

## Dive into CDISC Express

Jiangtang Hu

[Jiangtang.hu@gmail.com](mailto:Jiangtang.hu@gmail.com)

[www.jiangtanghu.com/](http://www.jiangtanghu.com/)

05/07/2011

<b>1. Download and Installation</b> .....	2
<b>2. Working Flow</b> .....	3
1) Create a new study ( <i>create_new_study.sas</i> ) .....	3
2) Generate mapping file ( <i>generate_mapping_template.sas</i> ) .....	3
3) Validate mapping file ( <i>Validate_Mapping_File.sas</i> ) .....	3
4) Generate SDTM datasets ( <i>generate_SDTM.sas</i> ) .....	3
5) Validate SDTM datasets ( <i>Validate_SDTM_Domains.sas</i> ) .....	4
6) Generate Define.xml and SAS transport files ( <i>generate_Definexml.sas</i> ) .....	4
<b>3. Step 1 of 6: Create a new study</b> ( <i>create_new_study.sas</i> ) .....	4
<b>4. step 2 of 6: Generate mapping file</b> .....	6
4.1 Get the blank template mapping file ( <i>generate_mapping_template.sas</i> ) .....	7
4.2 Navigate mapping file .....	8
4.3 Data manipulation techniques in CDISC Express .....	13
4.3.1 Reference one dataset .....	13
4.3.2 Assignment .....	14
4.3.3 Match-merging .....	16
4.3.4 Concatenating .....	19
4.3.5 Transpose .....	20
4.3.6 All others: use macro! .....	25
<b>5. step 3 of 6: Validate mapping file</b> ( <i>Validate_Mapping_File.sas</i> ) .....	26
<b>6. step 4 of 6: Generate SDTM datasets</b> ( <i>generate_SDTM.sas</i> ) .....	27
<b>7. step 5 of 6: Validate SDTM datasets</b> ( <i>Validate_SDTM_Domains.sas</i> ) .....	28
<b>8. step 6 of 6: Generate Define.xml and xpt</b> ( <i>generate_Definexml.sas</i> ) .....	29
9. Recommended reading and action taken .....	30

Recently I did some research on [Clinovo](#)'s open source application, [CDISC Express](#), a SAS application based on Excel framework designed to map clinical data to CDISC SDTM domains automatically. Not perfect yet, but it is easily understandable and practically usable after few hours' of exploration of user guide. And most important, it is on the right way: an automatic CDISC converter is the magic weapon in almost every clinical programmer's dream.

CDISC Express is the first and only practically usable open source CDISC converter I even met. I wrote [a post](#) one month ago when I first tested it with great interests and reported some issues to [fix system](#). Then I also had the great opportunity to discuss the software via email with its core developer, Romain Miralles. This post is just my personal notes on how to use and dig into the

software, and will be best serve as a working documentation. You can return to me for any questions and comments.

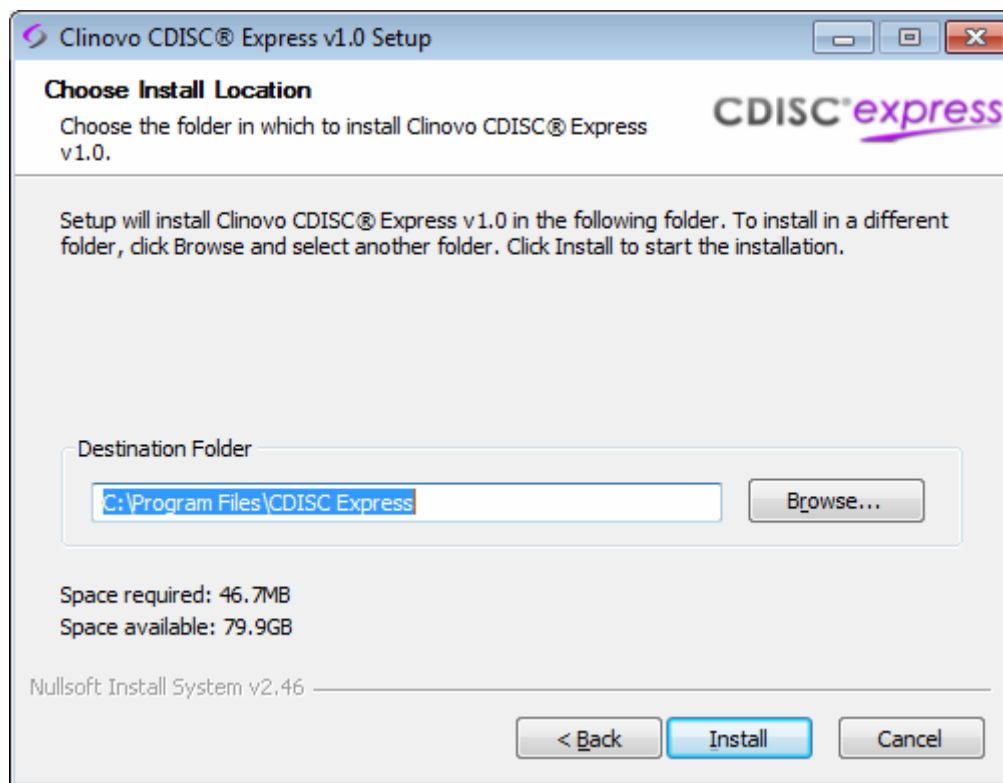
## 1. Download and Installation

You can get CDISC Express for free in

<http://www.clinovo.com/cdisc/download>

It is a window application and will be installed by default in

**C:\Program Files\CDISC Express\**



After installation, this path will be coded as a macro variable &CDISCPATH in the following six SAS files which are all located in **C:\Program Files\CDISC Express\programs\**:

```
create_new_study.sas
generate_Definexml.sas
generate_mapping_template.sas
generate_SDTM.sas
Validate_Mapping_File.sas
Validate_SDTM_Domains.sas
```

The macro variable reads as

```
%LET CDISCPATH = C:\Program Files\CDISC Express;
```

If you change the destination folder at the installation stage, e.g., to **D:\CDISC Express\**, the value of the macro variable &CDISCPATH will be changed accordingly in the six files mentioned before:

```
%LET CDISCPATH = D:\CDISC Express;
```

Note that if you want copy the whole folder of files to another destination, you should at least manually change the value of &CDISCPATH in such six files or add some codes to capture the path accordingly. From this point of view, the setting of path of CDISC Express is not that portable. Recommend that if you have such needs, just re-install the software in any destination you want. It will not write any records into registry and you can have many copies in one machine.

The following discussion assumes the software roots in **C:\Program Files\CDISC Express\**.

## 2. Working Flow


You can follow all the 6 action steps one by one coded in

**C:\Program Files\CDISC Express\programs\**

- 1) Create a new study (*create\_new\_study.sas*)

Simple and easy. Just assign a new study name in a macro call and run.

- 2) Generate mapping file (*generate\_mapping\_template.sas*)

This is the critical part and most time consuming. You should design mapping rules for every domain needed in Excel spreadsheets (the MAPPING FILE). If done, all other tasks, such as generate SDTM datasets, SAS transport files, define.xml and validation, can be well done by just clicking buttons .

- 3) Validate mapping file (*Validate\_Mapping\_File.sas*)

For validating the mapping file, just click the button. As mentioned, the most important work is designing the mapping file. It would be back and forth to design mapping file and validate it.

- 4) Generate SDTM datasets (*generate\_SDTM.sas*)

If mapping file is OK, click the button.

5) Validate SDTM datasets (*Validate\_SDTM\_Domains.sas*)

Click the button.

6) Generate Define.xml and SAS transport files (*generate\_Definexml.sas*)

Click the button.

