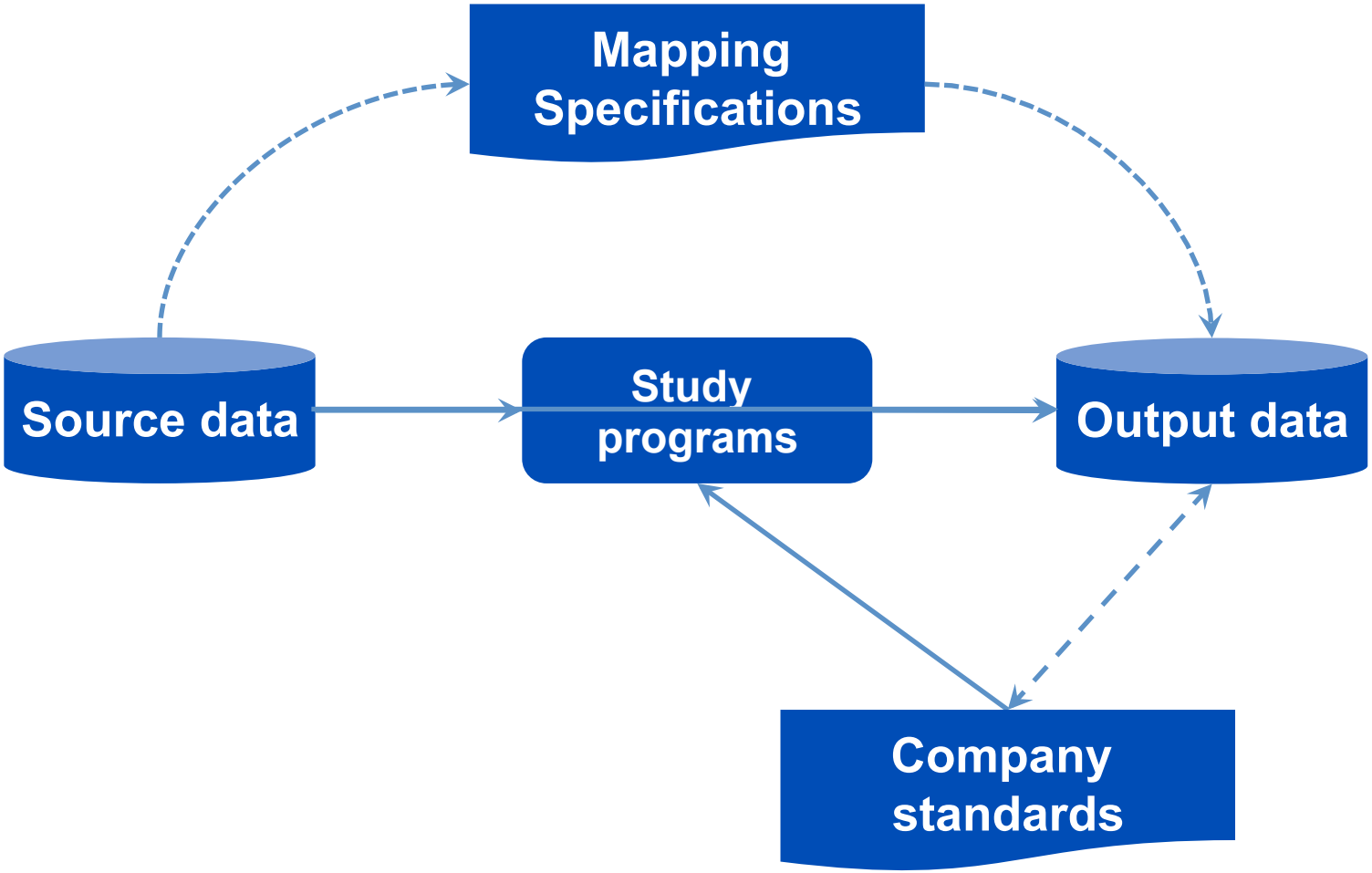
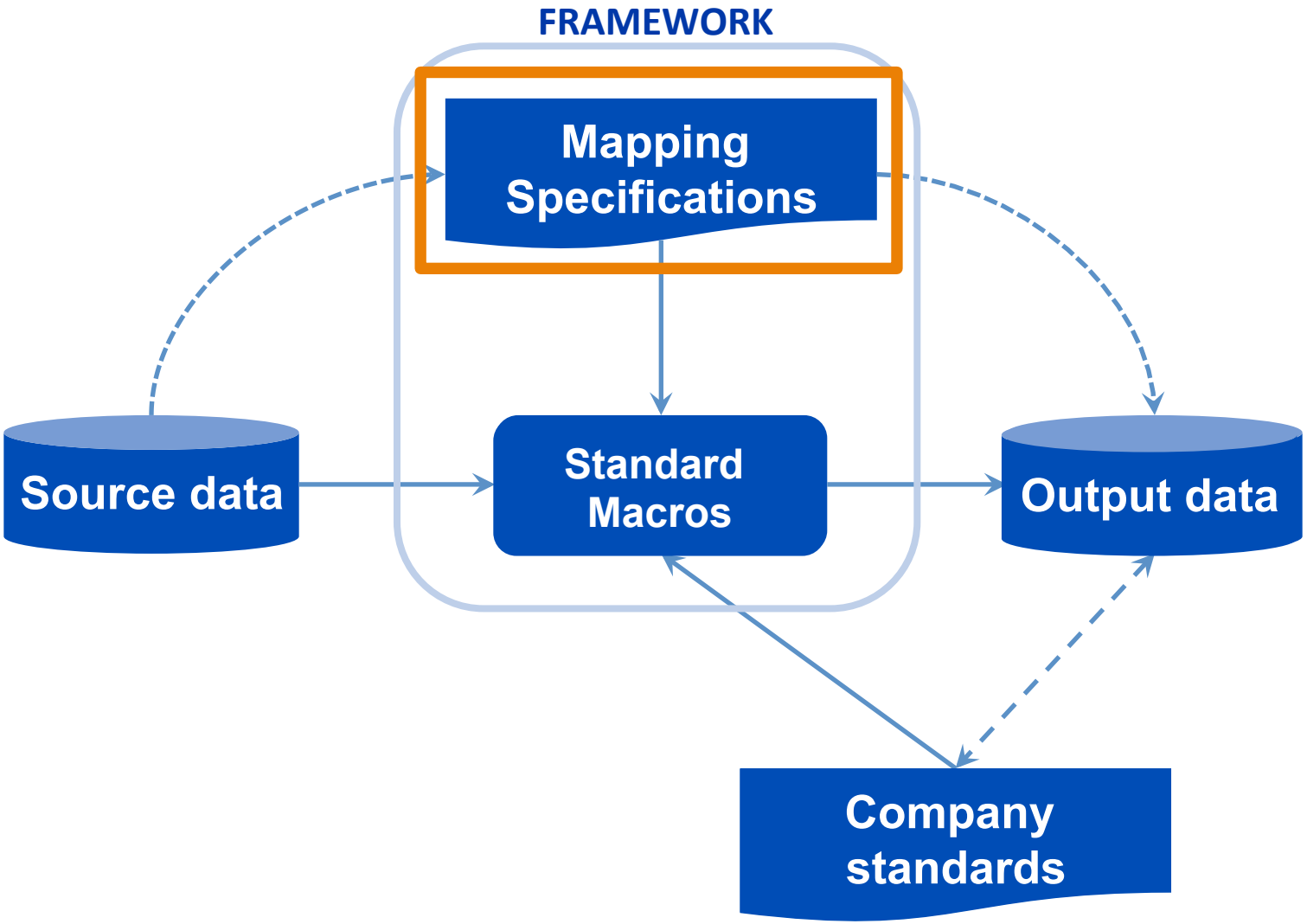
Do It Yourself:
Create your own SDTM mapping framework

