Biographies

Reshma Radhakrishnan (Author)

She is working as Principal Statistical Programmer at Ephicacy Lifesciences Analytics in India. She has over 16 years of experience in leading clinical studies with end-to-end implementation of CDISC standards and submission deliverables. She has expertise in SAS programming, CDISC standards, and regulatory submissions, with a growing focus on R programming, R Shiny, and Generative AI for clinical trial programming automation and process improvement.

She holds a Bachelor of Engineering (BE) in Computer Science from Anna University, Chennai, India, earned in 2009.

Dineshkumar Subbu(Co-Author)

Expertise in R Developer with over 9 years of experience in R and Shiny application development. I specialize in building robust, scalable web applications using R and Shiny, along with providing DevOps support and automation for production environments. My interests lie in data analysis and machine learning algorithms, with a focus on developing practical and efficient solutions for complex data-driven challenges. I holds Bachelor of Computer applications (BCA.) degree from Madras University, Chennai (India) in 2003.

From Spec to Code: Automating ADaM Program Generation Using R Shiny

Track: Automation Tool
Presentation No #. AT046

Presenter: Reshma Radhakrishnan
Co-Presenter: Dinesh Kumar Subbu

Disclaimer: The opinions expressed in this talk are solely those of the authors and do not reflect the views of their employer