### 3. Step 1 of 6: Create a new study (*create\_new\_study.sas*)

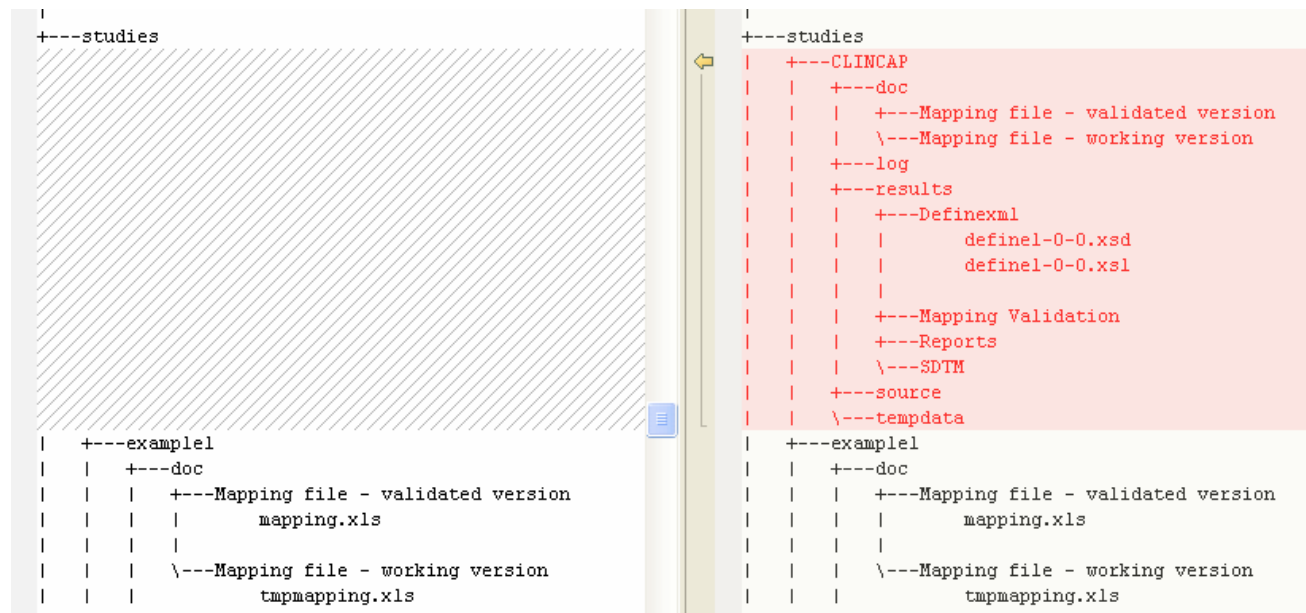
Open *create\_new\_study.sas* in **C:\Program Files\CDISC Express\programs\**, you can see only one line of a macro call:

```
%addnewstudy(studyname=my new study);
```

Just assign a study name to the macro variable, &studyname, e.g. “CLINCAP”:

```
%addnewstudy(studyname= CLINCAP);
```

Submit the codes, you can find a folder named “CLINCAP” with the same structure as the two demo studies imbedded in this application(example1 and example2) in **C:\Program Files\CDISC Express\studies\**, see(the left and right panels are folders and files before and after the execution of *create\_new\_study.sas*. The following the same):



Folder ‘doc’ is used to hold the mapping files;

Folder ‘log’ used to hold log files generated by following macro calls, such as generating SDTM domains;

Folder ‘results’ and its subfolder will hold all the outputs, such as define.xml, SAS transport file, validation reports and SDTM datasets;

Folder ‘source’ holds all the clinical raw data used as inputs for SDTM domains;

Folder ‘tempdata’ holds all the temporary datasets generated by following macro calls.

Also, a configuration file named *CLINCAP\_configuration.sas* put in **C:\Program Files\CDISC Express\programs\study configuration\**. This file is used to set some study level parameters, such as lab and toxicity specifications (details in **C:\Program Files\CDISC Express\specs\Lab specs\**).

Two versions of SDTM implementation guides are supported by CDISC Express, CDISC SDTM Implementation Guide Version 3.1.1 and Version 3.1.2. You can find the corresponding specification files in **C:\Program Files\CDISC Express\specs\SDTM specs\**:

SDTM\_Specs\_3\_1\_1.xls  
SDTM\_Specs\_3\_1\_2.xls

The choosing of SDTM implementation version is also coded in the configuration file, in Line 41:

```
%LET SDTMSPECFILE=SDTM_Specs_3_1_1.xls;
```

Version 3.1.1 is used by default. You can also choose Version 3.1.2 if needed:

```
%LET SDTMSPECFILE=SDTM_Specs_3_1_2.xls;
```

Assign a study name and choose a SDTM implementation version. That’s all needed in step 1. Let’s take few minutes to navigate the software. CDISC Express is a set of macros and Excel files. It is important to know the file structure first.

## C:\Program Files\CDISC Express\

—documentation	:FAQ, Quick Start, User Guide
—macros	
—ClinMap	:system level macros
—function_library	:study level macros
—programs	: "action taken" macros
—study configuration	:study parameters configuration, e.g, choose SDTM version
—SDTM Validation	:For validation of SDTM domains
—study1	
—specs	:specification files
—Excel engine	:ExcelXP tagset file
—Lab specs	:lab and toxicity
—Mapping validation	:validation rules
—SDTM specs	:hold two versions of SDTM implementation
—SDTM Terminology	:SDTM codelist(including NCI terminology)
—studies	
—example1	
—temp	:hold temporary data not specified to any studies

As we already got, all the “action taken” programs such as *create\_new\_study.sas* are located in **C:\Program Files\CDISC Express\programs\**. In *create\_new\_study.sas*, one macro is called, *%addnewstudy*, which is in **C:\Program Files\CDISC Express\macros\ClinMap\**.

Note that in **C:\Program Files\CDISC Express\macros\**, there are two sets of macros in different folders:

**C:\Program Files\CDISC Express\macros\ClinMap\**: this folder holds all “system” level macros used by the application only. No modification encouraged.

**C:\Program Files\CDISC Express\macros\function\_library\**: macros used for mapping among studies. You can also create you own macro in this folder. The application imbedded macros also documented in user guide.

#### 4. step 2 of 6: Generate mapping file

Generating template (blank) mapping file only needs pieces of effort by submitting *generate\_mapping\_template.sas*. The toughest one is to fill it with mapping rules according to specified study.

#### 4.1 Get the blank template mapping file (generate\_mapping\_template.sas)

To get the blank template mapping file, just fill the one line of macro call in generate\_mapping\_template.sas:

```
%createmapping(filespec=SDTM_Specs_3_1_1.xls, Dom=CM AE TV,  
req=YES, perm=YES, exp=YES);
```

Also, you can specify SDTM implementation version, 3.1.1 or 3.1.2. For domains (&Dom), DM, CO and SUPPQUAL will be created automatically; you should list others accordingly:

```
SDTM 3.1.1: SV CM EX AE DS MH DV EG IE LB PE QS SC VS TV TI XD SU XR  
XS XE TR (Total: 22)
```

```
SDTM 3.1.2: AE CE CM DA DS DV EG EX FA IE LB MB MH MS PC PE PP QS SC  
SE SU SV TA TE TI TS TV VS (Total: 28)
```

You should also choose the “CORE” variable (REQUIRED, PERMISSIBLE and EXPECTED) by triggering &req, &perm, and &exp to “YES” or “NO”. Note that

```
REQUIRED and EXPECTED variables must always be included (req=YES, exp=YES);  
PERMISSIBLE variables included if needed (perm=YES or perm=NO)
```

Submit *generate\_mapping\_template.sas* and you can get a blank template mapping file tmpmapping.xls in **C:\Program Files\CDISC Express\temp\**.

