what physical path on which server and where the data is located. This can also be in the form of a SAS LIBNAME.
Key Fields	Keys usually correlate to the sort order of the data. These variables are usually used to merge the datasets together.
Format Library	This is where the SAS format catalog is stored.
Dataset Name	The name of the SAS dataset that is being captured.
Number of Variables	A count of the number of variables for each dataset.

Number of Records	Number of observations or rows within each dataset.
Dataset Comment	A descriptive text describing the dataset. This can contain the dataset label and other descriptive text explaining the data.

SAS Tools such as PROC CONTENTS can contribute to most of these items. However, comments and key fields can be edited which may differ from what is stored in the dataset.

### Variable Level Information


The last step and level to the domain documentation is the variable level. This includes the following:

Metadata	Description
Variable Name	The name of the SAS variable.
Type	The variable type which includes values such as Character or Numeric.
Length	The variable length.
Label	The descriptive label of the variable.
Format	SAS formats used. If it is a user defined format, it would need to be decoded.
Origins	The document where the variable came from. Sample values include: Source or Derived.
Role	This defines what type of role the variable is being used for. Example values include: Key, Ad Hoc, Primary Safety, Secondary Efficacy
Comment	This is a descriptive text explaining the meaning of the variable or how it was derived.

Similar to the data set level metadata, some of the variable level attributes can be captured through PROC CONTENTS. However, fields such as origins, role and comments need to be edited by someone who understands the meaning of the data.

### Automation with Definedoc™

Tools such as PROC CONTENTS and Excel do have capabilities to customize and automate the documentation to a degree. They are not however intended specifically for creating data definition documentation. These tools therefore have limitations. A tool that was developed entirely in SAS specifically for generating this type of documentation is Defindoc™. This tool contains both a graphical user interface and a macro interface to fit the user's requirements. The tool addresses all the disadvantages of the manual methods. It uses a similar PROC CONTENTS type of mechanism of capturing the initial metadata. However, it only retains the specific information that is pertinent to the data definition documentation.

 **Edit Definition**

	Dataset Name	Variable Name	Variable Label
1	AE	DOMAIN	Domain Abbreviation
2	AE	STUDYID	Study Identifier
3	AE	USUBJID	Unique Subject Identifier
4	AE	AEBODSYS	Body System or Organ Class
5	AE	AECAT	Category for Adverse Event
6	AE	AEDECOD	Dictionary-Derived Term
7	AE	AEENDTC	End Date/Time of Adverse Event
8	AE	AEMODIFY	Modified Reported Term
9	AE	AEOUT	Outcome of Adverse Event
10	AE	AEREL	Causality
11	AE	AERELNST	
12	AE	AESCAT	Subcategory for Adverse Event
13	AE	AESEQ	Sequence Number
14	AE	AESHOSP	Requires or Prolongs Hospitalization
15	AE	AESTDTC	Start Date/Time of Adverse Event
16	AE	AETERM	Reported Term for the Adverse Event
17	AE	AETOXGR	Standard Toxicity Grade

Value Update:  ...

Apply

Definedoc automatically captures attributes pertaining to information captured by PROC CONTENTS. For other values, it presents possible values that users can select for more consistency.

**Role Field**

Role Field

- Key
- Ad Hoc
- Primary Safety
- Secondary Efficacy

The tool also keeps track of all edits in an audit trail capturing who has updated what column so that if anything goes wrong, it can easily be traced back and fixed. One of the main advantages is that if any of the variable attributes are updated, this can be “refreshed” with a click of a button. It will not affect those fields that the user has entered, but rather, it updates other attributes such as variable names and labels.

Definedoc has the flexibility of exporting the pertinent information to an excel spreadsheet so that those users who prefer to edit their values within Excel can do so.

1	Library	Dataset	Variable	Type	Length	Variable Label	Format	Decode	Origins	Role	Comment
2	c:\apache\AE		DOMAIN	Character	30	Domain Abbreviation			Derived		
3	c:\apache\AE		STUDYID	Character	10	Study Identifier			Source	Key	
4	c:\apache\AE		USUBJID	Character	10	Unique Subject Identifi			Source	Key	5-DIGIT SUBJECT IDENTIF
5	c:\apache\AE		AEBODS\	Character	30	Body System or Organ Class			Source		
6	c:\apache\AE		AECAT	Character	30	Category for Adverse Event			Source		VALUES: ADEERS, DOSE
7	c:\apache\AE		AEDECOD	Character	40	Dictionary-Derived Term			Source		
8	c:\apache\AE		AEENDTC	Character	10	End Date/Time of Adverse Event			Source	Key	
9	c:\apache\AE		AEMODIF	Character	40	Modified Reported Term			Source		
10	c:\apache\AE		AEOUT	Character	1000	Outcome of Adverse Event			Source		AE DESCRIPTION WAS SF
11	c:\apache\AE		AEREL	Character	260	Causality			Source		
12	c:\apache\AE		AERELNS	Character	195				Source		CONCOMITANT MEDICAT
13	c:\apache\AE		AESCAT	Character	30	Subcategory for Adverse Event			Source		
14	c:\apache\AE		AESSEQ	Numeric	8	Sequence Number			Derived		
15	c:\apache\AE		AESHOSF	Character	30	Requires or Prolongs Hospitalization			Source		VALUES: NO, YES.
16	c:\apache\AE		AESTDTC	Character	10	Start Date/Time of Adverse Event			Source		
17	c:\apache\AE		AETERM	Character	150	Reported Term for the Adverse Event			Source		
18	c:\apache\AE		AETOXGF	Character	10	Standard Toxicity Grade			Source		VALUES: 1, 2, 3, 4, 5.
19	c:\apache\CM		DOMAIN	Character	30	Domain Abbreviation			Derived		
20	c:\apache\CM		STUDYID	Character	10	Study Identifier			Source	Key	