Bas van Bakel, OCS Consulting, Netherlands

Scope



Scope

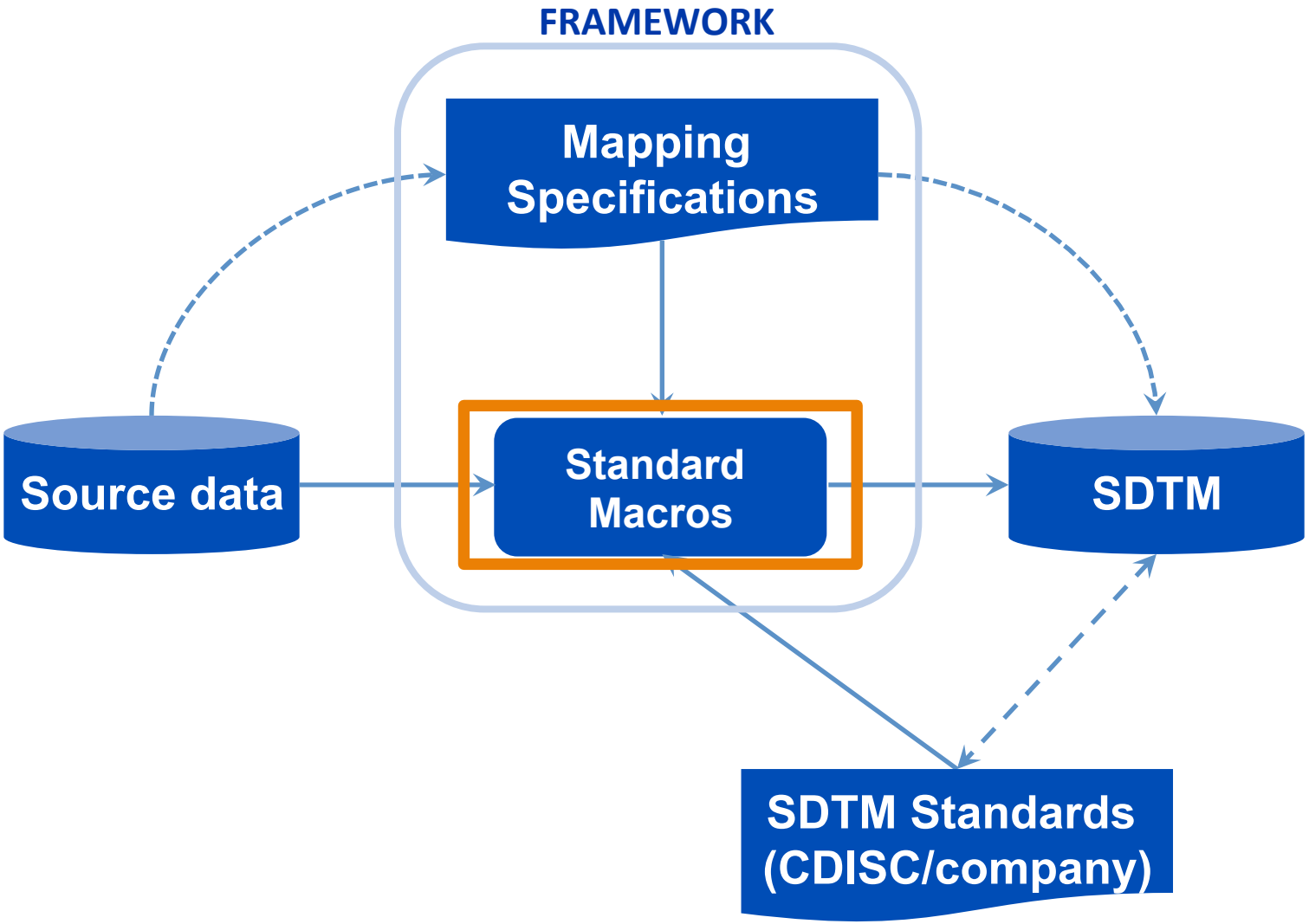


Mapping Specification document

- Microsoft Excel spreadsheet that contains **all** source-to-target specifications and, directly next to them, the translation of these specifications into code or pseudo-code.

	A	B	C	D	E	F
1	DATASET	VARIABLE	SDTM_DS	SDTM_VAR	SPECIFICATION	UNCTION
108	RAWDATA.AE	PID	AE	USUBJID	Concatenate with a '-' the calculated value of studyid and the (numeric) source PID as character value in a six digit format	UNCTION [usubjid = STRIP(studyid) || "-" || STRIP(PUT(pid,Z6.));]
109	RAWDATA.AE	AE_DESC	AE	AETERM	Copy from source variable AE_DESC, but store in uppercase and left justify	UNCTION [aeterm = STRIP(UPCASE(ae_desc));]
111	RAWDATA.AE	AE_DESC	AE		Keep only records with non-missing AE_DESC.	HERE [ae_desc NE ' ']
112	RAWDATA.AE	AE_SRDAT	AE	AESTDTC	Generate ISO8601 format from rawdate variable in character format	UNCTION [%iso_from_rawdata_date(outvar=aestdtc, invar=ae_srdat);]
113	RAWDATA.AE	AE_ERDAT	AE	AEENDTC	Generate ISO8601 format from rawdate variable in character format	UNCTION [%iso_from_rawdata_date(outvar=aeendtc, invar=ae_erdat);]
114	RAWDATA.AE	AE_INTEN	AE	AESEV	Recode source value to target value using the AESEV recoding list	ECODE [AESEV]
115	RAWDATA.AE	AE_SER	AE	AESER	Recode source value to target value using the AESER recoding list	ECODE [AESER]
123	RAWDATA.AE	MD_CODE	AE	AELLTCD	Store the (character) source MD_CODE as numeric value	UNCTION [aelltcd = INPUT(STRIP(md_code),BEST.);]
152	RAWDATA.AE	ENTRYDT			Not mapped	OT MAPPED
153	RAWDATA.AE		AE	DOMAIN	Set to 'AE' for all records.	UNCTION [domain = 'AE';]
154	RAWDATA.AE		AE		**POSTSTEP1: Merge the processed mapped RAWDATA.AE data with the MEDDR dataset:** Add information from MEDDRA.MEDDRA onto the processed RAWDATA.AE dat by merging on AELLTCD and by keeping only the records from RAWDATA.AE.	OSTSTEP1 [ROC sort DATA=work.mapped_rawdata_ae; BY aelltcd; RUN; ROC sort DATA=work.mapped_meddra_meddra; BY aelltcd; RUN; ATA work.mapped_rawdata_ae; MERGE work.mapped_rawdata_ae(IN=a) work.mapped_meddra_meddra; Y aelltcd; F a; UN;]
155	MEDDRA.MEDDRA	LLT_CODE	AE	AELLTCD	Store the (character) source LLT_CODE as numeric value.	UNCTION [aelltcd = INPUT(STRIP(llt_code),BEST.);]
156	MEDDRA.MEDDRA	PT_CODE	AE	AEPTCD	Store the (character) source PT_CODE as numeric value.	UNCTION [aeptcd = INPUT(STRIP(pt_code),BEST.);]
157	MEDDRA.MEDDRA	SOC_CODE	AE	AESOCCD	Store the (character) source SOC_CODE as numeric value.	UNCTION [aesoccd = INPUT(STRIP(soc_code),BEST.);]
158	MEDDRA.MEDDRA	SOC_CODE	AE	AEBDSYCD	Store the (character) source SOC_CODE as numeric value.	UNCTION [aebdsycd = INPUT(STRIP(soc_code),BEST.);]
159	MEDDRA.MEDDRA	HLGTCODE	AE	AEHLGTCD	Store the (character) source HLGTCODE as numeric value.	UNCTION [aehlgtcd = INPUT(STRIP(hlgtcode),BEST.);]
160	MEDDRA.MEDDRA	HLT_CODE	AE	AEHLTCD	Store the (character) source HLT_CODE as numeric value.	UNCTION [aehltcd = INPUT(STRIP(hlt_code),BEST.);]
161	MEDDRA.MEDDRA	PRIMARY	AE		Keep only records for which PRIMARY is set to 'Y'. This will ensure only the primary paths are used for coding	HERE [primary = 'Y']