	A	B	C	D	E	F
1	Dataset	Merge Key	CDISC variable	Expression	Comments	
2			STUDYID			
3			DOMAIN			
4			USUBJID			
5			SUBJID			
6			RFSTDTC			
7			RFENDTC			
8			SITEID			
9			INVID			
10			INVNAM			
11			BRTHDTC			
12			AGE			
13			AGEU			
14			SEX			
15			RACE			
16			ETHNIC			
17			ARMCD			
18			ARM			
19			COUNTRY			
20			DMDTC			
21			DMDY			
22						
23						

Copy it to **C:\Program Files\CDISC Express\studies\CLINCAP\doc\Mapping file - working version** for example used for study “CLINCAP” and then fill all the blank columns (it NEEDS efforts!).

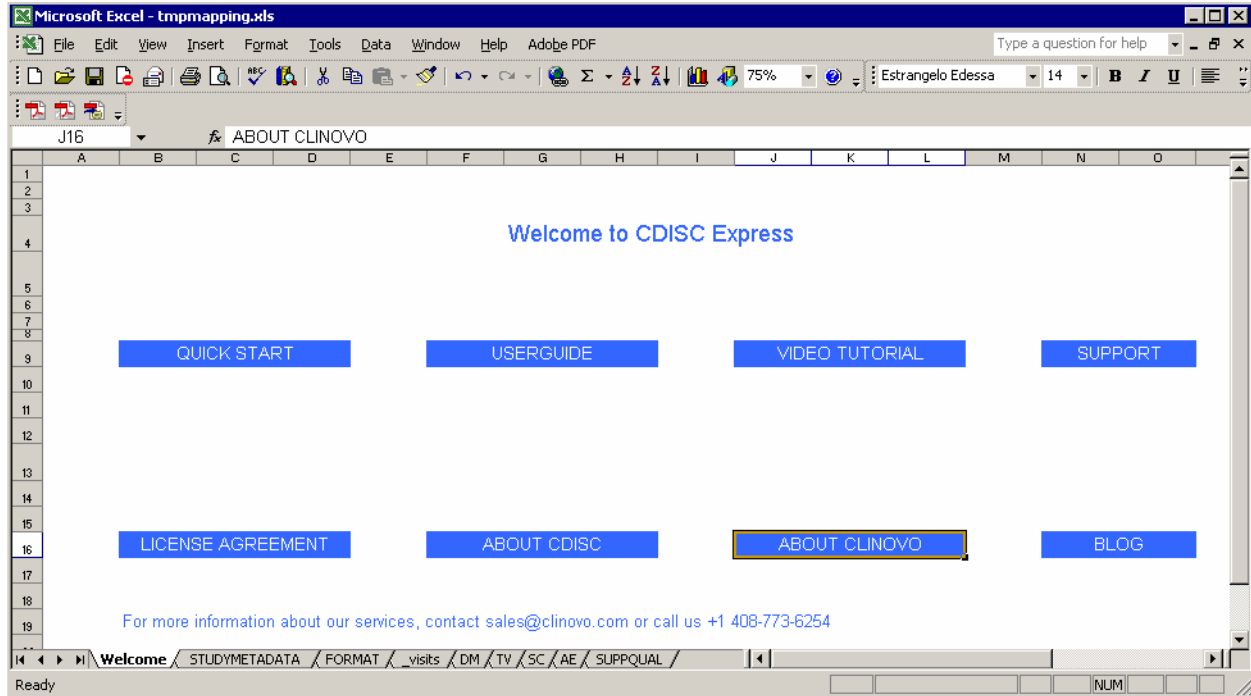
If this mapping file passes the validation process, a final version named mapping.xls will be copied automatically to **C:\Program Files\CDISC Express\studies\CLINCAP\doc\Mapping file - validated version\** for later processing.

Note that if you already have some validated mapping file for other studies, it would serve as a good start rather than using the blank template from the scratch.

#### 4.2 Navigate mapping file

Let’s take a look at the “real” worked mapping file for a demo study first, in **C:\Program Files\CDISC Express\studies\example1\doc\Mapping file - working version\**.

The first sheet is a welcome dashboard:



Then StudyMetadata sheet, a XML metadata specification used to generate define.xml. you need only add some information in “Values” column:

The screenshot shows a Microsoft Excel window titled "Microsoft Excel - tmpmapping.xls". The active sheet is "XMLField". The table content is as follows:

1	A	B	C	D	E
	XMLField	XMLElement	Status	Values	Comments
2	ODM Attributes	FileType	Required	Snapshot	
3		FileOID	Required		
4		PriorFileOID	Optional		Reference to the previous file (if any)
5		ODMVersion	Required		
6		Originator	Optional		The organization that generated the define.xml file
7		SourceSystem	Optional		The computer system, database management system, etc. that is the source of the define.xml information
8		SourceSystemVersion	Optional		The version of "SourceSystem" above
9		CreationDateTime	Required		** Do not fill with any value
10	ODM Child Element	Study	Required		
11	Study Attributes	OID	Required		
12	Study Child Elements	GlobalVariables	Required		
13		MetadataVersion	Required		
14	Global Variable Child Elements	StudyName	Required		Name of Study
15		StudyDescription	Required		Description of Study

The FORMAT sheet:

	A	B	C
1	format	from	tovalue
2	\$action_taken		
3		NONE	DOSE NOT CHANGED
4		DRUG PERMANENTLY DISCONTINUED	DRUG WITHDRAWN
5		DOSE INCREASED	DOSE INCREASED
6		DOSE HELD	DRUG INTERRUPTED
7		DOSE DECREASED	DOSE REDUCED
8		INFUSION INTERRUPTED	DRUG INTERRUPTED
9		INFUSION RATE DECREASED	NA
10	\$yn		
11		YES	Y
12		NO	N
13		UNK	U

Such format structure is similar with the one we export the format from a format catalog using

```
proc format library=library cntlout=format_out;
run;
```

In most production environment, programmers get formats from clinical data management group. If the entire formats are assigned into proper libraries (work or library), you don't need to export such formats into this spreadsheet. Of course in the format sheet, you can type some customized format.

A typical domain sheet (AT LAST!) that needs efforts and our understanding of the software, DM for example:

	A	B	C	D
	Dataset	Merge Key	CDISC variable	Expression
1				
2	demog	sitecode	STUDYID	study
3			DOMAIN	&domain
4			USUBJID	%CONCATENATE(_variables=study sitecode patid)
5			SUBJID	patid
6			RFSTDTC	%D_RFST(_dataset=trinf,_date=trinfdt,_key=patid,_ivrsds=ivrs,_ivrsdt=randat)
7			RFENDTC	%SORTLOOKUP(_dataset=disc,_variable=fupdat,_key=patid,_sort_variable=fupdat,_keep=last)
8			SITEID	sitecode
9			INVID	incode
10			BRTHDTC	%FORMAT(_variable=brthdat,_format=yymmdd10)
11			SEX	%GENDER(_gender=sex)
12			RACE	%D_RACE()
13			ETHNIC	upcase(ethnic)
14			COUNTRY	"USA"
15			DMDTC	%FORMAT(_variable=formdat,_format=yymmdd10)
16			ARMCD	""
17			ARM	""
18	siteinv(drop=incode)	sitecode	INVNAM	trim(lname)  " "  fname
19			COUNTRY	"USA"
20	eligassess		USUBJID	%CONCATENATE(_variables=study sitecode patid)
21	all		AGE	year(infcondt)-year(brthdat)-(month(brthdat)>month(infcondt))- (month(brthdat)=month(infcondt) and day(brthdat)>day(infcondt))
22			AGEU	%CONVERTIF(_if_variable=@AGE,_if_value=._then_value=, else_value=YEARS)

From the 'Dataset' column, three raw datasets from **C:\Program Files\CDISC Express\studies\example1\source\** needed to map into DM domain, demog, siteinv and eligassess. Note that you can use any data step options such as drop=, rename=, where= for the input datasets.

At the last of 'Dataset' column, "all" indicates that all the previous datasets mentioned above should be merged together for final processing.

In the 'Merge Key' column, 'sitecode' is designed to datasets demog and siteinv which means demog and siteinv should be merged by the common key, 'sitecode'.