This provides the best of both worlds. It captures just the values that you want and exports this to Excel for those who prefer this interface. Once you are finished with editing the information in Excel, the same spreadsheet can be re-imported so that the information is handled centrally. Besides the dataset and variable level metadata information, Definedoc also helps automate the capture of the high level general information.

**General Information**

Output Library:

Company Name:

Product Name:

Protocol:

XPT Link Location:   (\*)  
(i.e.../xptfiles)

CRF Link Location:   (\*)

Split Char:

Layout

Left Title:  Center Title:  Right Title:

Left Footnote:  Center Footnote:  Right Footnote:

\* NOTE: This is relative to the output path: c:\temp.

This handles both the editing of the information and layout of the final report.

### Generating Documentation

The last step in the process is to generate the documentation in either PDF or XML format. The challenge is that in order to make the documentation useful, it requires hyperlinks to link the information together. The manual method does allow you to format the information in Word and this can be converted into PDF format. Even though Word and Excel can generate XML, it does not have the proper schema so there is no manual way of generating the XML version of the report. Definedoc has the flexibility of generating the report in Excel, RTF, PDF and XML.

Output File:  ...   
 (i.e. define.xls, define.rtf, define.pdf, define.xml)

It utilizes ODS within SAS to produce the output in all these formats. In addition to the XML file, Definedoc also produces the accompanying cascading style sheet to format the XML so that you can view this within a browser in a similar format as in a web browser. An example PDF output would look like:

**Sample Company, Inc. Data Definition Table Study ABC1234**

**AE : Adverse Events : (AE.XPT)**

Variable Name	Type	Length	Variable Label	Format	Origins	Role	Comment
DOMAIN	Character	30	Domain Abbreviation		Derived		
STUDYID	Character	10	Study Identifier		Source	Key	
USUBJID	Character	10	Unique Subject Identifier	11.	Source	Key	6-DIGIT SUBJECT IDENTIFIER (NUMERIC)
AEBODSYS	Character	30	Body System or Organ Class		Source		
AECAT	Character	30	Category for Adverse Event		Source		VALUES: ADEERS, DOSE DISCONTINUATION FORM 2112, DOSE REDUCTION FORM 2112, ON-STUDY (EPP), ON-STUDY FORM 1559 (NON-EPP), TOXICITY (EPP), TOXICITY FORM 1580 (NON-EPP).
AEDECOD	Character	40	Dictionary-Derived Term		Source		
AEENDTC	Character	10	End Date/Time of Adverse Event		Source	Key	
AEMODIFY	Character	40	Modified Reported Term		Source		
AEOUT	Character	1000	Outcome of Adverse Event		Source		AE DESCRIPTION WAS SPLIT INTO 10 VARIABLES TO COMPLY WITH LENGTH LIMITATIONS FOR SAS TRANSPORT FILES. CONCATENATE FIELDS 1-10 TO RECREATE THE ORIGINAL TEXT.
AEREL	Character	260	Causality		Source		
AERELNST	Character	195			Source		CONCOMITANT MEDICATION ATTRIBUTION AND NAME.

## CONCLUSION

There are many challenges in working with the CDISC SDTM version 3.1. It is clear from the structures that it is useful from a reviewer's perspective but it is structured very different from how users would use it to perform analysis. It is intended to be used for submissions so transformation is going to be necessary. Since the transformations are handled differently for each variable, the sum of the work can be tremendous. It does require organization before execution and optimization in implementation. The techniques, methodologies and tools presented in this paper demonstrate ways of optimizing working with CDISC data which is based on real world experience. Armed with these approaches based on real examples, you can avoid the mistakes and implement CDISC with success.

## REFERENCES

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.

CDISC Builder, Transdata and Definedoc and other MXI (Meta-Xceed, Inc.) product names are registered trademarks of Meta-Xceed, Inc. in the USA.

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

## **ABOUT THE AUTHOR**

Sy Truong is President of MXI (Meta-Xceed, Inc.) They may be contacted at:

Sy Truong

1751 McCarthy Blvd.

Milpitas, CA 95035

(408) 955-9333

sy.truong@meta-x.com