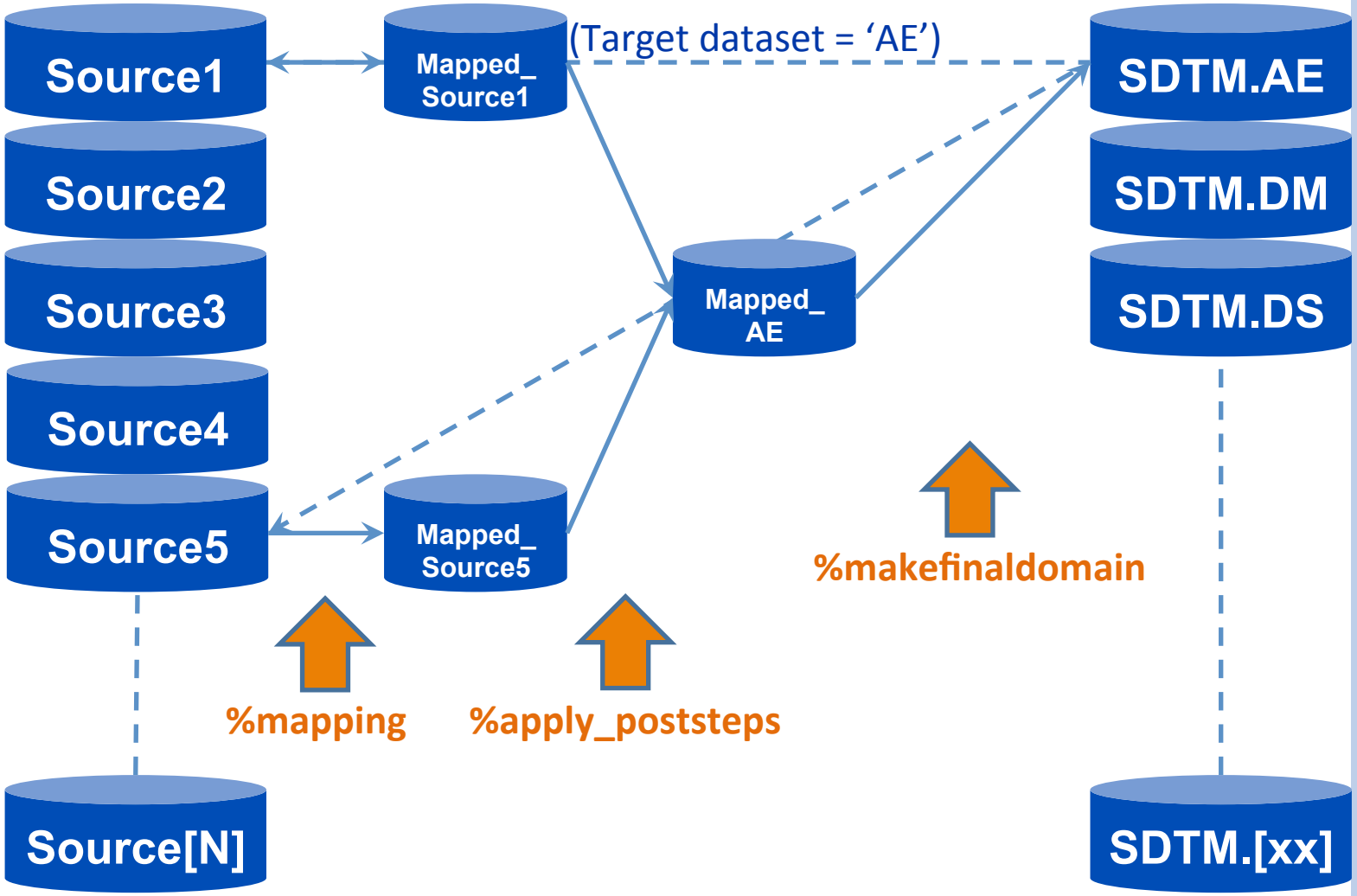
Scope



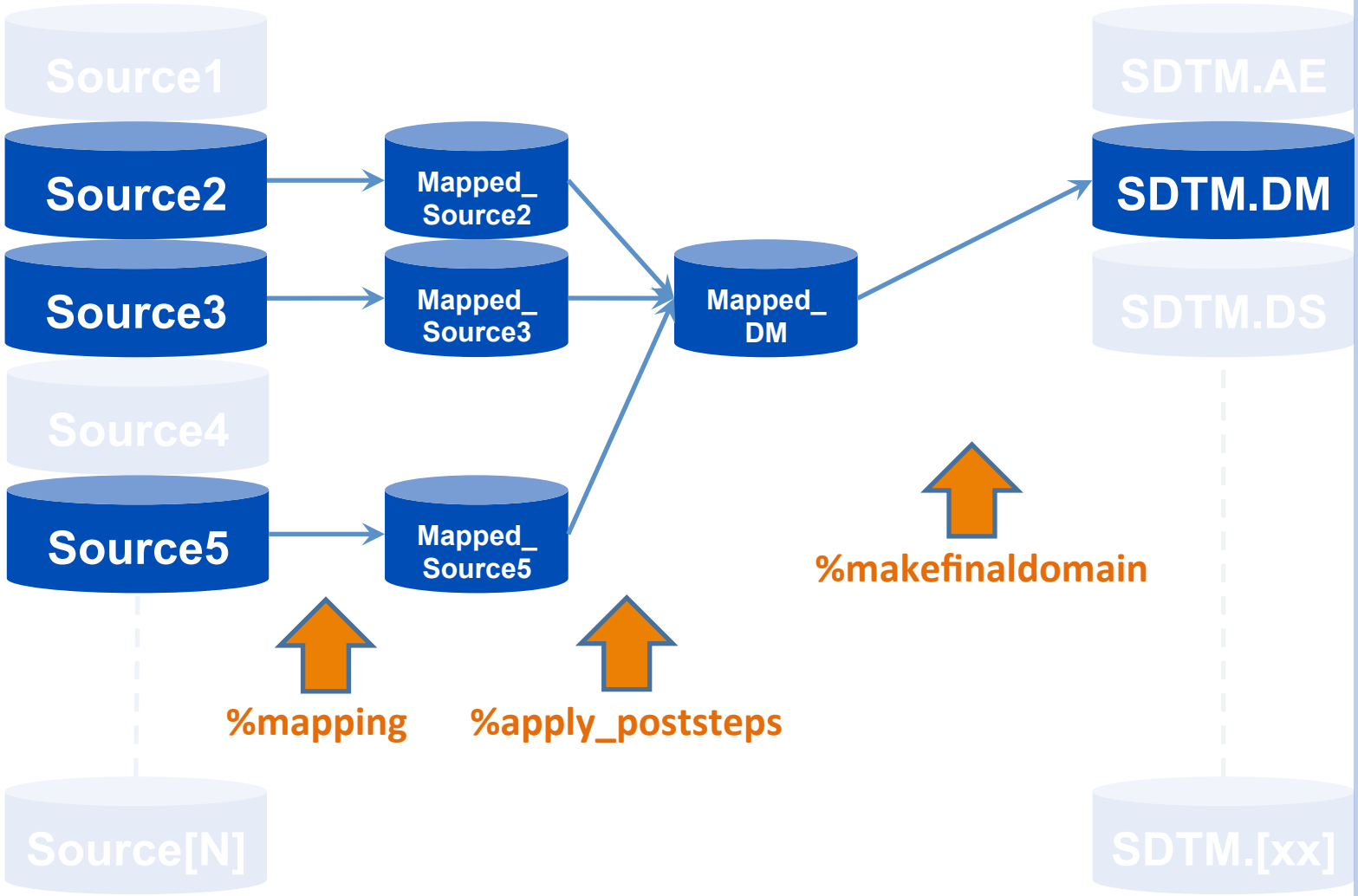
Framework Macros

- Independent of the SDTM domain that is to be generated there will always be three macros that are executed in sequence.
  - **%mapping:** Each of the source datasets are converted into intermediate ‘mapped’ dataset(s) by using some of the (pseudo-) code as specified in the mapping specification document.
  - **%apply_poststeps:** Combines/processes ‘mapped’ dataset(s) into one single ‘near final’ intermediate SDTM dataset by using the remainder of the (pseudo-) code.
  - **%makefinaldomain:** Generates SDTM dataset and Supplemental Qualifier dataset from the ‘near final’ dataset by aligning the datasets and variables with the attributes as specified in the ‘target metadata’.