As we mentioned, all the previous datasets should be merged at last. But there is no common key settled in the 'Merge Key' column. It is a common rule: if no key specified for merge, USUBJID is used by default.

The third column is 'CDISC variable', which list all the needed variables according to implementation version. An important note: you do not need to implement all the variables

according to the order as they appear in the blank template mapping file. In the previous blank file, “AGE” in DM domain is ordered in Line 12, but in this working file, “AGE” is calculated in the second last order. The variable order of final DM domain will be as same as the blank one.

It makes sense in practice. For example, the sequential variable, e.g. AESEQ is ordered after USUBJID, but you can only get the sequential number when all other variables well done. So SEQ variables are always computed in the final stage in a working mapping file.

“Expression” column specify the mapping rule from raw datasets to SDTM domains. Assignments, expressions and macro calls (rooted in **C:\Program Files\CDISC Express\macros\function\_library\**) are allowed in this column and most of them are straightforward. We will discuss more in the following section.

Sum up, we can “translate” this mapping sheet to SAS codes for better understanding of CDISC Express architecture:

```
data tem1;
    set demog;
    STUDYID=study;
    DOMAIN = &domain;
    USUBJID=%CONCATENATE(_variables=study sitecode patid);
    SUBJID =patid;

RFSTDTC=%D_RFST(_dataset=trtinf,_date=trtinfdt,_key=patid,_ivrsd
s=ivrs,_ivrsdt=randdat);

RFENDTC=%SORTLOOKUP(_dataset=disc,_variable=fupdat,_key=patid,_s
ort_variable=fupdat,_keep=last);
    SITEID =sitecode;
    INVID =invcode;
    BRTHDTC=%FORMAT(_variable=brthdat,_format=yymmdd10);
    SEX =%GENDER(_gender=sex);
    RACE =%D_RACE();
    ETHNIC =upcase(ethnic);
    COUNTRY="USA";
    DMDTC =%FORMAT(_variable=formdat,_format=yymmdd10);
    ARMCD = " ";
    ARM = " ";

run;

data tem2;
    merge tem1 siteinv(drop=invcode);
    by sitecode;
    INVNAM=trim(lname) || ", " || fname;

run;

data dm;
```

```

merge tem2 eligassess;
by patid;
AGE =year(infcondt)-year(brthdat)-
(month(brthdat)>month(infcondt))-(month(brthdat)=month(infcondt)
and day(brthdat)>day(infcondt));
AGEU=%CONVERTIF(_if_variable=@AGE,_if_value=.,_then_value=,
_else_value=YEARS);
run;

```

### 4.3 Data manipulation techniques in CDISC Express

CDISC Express supplies relative rich sets of data manipulation techniques assembling with SAS languages used for data mapping. Following is a not limited listing and I will keep it updated.

#### 4.3.1 Reference one dataset

A raw dataset name appear in “Dataset” column indicate a “set” operation in SAS.

All dataset options can be used when referencing a dataset, such as

```

siteinv(drop=invcode)
siteinv(rename=(invcode=inv))
siteinv(where=(invcode ne ""))

```

You can also reference an external dataset. You should incorporate the external file in spreadsheet with name beginning with an underscore, “\_”, and “\_visits” in this case:

	A	B	C	D	E
1	<b>VISITNUM</b>	<b>VISIT</b>	<b>VISITDY</b>		
2	1	SCRN	-1		
3	2	VIS1	1		
4	3	VIS2	10		
5	4	VIS3	24		
6	5	VIS4	38		
7	6	VIS5	50		
8	7	VIS6	68		
9					

Then you can use it in any domains needed, e.g., TV domain:

	A	B	C	D
1	Dataset	Merge Key	CDISC variable	Expression
2	%cpd_importlist(dataset=temptv, worksheet=_visits)		STUDYID	"CLIN123"
3			DOMAIN	&domain
4			VISITNUM	visitnum
5			VISIT	visit
6			VISITDY	visitdy
7			ARMCD	"DrugA"
8			ARM	"DrugA: A Receptor"
9			TVSTRL	""
10			TVENRL	""
11				

There is a macro %cpd\_importlist used to import the external dataset, “\_visits”. Again, this macro roots in **C:\Program Files\CDISC Express\macros\function\_library\**.

Using a macro call to re-sharp or modify an input dataset offers great flexibility referencing data. We will also discuss the benefits later on.

#### 4.3.2 Assignment

You can assign a number, string and a dataset variable with any valid SAS functions to a SDTM domain variable in “Expression” column.

Sometimes a temporary variable needed for later calculation. You can produce such temporary variable in “Dataset” column with an assignment in the “Expression” column just similar with any other domain variables. Two differences: first, such temporary variables named begin with an asterisk, “\*”; second, all temporary variables will not be included in the final domain. Once created, such temporary variables can be used for any other expressions.

Dataset	Merge Key	CDISC variable	Expression
eligassess	patid	USUBJID	%CONCATENATE(_variables=study sitecode patid)
eligcrit	patid	IETESTCD	eligcritno
		IETEST	%FORMAT(_variable=eligcritno,_format=ie_desc)
		*tmp_ie	substr(eligcritno,1,1)
		IECAT	%CONVERTIF(_if_variable=*tmp_ie,_if_value=E,_then_value=EXCLUSION)
		IECAT	%CONVERTIF(_if_variable=*tmp_ie,_if_value=I,_then_value=INCLUSION)
		IEORRES	%CONVERTIF(_if_variable=*tmp_ie,_if_value=I,_then_value=N)
		IESTRESC	%CONVERTIF(_if_variable=*tmp_ie,_if_value=I,_then_value=N)
		IEDTC	%FORMAT(_variable=formdat,_format=yymmdd10)
		IEDY	%STUDYDAY(_date=formdat)
		USUBJID	%CONCATENATE(_variables=study sitecode patid)
all		STUDYID	study
		DOMAIN	&domain
		IESEQ	%SEQUENCE()

There are three special symbols used in “Dataset” column of CDISC Express. Asterisk, “\*” indicates a temporary variable, while other two are

Tilde, “~” : indicate a variable used for supplemental domain (SUPPQUAL).

Number sign, “#”: indicate a variable used for comments domain (CO).

Another symbol, at sign, “@”, used in “Expression” column, indicated referencing a variables produced before:

AGE	year(infcondt)-year(brthdat)-(month(brthdat)>month(infcondt))- (month(brthdat)=month(infcondt) and day(brthdat)>day(infcondt))
AGEU	%CONVERTIF(_if_variable=@AGE,_if_value=._then_value=._else_value=YEARS)

In this case, “AGEU” uses “AGE” as input, while “AGE” is calculated before. “@AGE” just indicates the dependency. In concept, it looks like the “calculated” option in SAS PROC SQL:

```

proc sql ;
  select (AvgHigh - 32) * 5/9 as HighC ,
         (AvgLow  - 32) * 5/9 as LowC  ,
         (calculated HighC - calculated LowC)
         as Range
  from temps;
quit;

```

### 4.3.3 Match-merging

We already got a math-merging example before. If “all” appears as a dataset in the “Dataset” column, all the previous datasets should be merged first for later processing by the common key specified in “Merge Key” column. If no key assigned, patient ID is used by the system.

CDISC Express also supports two types of join, inner join and outer join (left, right, full) using data steps. The implementation has slightly difference with standard SQL, but the ideas are same.

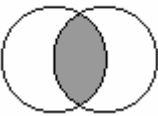
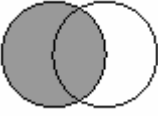
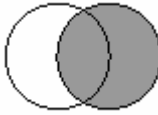
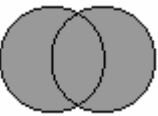
We add a new column, “Join”, usually beside the “Merge Key” column.

	A	B	C	D	
1	Dataset	Merge Key	Join	CDISC variable	
2	demog	sitecode	O	STUDYID	study
3				DOMAIN	&domain
4				USUBJID	%CONCATENATE( _variables=study :
5				SUBJID	patid
6				RFSTDTC	%D_RFST( _dataset=trtin, _date=trtin
7				RFENDTC	%SORTLOOKUP( _dataset=disc, _var
8				SITEID	sitecode
9				INVID	incode
10				BRTHDTC	%FORMAT( _variable=brthdat, _format
11				SEX	%GENDER( _gender=sex)
12				RACE	%D_RACE()
13				ETHNIC	upcase(ethnic)
14				COUNTRY	"USA"
15				DMDTC	%FORMAT( _variable=formdat, _forma
16				ARMCD	""
17				ARM	""
18	siteinv(drop=incode)	sitecode	I	INVNAM	trim(lname)  " "  fname
19				COUNTRY	"USA"
20	eligassess			USUBJID	%CONCATENATE( _variables=study :

There are two values for “Join”, “O” or “I” while “O” stands for “outer join” and “I”, “inner join”. A join indicator “I” equals a dataset option “in=” in action while “O” means no. Use the above as illustration, the corresponding SAS codes behind look like

```
data temp;
  merge demog(in=a) siteinv(in=b);
  by sitecode;
  if b;
run;
```

This is so called “right outer join”. The combination of “I” and “O” in these two datasets can perform all the four types of join, one inner join and three outer join:

Join type		SAS codes behind CDISC Express (illustration)	combination
Inner join		<pre>data inner; merge demog(in=a) siteinv(in=b); by sitecode; if a and b; run;</pre>	(I,I)
Outer join	Left outer join 	<pre>data left; merge demog(in=a) siteinv(in=b); by sitecode; if a; run;</pre>	(I,O)
	Right outer join 	<pre>data right; merge demog(in=a) siteinv(in=b); by sitecode; if b; run;</pre>	(O,I)
	Full outer join 	<pre>data full; merge demog(in=a) siteinv(in=b); by sitecode; run;</pre>	(O,O)

As we could see, if no “Join” column specified, CDISC Express will perform inner join by default.

So far CDISC Express cannot support multiply merge keys. For example, the following file is illegal currently:

Dataset	Merge Key
arm	siteid, grpno
armdescri	siteid, grpno

The developer Romain indicated that such enhancements would be raised to the next round of product road map and he also proposed a work around. To use multiple keys for merging, we can

create a temporary variable holding such multiple keys as a concatenation then this temporary variable can be used as a single merging key.

#### 4.3.4 Concatenating

Above we discussed lots about “merge” operation in CDISC Express. This section dedicated for “set” operation. We already know how to “set” one dataset for referencing, but how to “set” multiple datasets, i.e, “Concatenating”?

Symmetrically, an “all” appears in “Dataset” column indicating merging operation, an “all (stack)” indicates concatenating operation:

	A	B	C	
1	Dataset	CDISC variable	Expression	Co
42	vsigns(where=(height ne .))	VSTESTCD	"HEIGHT"	
43		VSTEST	"Height"	
44		VSORRES	put(height,best12.)	
45		VSORRESU	"cm"	
46		VSSTRESC	put(height,best12.)	
47		VSSTRESN	height	
48		VSSTRESU	"cm"	
49		VSSTAT	%CONVERTIF(_if_variable=vitalsyn,_if_value=No,_then_value=NOT DONE)	
50	vsigns(where=(weight ne .))	VSTESTCD	"WEIGHT"	
51		VSTEST	"Weight"	
52		VSORRES	put(weight,best12.)	
53		VSORRESU	"kg"	
54		VSSTRESC	put(weight,best12.)	
55		VSSTRESN	weight	
56		VSSTRESU	"kg"	
60	all(stack)	STUDYID	study	
61		DOMAIN	&domain	
62		USUBJID	%CONCATENATE(_variables=study sitecode patid)	
63		VSSEQ	%SEQUENCE()	
64		VISITNUM	%VISITLOOKUP(_variable=visitnum,_key=foldercd vitaldt)	

The above file can be also translated to SAS codes for better understanding:

```

data height;
  set vtsigns(where=(height ne .));
  VSTESTCD="HEIGHT";
  VSTEST   ="Height";
  VSORRES  =put(height,best12.);
  VSORRESU="cm";
  VSSTRESC=put(height,best12.);
  VSSTRESN=height;
  VSSTRESU="cm";
run;

data weight;
  set vtsigns(where=(weight ne .));
  VSTESTCD="WEIGHT";
  VSTEST   ="Weight";
  VSORRES  =put(weight,best12.);
  VSORRESU="kg";
  VSSTRESC=put(weight,best12.);
  VSSTRESN=weight;
  VSSTRESU="cm";
run;

data vs;
  set height weight;
  STUDYID =study;
  DOMAIN  =&domain;
  USUBJID =%CONCATENATE(_variables=study sitecode patid);
  VSSEQ   =%SEQUENCE();
  . . .
run;

```

### 4.3.5 Transpose

Clinical SAS programmers do lots of transpose operation to re-sharp the raw data to fit the CDISC standards. Currently there is no explicit guide in CDISC Express on how to transpose, but this is not the end of story.

There are two types of transpose:

	A	B	C	D	E	F	G	H	I	J	K
1		March	April	May	June	July	August	September	October	November	December
2	Jeff	1	4	7	3	4	3	8	5	8	4
3	Fred	2	5	8	5	8	4	9	6	9	5
4											
5											
6						Jeff	Fred				
7					March	1	2				
8					April	4	5				
9					May	7	8				
10					June	3	5				
11					July	4	8				
12					August	3	4				
13					September	8	9				
14					October	5	6				
15					November	8	9				
16					December	4	5				

Type I: from a wide dataset (more variables, less observations) to a long dataset (less variables, more observations), e.g. transposing a one-row-per-subject datasets to a multiple-row-per-subject dataset

	A	B	C	D	E	F	G	H	I	J	K
1						Jeff	Fred				
2					March	1	2				
3					April	4	5				
4					May	7	8				
5					June	3	5				
6					July	4	8				
7					August	3	4				
8					September	8	9				
9					October	5	6				
10					November	8	9				
11					December	4	5				
12											
13											
14		March	April	May	June	July	August	September	October	November	December
15	Jeff	1	4	7	3	4	3	8	5	8	4
16	Fred	2	5	8	5	8	4	9	6	9	5
17											

Type II: from a long dataset (less variables, more observations) to a wide dataset (more variables, less observations), e.g. transposing a multiple-row-per-subject dataset to a one-row-per-subject datasets

As good practices, in SAS we always use data steps with “output” statement to perform type I transpose and use PROC TRANSPOSE for type II. Although CDISC Express doesn’t support transpose operation in an explicit way, at least you can perform type I transpose and surprisingly we already saw it before!