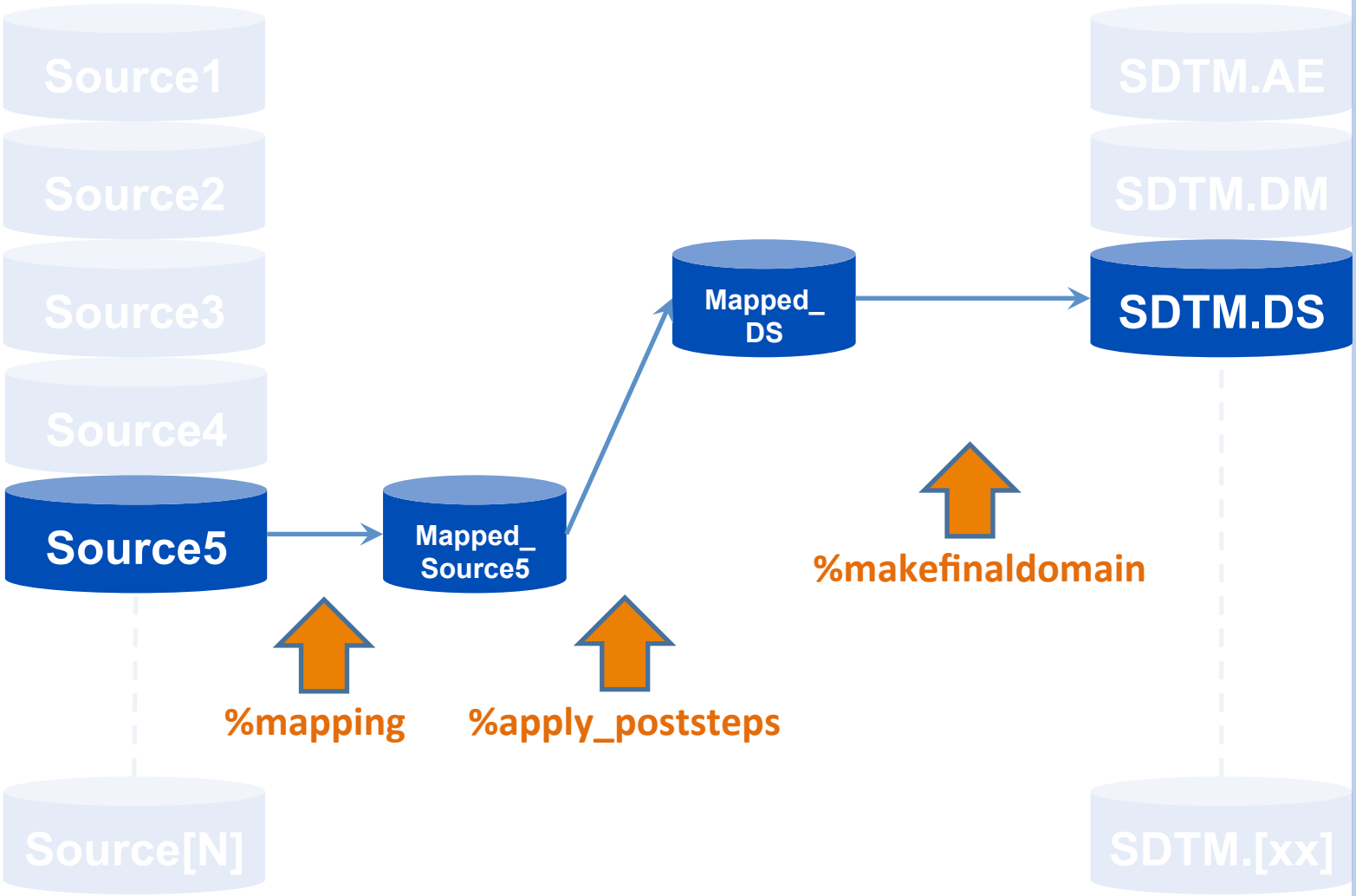
Framework Macros



Framework Macros

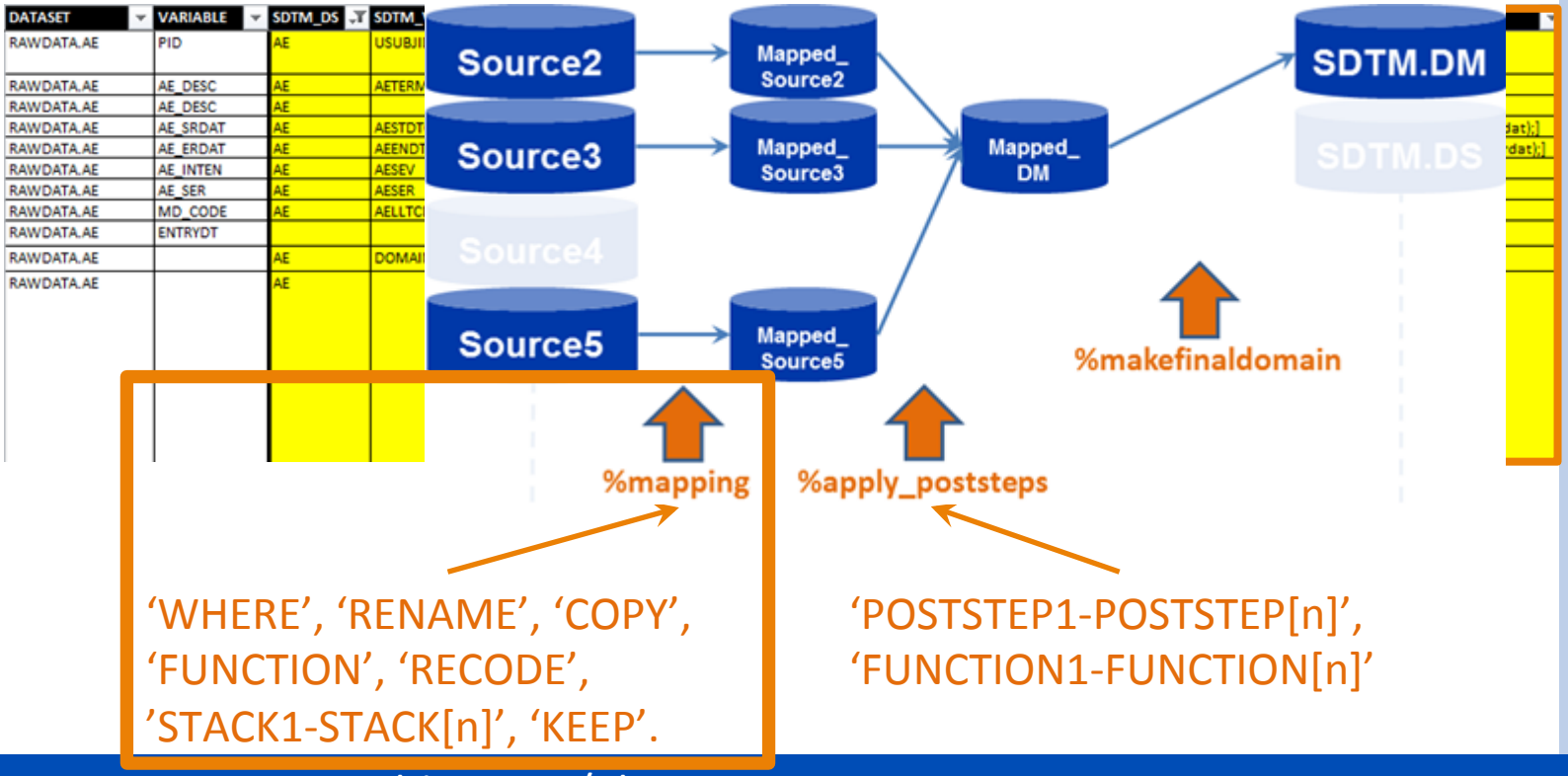


Framework Macros



Framework Macros

- Depending on the function that is specified in the FUNCTION column of the mapping specification the SAS code is executed either when the source data is converted to the ‘mapped’ datasets (%mapping) or when post processing and combining of the ‘mapped’ datasets take place (%apply_poststeps)



%Mapping Macro

- Each source dataset for which any row is available that is marked as being needed to generate the SDTM domain will be read by the %mapping macro. Only the source variables that are marked as being needed in this process are read from the source datasets.

DATASET	VARIABLE	SDTM	SDTM_VAR	SPECIFICATION	FUNCTION
CDASH.CDASH_AE	VISITNUM			Not mapped	NOT MAPPED
CDASH.CDASH_AE	SUBJID	DM		Rename for processing (SDTM output variable has the same name)	RENAME [_subjid]
CDASH.CDASH_AE	SUBJID	DM	SUBJID	Store the (numeric) source SUBJID as character value in a six digit format	FUNCTION [subjid = STRIP(PUT(_subjid,Z6.));]
CDASH.CDASH_AE	SUBJID	AE	STUDYID	Set equal to the study number defined in the input program (&studynum).	FUNCTION [studyid = STRIP("&studynum");]
CDASH.CDASH_AE	SUBJID	AE	USUBJID	Concatenate with a '-' the calculated value of studyid and the (numeric) source SUBJID as character value in a six digit format	FUNCTION [usubjid = STRIP(studyid) || "-" || STRIP(PUT(subjid,Z6.))
CDASH.CDASH_AE	AEYN	AE		Keep only records that indicate an AE occurred by keeping only records with AEYN = 'Y'	WHERE [aeyn = 'Y']
MEDDRA.MEDDRA	LLT_CODE	AE	AELLTCD	Store the (character) source LLT_CODE as numeric value.	FUNCTION [aelltcd = INPUT(STRIP(llt_code),BEST.);]
MEDDRA.MEDDRA	PT_CODE	AE	AEPTCD	Store the (character) source PT_CODE as numeric value.	FUNCTION [aeptcd = INPUT(STRIP(pt_code),BEST.);]
MEDDRA.MEDDRA	SOC_CODE	AE	AESOCCD	Store the (character) source SOC_CODE as numeric value.	FUNCTION [aesoccd = INPUT(STRIP(soc_code),BEST.);]
MEDDRA.MEDDRA	SOC_CODE	AE	AEBDSYCD	Store the (character) source SOC_CODE as numeric value.	FUNCTION [aebdsycd = INPUT(STRIP(soc_code),BEST.);]
MEDDRA.MEDDRA	HLGTCODE	AE	AEHLGTCD	Store the (character) source HLGTCODE as numeric value.	FUNCTION [aehlgtcd = INPUT(STRIP(hlgtcode),BEST.);]
MEDDRA.MEDDRA	HLT_CODE	AE	AEHLTCD	Store the (character) source HLT_CODE as numeric value.	FUNCTION [aehltcd = INPUT(STRIP(hlt_code),BEST.);]
MEDDRA.MEDDRA	PRIMARY	AE		Keep only records for which PRIMARY is set to 'Y'. This will ensure only the primary paths are used for coding	WHERE [primary = 'Y']
MEDDRA.MEDDRA	SOC1_LOC			Not mapped	NOT MAPPED

This process ensures that you cannot use variables that you have not marked in the specifications as being needed—NOT MAPPED" is used to clearly indicate a variable is not used in the mapping process.

When reading the MEDDRA dataset only the variables AELLTCD, AEPTCD, AESOCCD, AEBDSYCD, AEHLGTCD, AEHLTCD and PRIMARY will be kept.

%Mapping Macro

- When the function "**WHERE [statement]**" is specified the exact where clause as specified within the brackets will be applied on the source dataset.

DATASET	VARIABLE	SDTM	SDTM_VAR	SPECIFICATION	FUNCTION
CDASH.CDASH_AE	VISITNUM			Not mapped	NOT MAPPED
CDASH.CDASH_AE	SUBJID	DM		Rename for processing (SDTM output variable has the same name)	RENAME [_subjid]
CDASH.CDASH_AE	SUBJID	DM	SUBJID	Store the (numeric) source SUBJID as character value in a six digit format	FUNCTION [subjid = STRIP(PUT(_subjid,Z6.));]
CDASH.CDASH_AE	SUBJID	AE	STUDYID	Set equal to the study number defined in the input program (&studynum).	FUNCTION [studyid = STRIP("&studynum");]
CDASH.CDASH_AE	SUBJID	AE	USUBJID	Concatenate with a '-' the calculated value of studyid and the (numeric) source SUBJID as character value in a six digit format	FUNCTION [usubjid = STRIP(studyid) || "-" || STRIP(PUT(subjid,Z6.)
CDASH.CDASH_AE	AEYN	AE		Keep only records that indicate an AE occurred by keeping only records with AEYN = 'Y'	WHERE [aeyn = 'Y']

Resolves to: WHERE=((aeyn = 'Y'))

- If there are multiple WHERE-clauses specified they will both be applied on the source dataset so only records fulfilling both clauses will be kept when reading the source dataset.

%Mapping Macro

- When the function "**RENAME [newvar]**" is specified the source variable will be renamed to the value within the brackets and the rename statement will be applied on the source dataset.

DATASET	VARIABLE	SDTM	SDTM_VAR	SPECIFICATION	FUNCTION
CDASH.CDASH_AE	VISITNUM			Not mapped	NOT MAPPED
CDASH.CDASH_AE	SUBJID	DM		Rename for processing (SDTM output variable has the same name)	RENAME [_subjid]
CDASH.CDASH_AE	SUBJID	DM	SUBJID	Store the (numeric) source SUBJID as character value in a six digit format	FUNCTION [subjid = STRIP(PUT(_subjid,Z6.));]
CDASH.CDASH_AE	SUBJID	AE	STUDYID	Set equal to the study number defined in the input program (&studynum).	FUNCTION [studyid = STRIP("&studynum");]
CDASH.CDASH_AE	SUBJID	AE	USUBJID	Concatenate with a '-' the calculated value of studyid and the (numeric) source SUBJID as character value in a six digit format	FUNCTION [usubjid = STRIP(studyid) || "-" || STRIP(PUT(subjid,Z6.)
CDASH.CDASH_AE	AEYN	AE		Keep only records that indicate an AE occurred by keeping only records with AEYN = 'Y'	WHERE [aeyn = 'Y']

Resolves to: RENAME=(subjid=_subjid)

%Mapping Macro

- When the function "**COPY**" is specified the source variable will be copied to the SDTM target variable by generating a SAS statement in the format [target variable] = [source variable];.
- If there are multiple COPY-functions specified for a source dataset they will be concatenated

DATASET	VARIABL	SDTI	SDTM_VA	SPECIFICATION	FUNCTION
CDASH.CDASH_DM	SUBJID	DM		Rename for processing (SDTM output variable has the same name)	RENAME [_subjid]
CDASH.CDASH_DM	SUBJID	DM	STUDYID	Set equal to the study number defined in the input program (&studynum).	FUNCTION [studyid = STRIP("&studynum");]
CDASH.CDASH_DM	SUBJID	DM	SUBJID	Store the (numeric) source SUBJID as character value in a six digit format	FUNCTION [subjid = STRIP(PUT(_subjid,Z6.));]
CDASH.CDASH_DM	SUBJID	DM	USUBJID	Concatenate the calculated value of STUDYID and SUBJID with a '-'.	FUNCTION [usubjid = STRIP(studyid) || "-" || STRIP(subjid);]
CDASH.CDASH_DM	SITEID	DM	SITEID	Copy directly from source variable	COPY
CDASH.CDASH_DM	BRTHDAT	DM	BRTHDTC	Generate ISO8601 format from CDASH --DAT variable in character format	FUNCTION [%iso_from_dat(outvar=brthdtc, invar=brthdat);]
CDASH.CDASH_DM	RACE	DM	RACE	Recode source value to target value using the RACE recoding list	RECODE [RACE]
CDASH.CDASH_DM	RACE	DM	RACEOR	Copy of RACE.	COPY
CDASH.CDASH_DM	RACEOTH	DM	RACEOTH	Copy of RACEOTH.	COPY
CDASH.CDASH_DM	ETHNIC	DM	ETHNIC	Recode source value to target value using the ETHNIC recoding list	RECODE [ETHNIC]
CDASH.CDASH_DM	SEX	DM	SEX	Recode source value to target value using the SEX recoding list	RECODE [SEX]
CDASH.CDASH_DM		DM	DOMAIN	Set to 'DM' for all records.	FUNCTION [domain = 'DM';]

Resolves to: SITEID = SITEID; RACEOR = RACE; RACEOTH=RACEOTH;

%Mapping Macro

- When the function "**FUNCTION [statement]**" is specified the exact code as specified in brackets will be executed.
- If there are multiple FUNCTION functions specified for a source dataset they will be concatenated in the order of which they are available in the mapping specification document.

DATASET	VARIABL	SDTI	SDTM_VA	SPECIFICATION	FUNCTION
CDASH.CDASH_DM	SUBJID	DM		Rename for processing (SDTM output variable has the same name)	RENAME [_subjid]
CDASH.CDASH_DM	SUBJID	DM	STUDYID	Set equal to the study number defined in the input program (&studynum).	FUNCTION [studyid = STRIP("&studynum");]
CDASH.CDASH_DM	SUBJID	DM	SUBJID	Store the (numeric) source SUBJID as character value in a six digit format	FUNCTION [subjid = STRIP(PUT(_subjid,Z6.));]
CDASH.CDASH_DM	SUBJID	DM	USUBJID	Concatenate the calculated value of STUDYID and SUBJID with a '-'.	FUNCTION [usubjid = STRIP(studyid) || "-" || STRIP(subjid);]
CDASH.CDASH_DM	SITEID	DM	SITEID	Copy directly from source variable	COPY
CDASH.CDASH_DM	BRTHDAT	DM	BRTHDTC	Generate ISO8601 format from CDASH --DAT variable in character format	FUNCTION [%iso_from_dat(outvar=brthdtc, invar=brthdat);]
CDASH.CDASH_DM	RACE	DM	RACE	Recode source value to target value using the RACE recoding list	RECODE [RACE]
CDASH.CDASH_DM	RACE	DM	RACEOR	Copy of RACE.	COPY
CDASH.CDASH_DM	RACEOTH	DM	RACEOTH	Copy of RACEOTH.	COPY
CDASH.CDASH_DM	ETHNIC	DM	ETHNIC	Recode source value to target value using the ETHNIC recoding list	RECODE [ETHNIC]
CDASH.CDASH_DM	SEX	DM	SEX	Recode source value to target value using the SEX recoding list	RECODE [SEX]
CDASH.CDASH_DM		DM	DOMAIN	Set to 'DM' for all records.	FUNCTION [domain = 'DM';]

Resolves to:

```
studyid = STRIP("&studynum");  subjid = STRIP(PUT(_subjid,Z6.);
usubjid = STRIP(studyid) || "-" || STRIP(subjid);
%iso_from_dat(outvar=brthdtc, invar=brthdat); domain = 'DM';
```

%Mapping Macro

- When the function "**RECODE**" is specified the value specified in brackets will be used to check a specific part of the mapping specification document for a recoding list.

DATASET	VARIABL	SDTI	SDTM_VA	SPECIFICATION	FUNCTION
CDASH.CDASH_DM	SUBJID	DM		Rename for processing (SDTM output variable has the same name)	RENAME [_subjid]
CDASH.CDASH_DM	SUBJID	DM	STUDYID	Set equal to the study number defined in the input program (&studynum).	FUNCTION [studyid = STRIP("&studynum");]
CDASH.CDASH_DM	SUBJID	DM	SUBJID	Store the (numeric) source SUBJID as character value in a six digit format	FUNCTION [subjid = STRIP(PUT(_subjid,Z6.));]
CDASH.CDASH_DM	SUBJID	DM	USUBJID	Concatenate the calculated value of STUDYID and SUBJID with a '-'.	FUNCTION [usubjid = STRIP(studyid) || "-" || STRIP(subjid);]
CDASH.CDASH_DM	SITEID	DM	SITEID	Copy directly from source variable	COPY
CDASH.CDASH_DM	BRTHDAT	DM	BRTHDTC	Generate ISO8601 format from CDASH --DAT variable in character format	FUNCTION [%iso_from_dat(outvar=brthdtc, invar=brthdat);]
CDASH.CDASH_DM	RACE	DM	RACE	Recode source value to target value using the RACE recoding list	RECODE [RACE]
CDASH.CDASH_DM	RACE	DM	RACEOR	Copy of RACE.	COPY
CDASH.CDASH_DM	RACEOTH	DM	RACEOTH	Copy of RACEOTH.	COPY
CDASH.CDASH_DM	ETHNIC	DM	ETHNIC	Recode source value to target value using the ETHNIC recoding list	RECODE [ETHNIC]
CDASH.CDASH_DM	SEX	DM	SEX	Recode source value to target value using the SEX recoding list	RECODE [SEX]
CDASH.CDASH_DM		DM	DOMAIN	Set to 'DM' for all records.	FUNCTION [domain = 'DM';]

CODELIST	TYPE	FROM	TO
ETHNIC	C2C		
ETHNIC	C2C	HISPANIC OR LATINO	HISPANIC OR LATINO
ETHNIC	C2C	NOT HISPANIC OR LATINO	NOT HISPANIC OR LATINO
RACE	C2C		
RACE	C2C	AMERICAN INDIAN OR ALASKA NATIVE	AMERICAN INDIAN OR ALASKA NATIVE
RACE	C2C	ASIAN - CENTRAL / SOUTH ASIAN HERITAGE	ASIAN
RACE	C2C	ASIAN - EAST ASIAN HERITAGE	ASIAN
RACE	C2C	ASIAN - SOUTH EAST ASIAN HERITAGE	ASIAN
RACE	C2C	BLACK OR AFRICAN AMERICAN	BLACK OR AFRICAN AMERICAN
RACE	C2C	OTHER	OTHER
RACE	C2C	WHITE - ARABIC / NORTH AFRICAN HERITAG	WHITE
RACE	C2C	WHITE - CAUCASIAN / EUROPEAN HERITAGE	WHITE
SEX	C2C		
SEX	C2C	F	F
SEX	C2C	M	M

%Mapping Macro

- Using the recode list IF-THEN-ELSE code will be generated to translate source values to SDTM values:

```
IF ETHNIC = '' THEN ETHNIC = '';
ELSE IF ETHNIC = 'HISPANIC OR LATINO' THEN ETHNIC = 'HISPANIC OR LATINO';
ELSE IF ETHNIC = 'NOT HISPANIC OR LATINO' THEN ETHNIC = 'NOT HISPANIC OR LATINO';

IF RACE = '' THEN RACE = '';
ELSE IF RACE = 'AMERICAN INDIAN OR ALASKA NATIVE' THEN RACE = 'AMERICAN INDIAN OR ALASKA NATIVE';
ELSE IF RACE = 'ASIAN – CENTRAL / SOUTH ASIAN HERITAGE' THEN RACE = 'ASIAN';
ELSE IF RACE = 'ASIAN – EAST ASIAN HERITAGE' THEN RACE = 'ASIAN';
ELSE IF RACE = 'ASIAN – SOUTH EAST ASIAN HERITAGE' THEN RACE = 'ASIAN';
ELSE IF RACE = 'BLACK OR AFRICAN AMERICAN' THEN RACE = 'BLACK OR AFRICAN AMERICAN';
ELSE IF RACE = 'OTHER' THEN RACE = 'OTHER';
ELSE IF RACE = 'WHITE – ARABIC / NORTH AFRICAN HERITAGE' THEN RACE = 'WHITE';
ELSE IF RACE = 'WHITE – CAUCASIAN / EUROPEAN HERITAGE' THEN RACE = 'WHITE';

IF SEX = '' THEN SEX = '';
ELSE IF SEX = 'F' THEN SEX = 'F';
ELSE IF SEX = 'M' THEN SEX = 'M';
```

CODELIST	TYPE	FROM	TO
ETHNIC	C2C		
ETHNIC	C2C	HISPANIC OR LATINO	HISPANIC OR LATINO
ETHNIC	C2C	NOT HISPANIC OR LATINO	NOT HISPANIC OR LATINO
RACE	C2C		
RACE	C2C	AMERICAN INDIAN OR ALASKA NATIVE	AMERICAN INDIAN OR ALASKA NATIVE
RACE	C2C	ASIAN - CENTRAL / SOUTH ASIAN HERITAGE	ASIAN
RACE	C2C	ASIAN - EAST ASIAN HERITAGE	ASIAN
RACE	C2C	ASIAN - SOUTH EAST ASIAN HERITAGE	ASIAN
RACE	C2C	BLACK OR AFRICAN AMERICAN	BLACK OR AFRICAN AMERICAN
RACE	C2C	OTHER	OTHER
RACE	C2C	WHITE - ARABIC / NORTH AFRICAN HERITAG	WHITE
RACE	C2C	WHITE - CAUCASIAN / EUROPEAN HERITAGE	WHITE
SEX	C2C		
SEX	C2C	F	F
SEX	C2C	M	M