Just back to section of concatenating. The example is taken from **C:\Program Files\CDISC Express\studies\example2\**,

	A	B	C	Co
1	<b>Dataset</b>	<b>CDISC variable</b>	<b>Expression</b>	
42	vtsigns(where=(height ne .))	VSTESTCD	"HEIGHT"	
43		VSTEST	"Height"	
44		VSORRES	put(height,best12.)	
45		VSORRESU	"cm"	
46		VSSTRESC	put(height,best12.)	
47		VSSTRESN	height	
48		VSSTRESU	"cm"	
49		VSSTAT	%CONVERTIF(_if_variable=vitalsyn,_if_value=No,_then_value=NOT DONE)	
50	vtsigns(where=(weight ne .))	VSTESTCD	"WEIGHT"	
51		VSTEST	"Weight"	
52		VSORRES	put(weight,best12.)	
53		VSORRESU	"kg"	
54		VSSTRESC	put(weight,best12.)	
55		VSSTRESN	weight	
56		VSSTRESU	"kg"	
60	all(stack)	STUDYID	study	
61		DOMAIN	&domain	
62		USUBJID	%CONCATENATE(_variables=study sitecode patid)	
63		VSSEQ	%SEQUENCE()	
64		VISITNUM	%VISITLOOKUP(_variable=visitnum,_key=foldercd vitaldt)	

```

data height;
  set vtsigns(where=(height ne .));
  VSTESTCD="HEIGHT";
  VSTEST   ="Height";
  VSORRES  =put(height,best12.);
  VSORRESU="cm";
  VSSTRESC=put(height,best12.);
  VSSTRESN=height;
  VSSTRESU="cm";

```

run;

```

data weight;
  set vtsigns(where=(weight ne .));
  VSTESTCD="WEIGHT";
  VSTEST   ="Weight";
  VSORRES  =put(weight,best12.);
  VSORRESU="kg";
  VSSTRESC=put(weight,best12.);
  VSSTRESN=weight;

```

```

VSSTRESU="cm";
run;

data vs;
  set height weight;
  STUDYID =study;
  DOMAIN  =&domain;
  USUBJID =%CONCATENATE(_variables=study sitecode patid);
  VSSEQ   =%SEQUENCE();
  . . .
run;

```

We can see the input data vtsigns is typical wide table (more variables, less observations) :

	Patient ID (PATID)	Folder Name (FOLDERNAME)	Temperature, C (TEMP)	Pulse, Heart Rate, min (PULSE)	Respiration, Rate, min (RESPRATE)	Blood Pressure, Systolic, mmHg (SYSTBP)	Blood Pressure, Diastolic, mmHg (DIASBP)	Weight, kg (WEIGHT)	Height, cm (HEIGHT)
1	1	Screening	36.4	62	22	162	65	83.6	152.4
2	1	Visit 1	36.7	69	18	154	62	82.2	.
3	1	Visit 2	36.5	78	18	148	72	81.3	.
4	1	Visit 3	37.6	76	18	154	67	77.2	.
5	1	Visit 4	36.7	85	18	141	65	81.8	.
6	1	Visit 5	36.1	89	18	157	72	80.4	.
7	1	Visit 6	36.3	79	16	132	63	81.3	.
8	1	Visit 7	36.6	70	16	158	65	80.9	.
9	1	Visit 8	36.2	67	18	159	72	80.9	.
10	1	Visit 9	36.8	68	18	150	72	80.9	.
11	1	Visit 10	36.4	84	18	128	72	80.9	.
12	1	Visit 11	36.3	80	18	132	88	79.5	.
13	1	Visit 12	36.4	68	18	132	80	80.4	.
14	1	Visit 13	36.7	70	18	150	80	80	.
15	1	Visit 14	36.3	72	18	152	84	80	.
16	1	Visit 15	37	79	19	130	80	80	.
17	1	Visit 16	36.9	80	16	152	80	79.5	.
18	1	Visit 17	36.7	80	18	203	94	78.1	.
19	1	Visit 18	36.9	76	18	137	58	77.7	.
20	1	Visit 19	36.3	96	18	150	75	78.6	.

And the final domain VS is a typical long table (less variables, more observations):

	Unique Subject Identifier (USUBJID)	Sequence Number (VSSEQ)	Vital Signs Test Short Name (VSTESTCD)	Result or Finding in Original Units (VSORRES)
1	CLIN123-S001-1	1	TEMP	36.4
2	CLIN123-S001-1	2	TEMP	36.7
3	CLIN123-S001-1	3	TEMP	36.5
4	CLIN123-S001-1	4	TEMP	37.6
5	CLIN123-S001-1	5	TEMP	36.7
6	CLIN123-S001-1	6	TEMP	36.1
7	CLIN123-S001-1	7	TEMP	36.3
8	CLIN123-S001-1	8	TEMP	36.6
9	CLIN123-S001-1	9	TEMP	36.2
10	CLIN123-S001-1	10	TEMP	36.8
11	CLIN123-S001-1	11	TEMP	36.4
12	CLIN123-S001-1	12	TEMP	36.3
13	CLIN123-S001-1	13	TEMP	36.4
14	CLIN123-S001-1	14	TEMP	36.7
15	CLIN123-S001-1	15	TEMP	36.3
16	CLIN123-S001-1	16	TEMP	37
17	CLIN123-S001-1	17	TEMP	36.9
18	CLIN123-S001-1	18	TEMP	36.7
19	CLIN123-S001-1	19	TEMP	36.9
20	CLIN123-S001-1	20	TEMP	36.3
21	CLIN123-S001-1	21	PULSE	62
22	CLIN123-S001-1	22	PULSE	69
23	CLIN123-S001-1	23	PULSE	78
24	CLIN123-S001-1	24	PULSE	76
25	CLIN123-S001-1	25	PULSE	85
26	CLIN123-S001-1	26	PULSE	89
27	CLIN123-S001-1	27	PULSE	79
28	CLIN123-S001-1	28	PULSE	70
29	CLIN123-S001-1	29	PULSE	67
30	CLIN123-S001-1	30	PULSE	68
31	CLIN123-S001-1	31	PULSE	84
32	CLIN123-S001-1	32	PULSE	80
33	CLIN123-S001-1	33	PULSE	68
34	CLIN123-S001-1	34	PULSE	70
35	CLIN123-S001-1	35	PULSE	72
36	CLIN123-S001-1	36	PULSE	79
37	CLIN123-S001-1	37	PULSE	80
38	CLIN123-S001-1	38	PULSE	80

So obviously, such concatenating operation just did a wonderful type I transpose, from a wide table to a long table! More often, the compact SAS codes for type I transpose look like:

```
data vs;
  set vtsigns;

  if height ne . then do;
    VSTESTCD="HEIGHT";
    VSTEST   ="Height";
    VSORRES  =put(height,best12.);
    VSORRESU="cm";
    VSSTRESC=put(height,best12.);
    VSSTRESN=height;
    VSSTRESU="cm";
    output;
  end;

  if weight ne . then do;
    VSTESTCD="WEIGHT";
    VSTEST   ="Weight";
    VSORRES  =put(weight,best12.);
    VSORRESU="kg";
    VSSTRESC=put(weight,best12.);
    VSSTRESN=weight;
    VSSTRESU="cm";
    output;
  end;

  . . .
run;
```

#### 4.3.6 All others: use macro!

Now we discussed almost all the common data derivation techniques in programmers' daily life and the corresponding implementation in CDISC Express. At least we have one question unsolved: how to perform type II transpose, i.e. from a long table to a wide table?

It would be an open question for the developers of the application. But we can also solve this problem in current framework: use macro, customized macro. You can use macros in "Expression" and "Dataset" column. Macro used in "Dataset" column returns a dataset, while macro in "Expression" column returns series of string: that's the basic structure you should consider when customize your own macros. For more, you can reference the macros in **C:\Program Files\CDISC Express\macros\function\_library\**. For example, &concatenate used in "Expression" column; &cpd\_importlist in "Dataset" column.

So it would be convenient to create temporary datasets using macros imbedded type II transpose operation in “Dataset” column. Every thing SAS can do, you can also implement it in CDISC Express. Just use macros, in “Expression” and “Dataset” column accordingly.