%Mapping Macro

- The recoding tab is also used to check that all source values are expected values by checking them against the recoding list:

```
IF RACE NOT IN ('', 'AMERICAN INDIAN OR ALASKA NATIVE', 'ASIAN – CENTRAL / SOUTH ASIAN HERITAGE',
'ASIAN – EAST ASIAN HERITAGE', 'ASIAN – SOUTH EAST ASIAN HERITAGE', 'BLACK OR AFRICAN AMERICAN',
'BLACK OR AFRICAN AMERICAN', 'OTHER', 'WHITE – ARABIC / NORTH AFRICAN HERITAGE', 'WHITE –
CAUCASIAN / EUROPEAN HERITAGE') THEN PUT "WAR" "NING: Variable RACE has value " […] "not in
recoding list";

IF ETHNIC NOT IN ('', 'HISPANIC OR LATINO', 'NOT HISPANIC OR LATINO') THEN PUT "WAR" "NING: Variable
ETHNIC has value " […] "not in recoding list";

IF SEX NOT IN ('', 'F', 'M') THEN PUT "WAR" "NING: Variable SEX has value " […] "not in recoding list";
```

CODELIST	TYPE	FROM	TO
ETHNIC	C2C		
ETHNIC	C2C	HISPANIC OR LATINO	HISPANIC OR LATINO
ETHNIC	C2C	NOT HISPANIC OR LATINO	NOT HISPANIC OR LATINO
RACE	C2C		
RACE	C2C	AMERICAN INDIAN OR ALASKA NATIVE	AMERICAN INDIAN OR ALASKA NATIVE
RACE	C2C	ASIAN - CENTRAL / SOUTH ASIAN HERITAGE	ASIAN
RACE	C2C	ASIAN - EAST ASIAN HERITAGE	ASIAN
RACE	C2C	ASIAN - SOUTH EAST ASIAN HERITAGE	ASIAN
RACE	C2C	BLACK OR AFRICAN AMERICAN	BLACK OR AFRICAN AMERICAN
RACE	C2C	OTHER	OTHER
RACE	C2C	WHITE - ARABIC / NORTH AFRICAN HERITAG	WHITE
RACE	C2C	WHITE - CAUCASIAN / EUROPEAN HERITAGE	WHITE
SEX	C2C		
SEX	C2C	F	F
SEX	C2C	M	M

%Mapping Macro

- When the function "**STACK1**", "**STACK2**" or "**STACK[n]**" is specified the exact function as specified in brackets will be executed.
- If there are multiple STACK[x] functions specified for a source dataset they will be concatenated in the order of which they are available in the mapping specification document.
- In principle STACK works in the same way as FUNCTION except that after each set of statements with the same stack number (STACK[x]) an **OUTPUT statement** will be executed.
- In practice the stack functions will therefore be used to **generate multiple output records from a single source record**.

%Mapping Macro

DATASET	VARIABLE	SDTM	SDTM_VAR	SPECIFICATION	FUNCTION
DEMOG	HT	VS	VSTESTCD	**For each record in DEMOG generate one 'height' record:** VSTESTCD = 'HEIGHT'	STACK1 [vstestcd = 'HEIGHT';]
DEMOG	HT	VS	VSTEST	**For each record in DEMOG generate one 'height' record:** VSTEST = 'Height'	STACK1 [vstest = 'Height';]
DEMOG	HT	VS	VSORRES	**For each record in DEMOG generate one 'height' record:** VSORRES = HT stored as character.	STACK1 [if missing(ht) then vsorres = ''; else vsorres = STRIP(PUT(ht,best.));]
DEMOG	HT	VS	VSSTAT	**For each record in DEMOG generate one 'height' record:** VSSTAT = 'NOT DONE' when HT is missing. Otherwise VSSTAT = missing.	STACK1 [if missing(ht) then vsstat = 'NOT DONE'; else vsstat = '';]
DEMOG	VISNUM	VS	VISITNUM	Copy from VISNUM	COPY
DEMOG	WT	VS	VSTESTCD	**For each record in DEMOG generate one 'weight' record:** VSTESTCD = 'WEIGHT'	STACK2 [vstestcd = 'WEIGHT';]
DEMOG	WT	VS	VSTEST	**For each record in DEMOG generate one 'weight' record:** VSTEST = 'Weight'	STACK2 [vstest = 'Weight';]
DEMOG	WT	VS	VSORRES	**For each record in DEMOG generate one 'weight' record:** VSORRES = WT stored as character.	STACK2 [if missing(wt) then vsorres = ''; else vsorres = STRIP(PUT(wt,best.));]
DEMOG	WT	VS	VSSTAT	**For each record in DEMOG generate one 'weight' record:** VSSTAT = 'NOT DONE' when WT is missing. Otherwise VSSTAT = missing.	STACK2 [if missing(wt) then vsstat = 'NOT DONE'; else vsstat = '';]

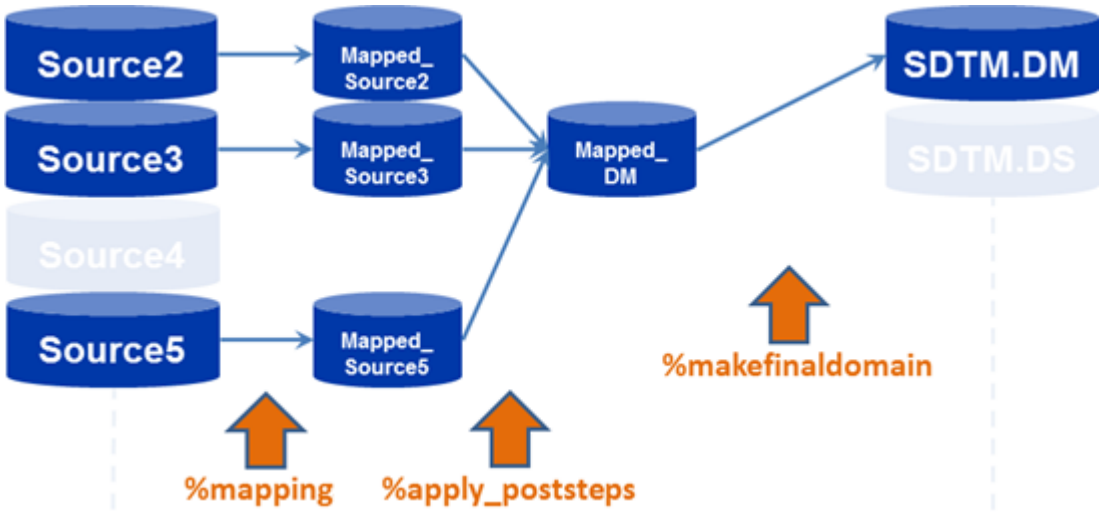
Resolves to:

```
vstestcd = 'HEIGHT';
vstest = 'Height';
if missing(ht) then vsorres = '';
else vsorres = STRIP(PUT(ht,best.));
if missing(ht) then vsstat = 'NOT DONE';
else vsstat = '';
OUTPUT;
```

```
vstestcd = 'WEIGHT';
vstest = 'Weight';
if missing(wt) then vsorres = '';
else vsorres = STRIP(PUT(wt,best.));
if missing(wt) then vsstat = 'NOT DONE';
else vsstat = '';
OUTPUT;
```

%Mapping Macro

- If you specify the function '**KEEP**' the variable from the source dataset will be retained in the 'mapped' version of the dataset and can be referenced in the code that is used by the %apply_poststeps macro.



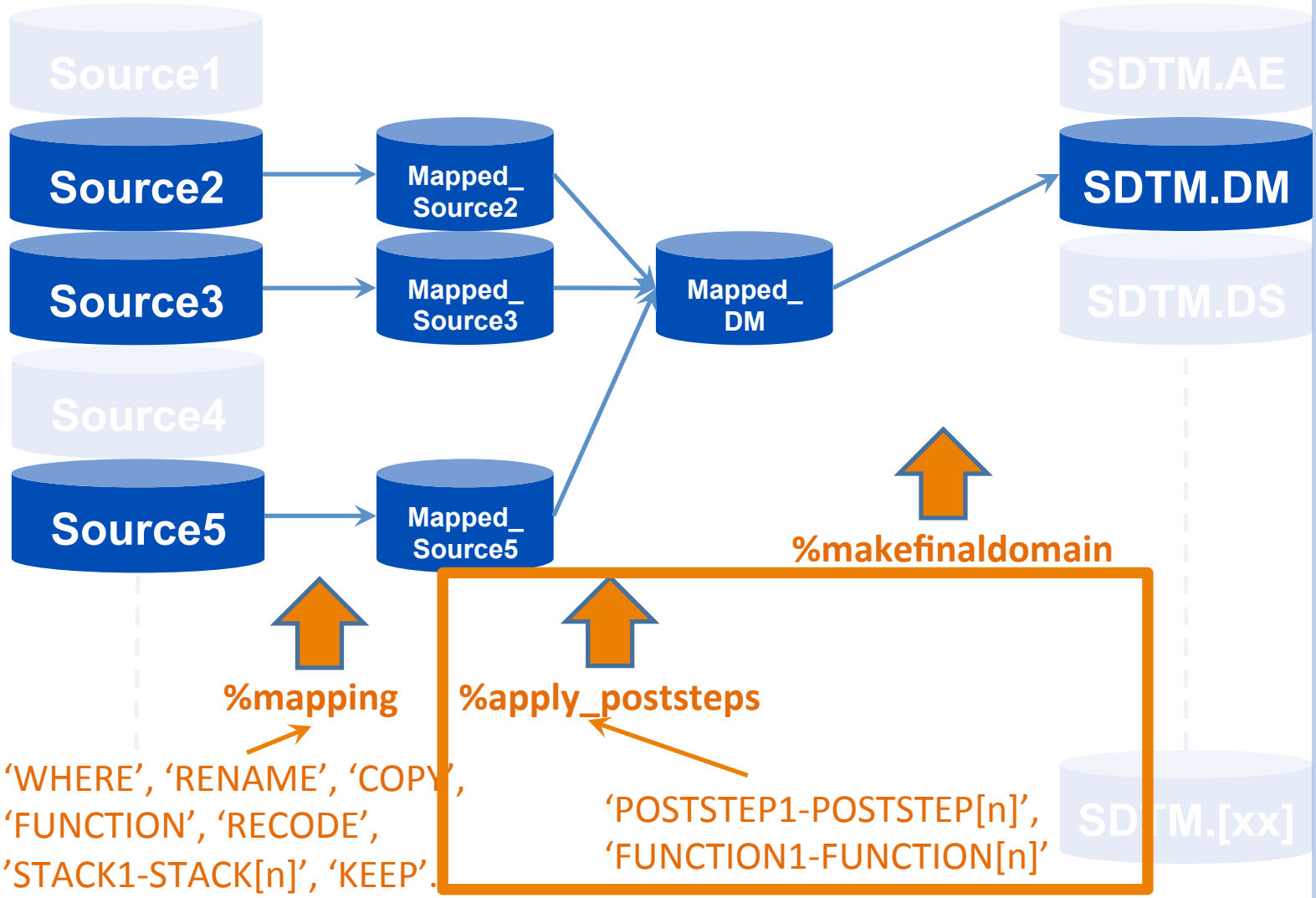
%Mapping Macro

- A simplified version of the single datastep that is generated by the %mapping macro:

```
DATA mapped_&currlib. &currds(KEEP = &manualkeepstr &tovarkeepstr);
  SET work.&currds(KEEP = &keepstr

                   %IF %BQUOTE(&wherestr) NE %THEN %DO; WHERE=((&wherestr)) %END;
                   %IF %BQUOTE(&renstr)   NE %THEN %DO; RENAME=(&renstr)    %END;
                  );
  /*Copy*/
  &copystr;

  /*Apply recodings (also generate warns when value not available*/
  &check;
  &recostr;

  /*Apply the functions*/
  &execfunc;

  /*Apply Stacking (And apply an OUTPUT after each set of stack functions)*/
  %IF &diffstack > 0 %THEN %DO;
    %DO curstack = 1 %TO &diffstack;
      &&stack&curstack OUTPUT;
    %END;
  %END;
RUN;
```

The ‘mapped’ data set is generated but only variables that are marked as [illegible] document and afterwards an OUTPUT statement is executed. If available the STACK2 functions are executed in the order as available in the mapping specification document and afterwards an OUTPUT statement is executed. (same logic applies for STACK3 up to STACK[n])

Framework Macros



%Apply_poststeps Macro

- The %apply_poststeps macro is used to obtain and execute the code specified within the POSTSTEP1-POSTSTEP[x] and FUNCTION1-FUNCTION[x] functions of the mapping specification document in the following order:

- When a function like "**POSTSTEP[x]**" is specified the exact SAS code as specified in brackets will be executed outside of a data step.
- Functions like "**FUNCTION[x]**" will be handled in exactly the same way as the FUNCTION used in the %mapping macro and will be executed within a DATA STEP. The only difference is the timing of when the code is executed:

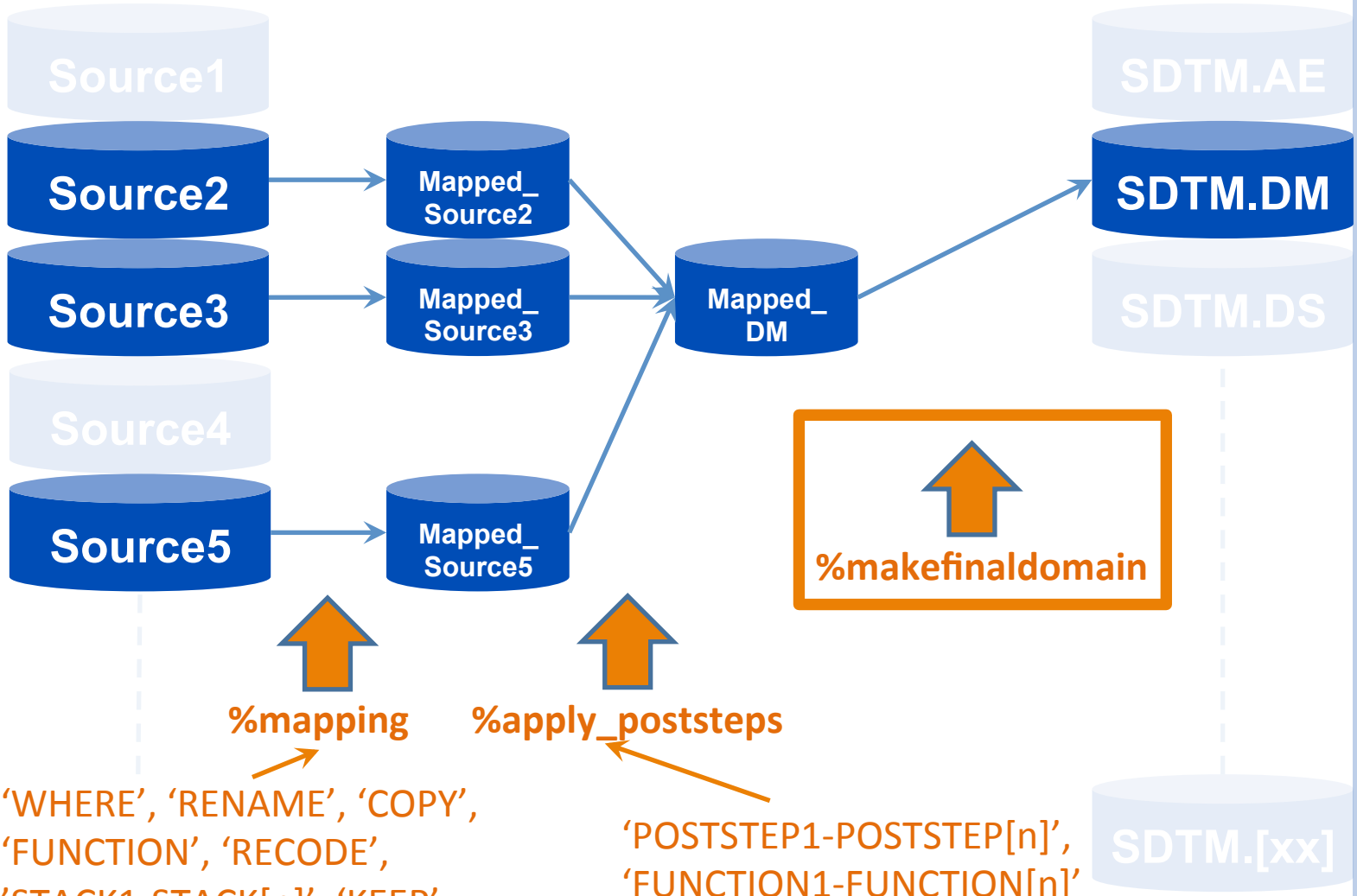
%Apply_poststeps Macro

DATASET	VAR	SDTM	SDTM_VAR	SPECIFICATION	FUNCTION
CDASH.CDASH_AE		AE		**POSTSTEP1: Merge the mapped CDASH_AE data with the SDTM.DM dataset:** Obtain the RFSTDTC from SDTM.DM by merging the mapped CDASH_AE data with the SDTM.DM domain on USUBJID and keep only records from the mapped CDASH_AE data.	POSTSTEP1 [PROC SORT DATA=work.mapped_cdash_cdash_ae; BY usubjid; RUN; DATA work.mapped_cdash_cdash_ae; MERGE work.mapped_cdash_cdash_ae(IN=a) insdtm.dm(KEEP = usubjid rfstdtc); BY usubjid; IF a; RUN;]
CDASH.CDASH_AE		AE	AESTDY	Calculate daynumber from RFSTDTC and calculated AESTDTC	FUNCTION1 [%calculatedayno(refdate=rfstdtc, date=aestdtc, dayvar=aestdy);]
CDASH.CDASH_AE		AE	AEENDY	Calculate daynumber from RFSTDTC and calculated AEENDTC	FUNCTION1 [%calculatedayno(refdate=rfstdtc, date=aeendtc, dayvar=aeendy);]
CDASH.CDASH_AE		AE		**POSTSTEP2: Merge the mapped CDASH_AE data with the MEDDRA dataset:** Add information from MEDDRA.MEDDRA onto the CDASH_AE data by merging on AELLTCD and by keeping only the records from CDASH_AE.	POSTSTEP2 [PROC SORT DATA=work.mapped_cdash_cdash_ae; BY aelltcd; RUN; PROC SORT DATA=work.mapped_meddra_meddra; BY aelltcd; RUN; DATA work.mapped_cdash_cdash_ae; MERGE work.mapped_cdash_cdash_ae(IN=a) work.mapped_meddra_meddra(KEEP = aelltcd aebdsycd aehlgtcd aehltcd aeptcd aesoccd); BY aelltcd; IF a; RUN;]
CDASH.CDASH_AE		AE	AELLT	Apply the format MD_LLT (stored within the format catalog stored in the MEDDRA folder) on character version of calculated AELLTCD to obtain the AELLT value.	FUNCTION2 [aellt = STRIP(PUT(STRIP(PUT(aelltcd,BEST.)), $MD_LLT.));]

```
PROC SORT DATA=work.mapped_cdash_cdash_ae;
 BY usubjid;
RUN;
DATA work.mapped_cdash_cdash_ae;
 MERGE work.mapped_cdash_cdash_ae(IN=a)
     insdtm.dm(KEEP = usubjid rfstdtc);
 BY usubjid;
 IF a;
RUN;
DATA work.mapped_cdash_cdash_ae;
 SET work.mapped_cdash_cdash_ae;
 %calculatedayno(refdate=rfstdtc, date=aestdtc, dayvar=aestdy);
 %calculatedayno(refdate=rfstdtc, date=aeendtc, dayvar=aeendy);
RUN;
```

```
PROC SORT DATA=work.mapped_cdash_cdash_ae; BY aelltcd; RUN;
PROC SORT DATA=work.mapped_meddra_meddra; BY aelltcd; RUN;
DATA work.mapped_cdash_cdash_ae;
 MERGE work.mapped_cdash_cdash_ae(IN=a)
     work.mapped_meddra_meddra(KEEP = aelltcd aebdsycd
                               aehlgtcd aehltcd aeptcd aesoccd);
 BY aelltcd;
 IF a;
RUN;
DATA work.mapped_cdash_cdash_ae;
 SET work.mapped_cdash_cdash_ae;
 aellt = STRIP(PUT(STRIP(PUT(aelltcd,BEST.)), $MD_LLT.));
RUN;
```