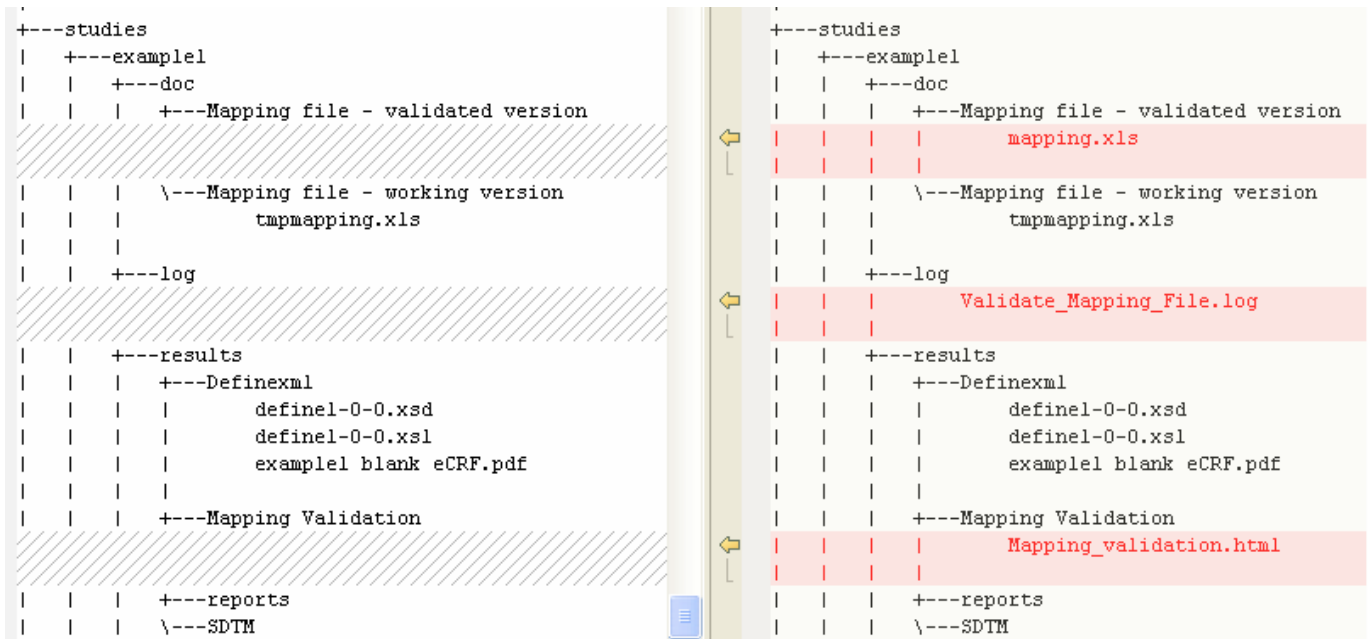
The raw data varies according to trial design and clinical data capture system and procedures. It is impossible and impractical to anticipate the CDISC SDTM converter such as CDISC Express to map all the data just clicking a button. The introducing of CDISC Express doesn’t keep programmers away. It just keeps most of the trivial work away from programmers’ daily life and let them more concentrated on creative work and be productive and efficient.

### 5. step 3 of 6: Validate mapping file (*Validate\_Mapping\_File.sas*)

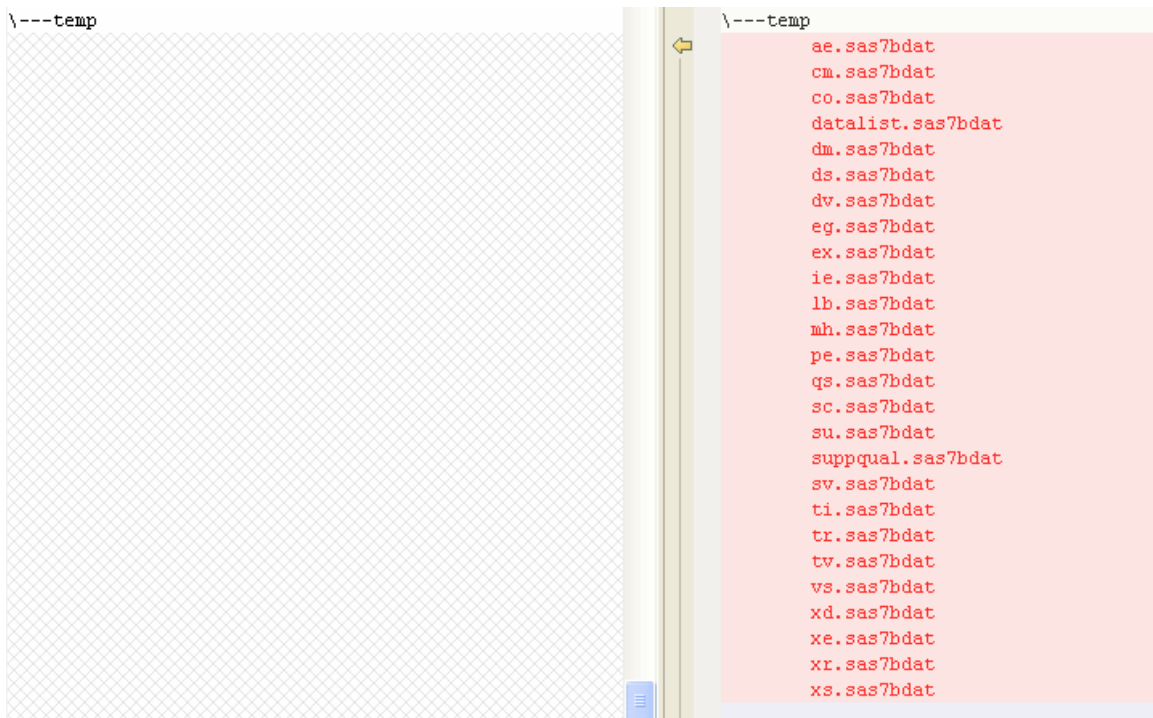
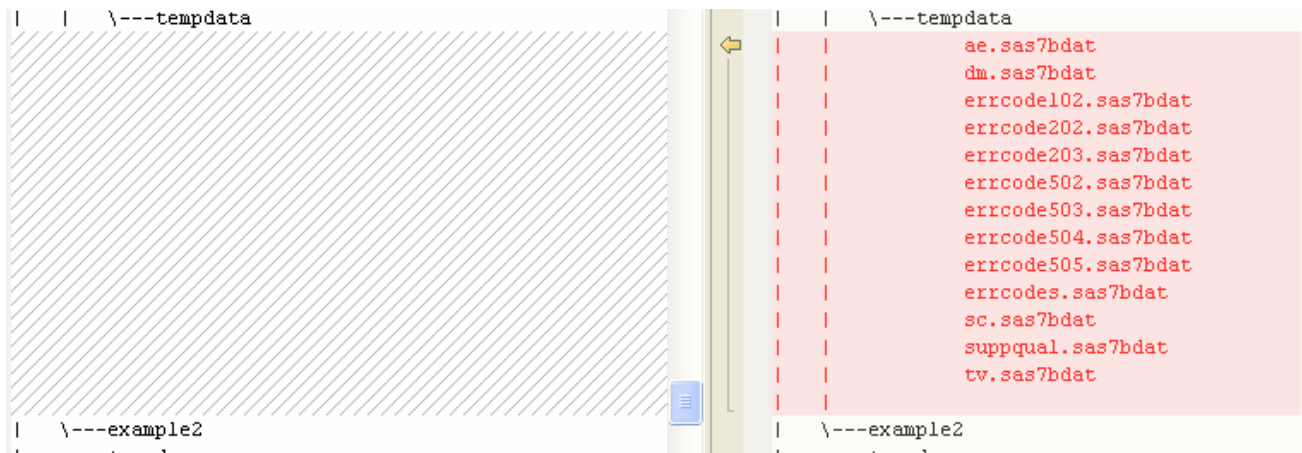
It would be back and forth to design, validate then modify and re-validate the mapping file. And sure finally, you will get all the work done, at least no syntax error (how to avoid semantic errors is upon your domain knowledge). A validated mapping file, named mapping.xls will be copied to ...\**doc\Mapping file - validated version**\ from the working file, tmpmapping.xls. You will see

The corresponding log file in folder ...\**log**\

A report in ...\**results\Mapping Validation**\, named Mapping\_validation.html



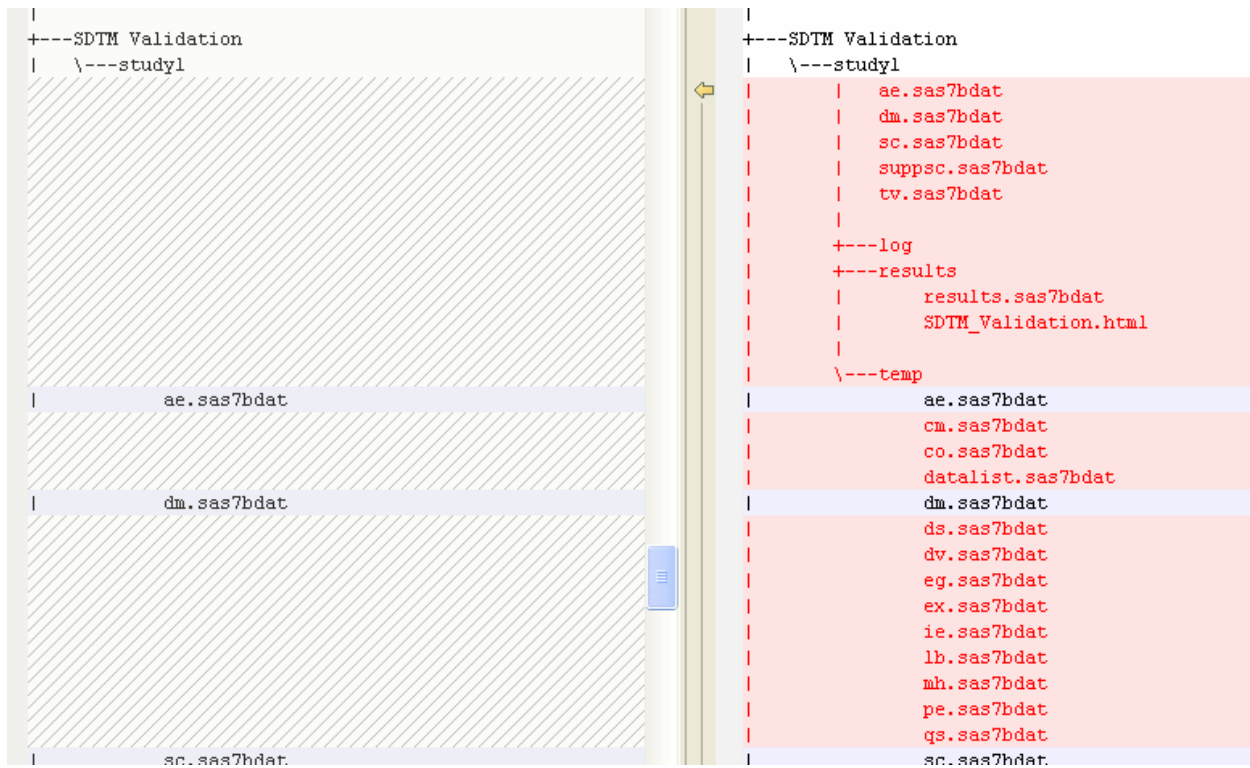
Also the temporary datasets in ...\**tempdata**\ and ...\**temp**:



**6. step 4 of 6: Generate SDTM datasets (*generate\_SDTM.sas*)**

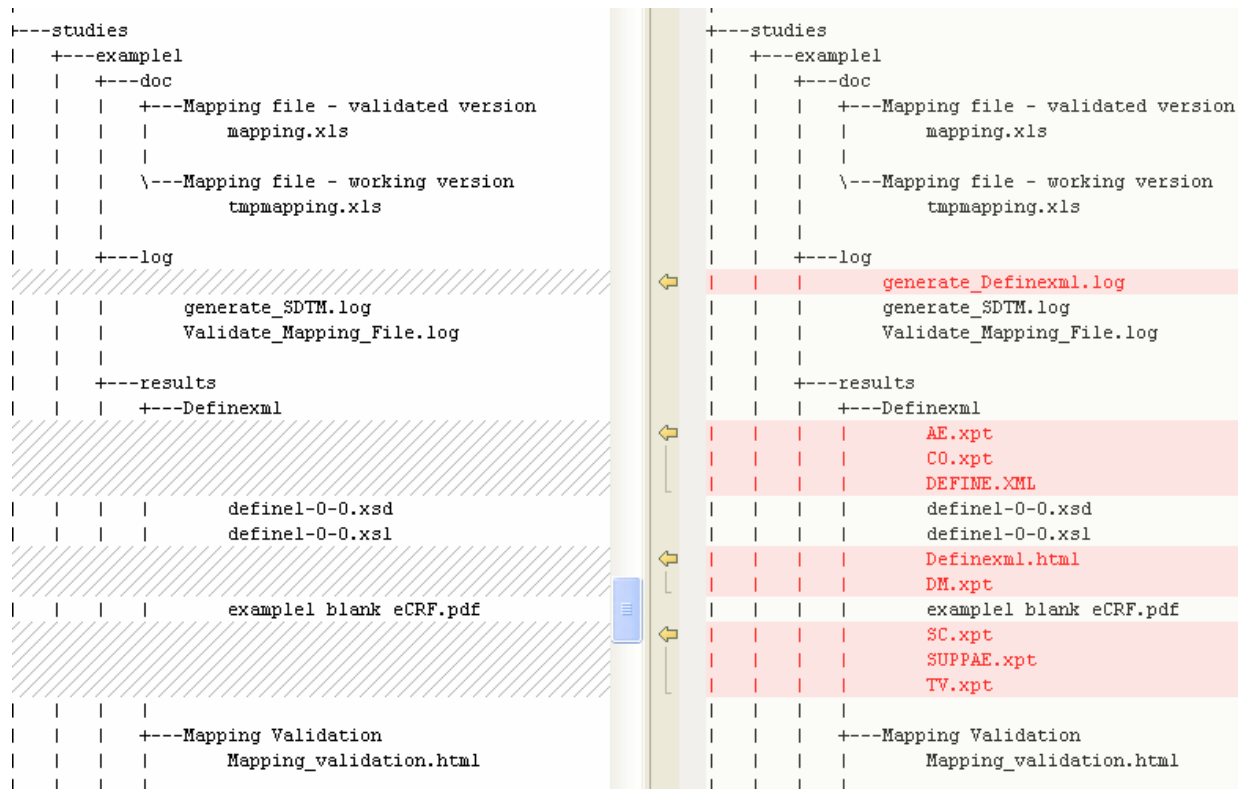
If mapping file is OK, generating SDTM domains is just clicking the button. After submitting the codes, you will see the log file, reports, SDTM datasets and temporary datasets in corresponding folders:





## 8. step 6 of 6: Generate Define.xml and xpt (*generate\_Definexml.sas*)

Get the final define.xml file and SAS transport files (.xpt):



## 9. Recommended reading and action taken

For a quick start and deep understanding, you could read the official documentations in the following sequence:

**C:\Program Files\CDISC Express\documentation\FAQ.htm**  
**C:\Program Files\CDISC Express\documentation\Quick Start.htm**  
**C:\Program Files\CDISC Express\documentation\User guide.htm**

A video tutorial would be also helpful:

**C:\Program Files\CDISC Express\documentation\videotutorial.htm**

A must-read conference paper, *An Excel Framework to Convert Clinical Data to CDISC SDTM Leveraging SAS Technology* by Sophie McCallum and Stephen Chan of Clinovo, supplies a wonderful discussion the architectures of CDISC Express:

<http://www.lexjansen.com/pharmasug/2011/ad/pharmasug-2011-ad08.pdf>

Of course, you do not need to review all the pages then get the confidence to use the software. Learn by doing and just dive into it. There is an opportunity for your practicing and you will also have a change to win an iPad2 from Clinovo's CDISC Express Contest:

<http://www.clinovo.com/cdisc/game>

The due day is July 15<sup>th</sup> and I already submitted my work. That's fun.