Framework Macros



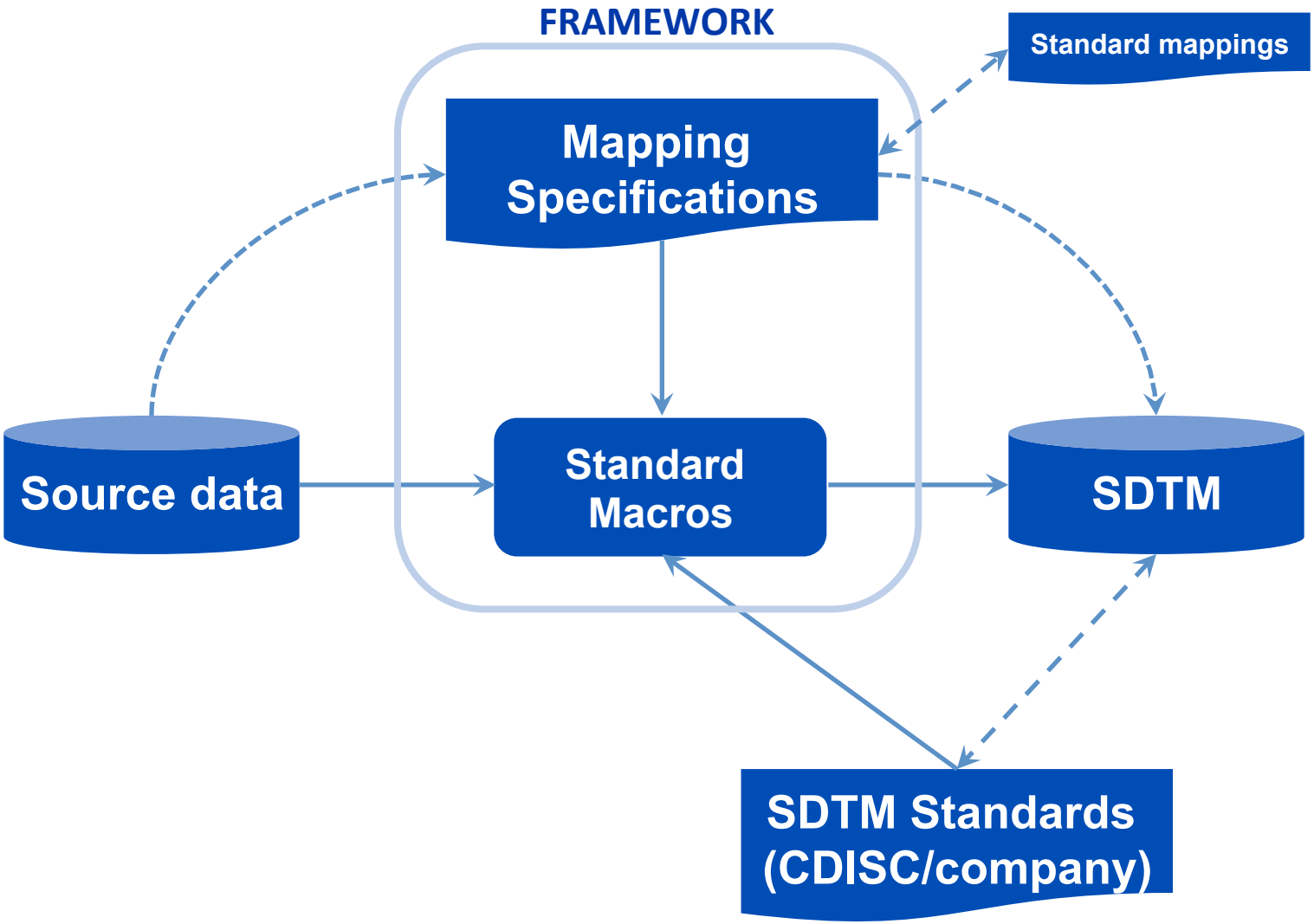
%Makefinaldomain

Using the near-final SDTM dataset and the 'target metadata' the macro %makefinaldomain will ensure that:

- Only variables that are defined in the 'target metadata' are kept in the output;
- Variables and domains get the attributes (type, length, label) as defined in the 'target metadata';
- Domains are sorted on the keys as defined in the 'target metadata' and the --SEQ variable is generated using those keys;
- The intermediate dataset is split into a SDTM 'main' domain and a Supplemental Qualifier domain (e.g. split into AE and SUPPAE).

Variables that need to go into the Supplemental Qualifier domain can be added to the intermediate 'main' dataset itself.

Standard Mappings



Standard Mappings

DATASET	VARIABLE	SD	SDTM_V	SPECIFICATION	FUNCTION	RULE_VERS	RULE_IDENTIFIER
RAWDATA.MEDIC	MARK_DT			Not mapped	NOT MAPPED	STANDARD_V1	
RAWDATA.MEDIC	MEDERDAT	CM	CMENDTC	Generate ISO8601 format from rawdate variable in character format	FUNCTION [%iso_from_rawdata_date(outvar=cmendtc, invar=mederdat);]	STANDARD_V1	
RAWDATA.MEDIC	MEDSRDAT	CM	CMSTDTC	Generate ISO8601 format from rawdate variable in character format	FUNCTION [%iso_from_rawdata_date(outvar=cmstdtc, invar=medsrdat);]	STANDARD_V1	
RAWDATA.MEDIC	TOT_DDOS	CM	CMDOSTXT	Set equal to uppercased, left justified, value of TOT_DDOS	FUNCTION [cmdostxt = UPCASE(STRIP(tot_ddos));]		
RAWDATA.MEDIC	MED_ROUT	CM	CMROUTE	Recode source value to target value using the CMROUTE recoding list	RECODE [CMROUTE]	STANDARD_V1	
RAWDATA.MEDIC	MEDINDIC	CM	CMINDC	Set equal to the uppercased value of MEDINDIC. Left justified. IF PROPH_CK is 'Y' concatenate (witha comma) the text 'IN ANTICIPATION OF STUDY VACCINE REACTION' onto the already calculated value of CMINDC.	FUNCTION [LENGTH _temp1 _temp2 $200; CALL MISSING(_temp1, _temp2); _temp1 = UPCASE(STRIP(medindic)); IF proph_ck = 'Y' THEN _temp2 = 'IN ANTICIPATION OF STUDY VACCINE REACTION'; cmindc = CATX(', ', _temp1, _temp2);]		
RAWDATA.MEDIC	PROPH_CK	CM	CMINDC	See derivation at MEDINDIC		STANDARD_V1	
RAWDATA.MEDIC	TRADNAME	CM	CMTRT	Set equal to uppercased, left justified, value of TRADNAME	FUNCTION [cmtrt = STRIP(UPCASE(tradname));]	STANDARD_V1	
RAWDATA.MEDIC	PID	CM	STUDYID	Set equal to the study number defined in the input program (&studynum).	FUNCTION [studyid = STRIP("&studynum");]	STANDARD_V1	
RAWDATA.MEDIC	PID	CM	USUBJID	Concatenate with a '-' the calculated value of studyid and the (numeric) source PID as character value in a six digit format	FUNCTION [usubjid = STRIP(studyid) || "-" || STRIP(PUT(pid,Z6.));]	STANDARD_V1	
RAWDATA.MEDIC		CM	DOMAIN	Set to 'CM' for all records.	FUNCTION [domain = 'CM';]	STANDARD_V1	
RAWDATA.MEDIC		CM	CMCAT	Set to 'CONCOMITANT MEDICATION' for all records	FUNCTION [cmcat = 'CONCOMITANT MEDICATION';]	STANDARD_V1	
RAWDATA.MEDIC		CM		**POSTSTEP3: Concatenate the processed MEDIC and processed VACC_CON datasets.**	POSTSTEP3 [DATA work.combined; SET work.mapped_rawdata_medic work.mapped_rawdata_vacc_con; RUN;]	STANDARD_V1	CONCAT_MED_CON

- All **standard rules** (i.e. the specification and function to convert source to target) are available in the **standard mapping specification**.
- Conversion programs will use the specification and function as available in the standard mapping specification if standard rule is specified.
- The specification and function specified in the study mappings specification will be compared with the standard mapping specification and warning messages are given if any difference is encountered.
- More info:

PAPER CD03 Phuse2016

Benefits

- Readability of the source-to-target mapping specification document;
- Study specific mappings never lead to adaptations of the programs, only to the mappings specification document;
- Distinction between standard and non-standard mapping rules;
- Standard metadata is used on several locations in the process which ensures all output generated is according to the defined standard. It is not possible to generate output that is not defined in the standards;
- All code that is generated and executed is SAS code and the MFILE option is used to print all executed code to a file. This is particularly useful when debugging or when the code needs to be sent to external parties;
- Warnings/errors are generated when the mapping specification document does not align with the source data or the standard mapping specification document;
- The available functions as currently available in the mapping specification document (in the column “FUNCTION”) can be extended if the need arises. For example, the STACK function was added at a later time.

Questions