Agenda

- Programming Challenges
- User Interface Overview
- Workflow
- Conclusion

Programming Challenge

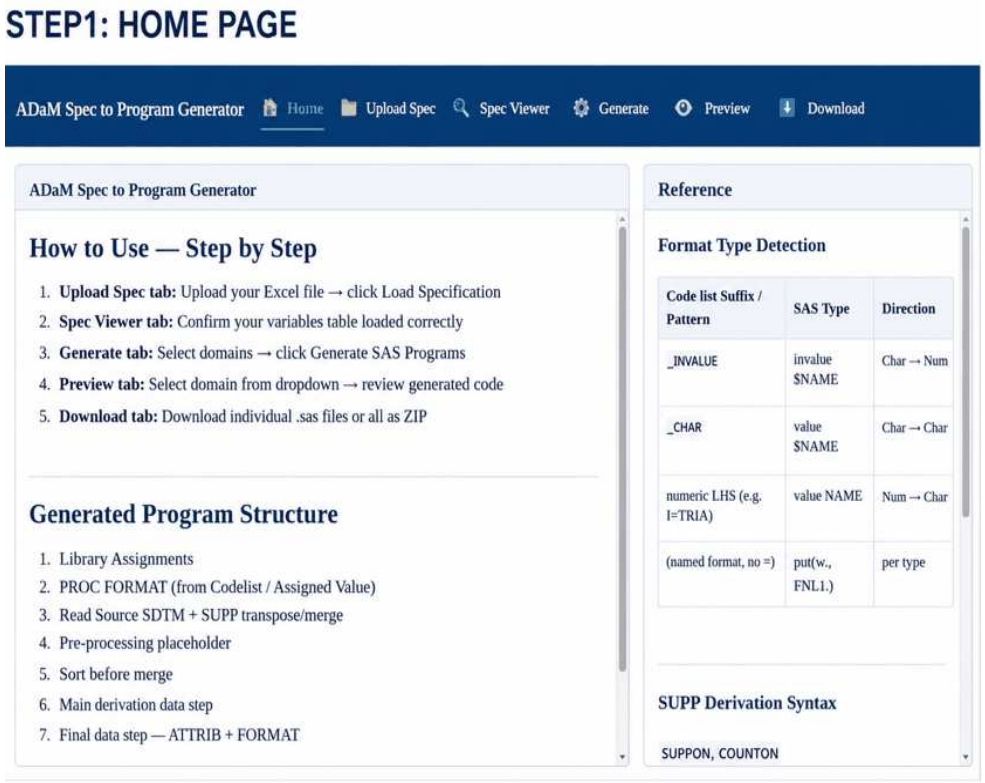


ADaM Shell Generator – User Interface Overview



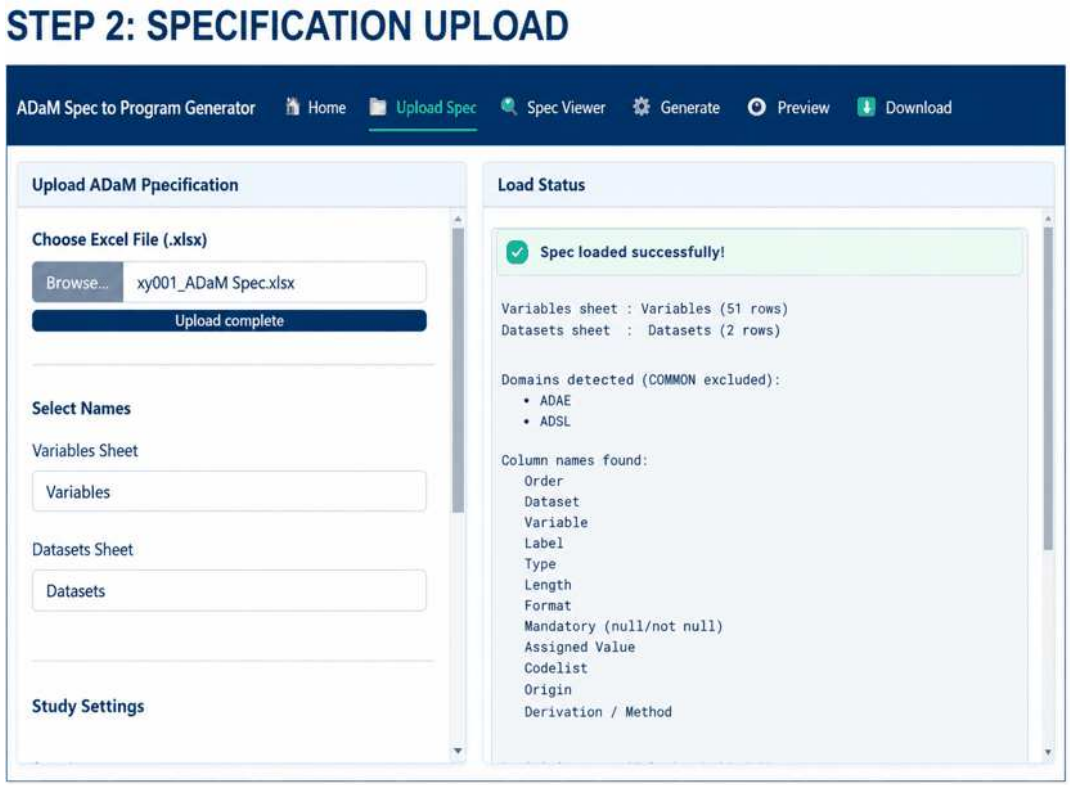
R Shiny Interface

STEP1: HOME PAGE



- Provides an overview of the tool's features and how to use it.

R Shiny Interface



- Upload the ADaM specification in this tab.
- Enter the study name and SDTM/ADaM library names for the program header.

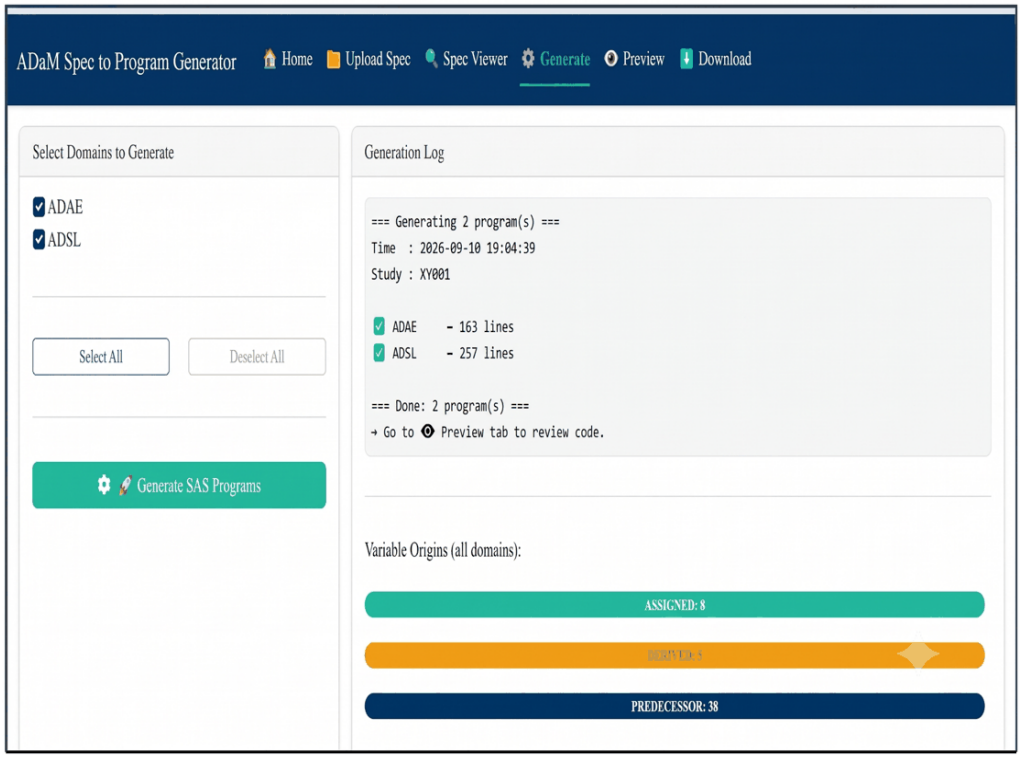
R Shiny Interface



- Allows users to preview the uploaded specification and verify the sheets.

R Shiny Interface

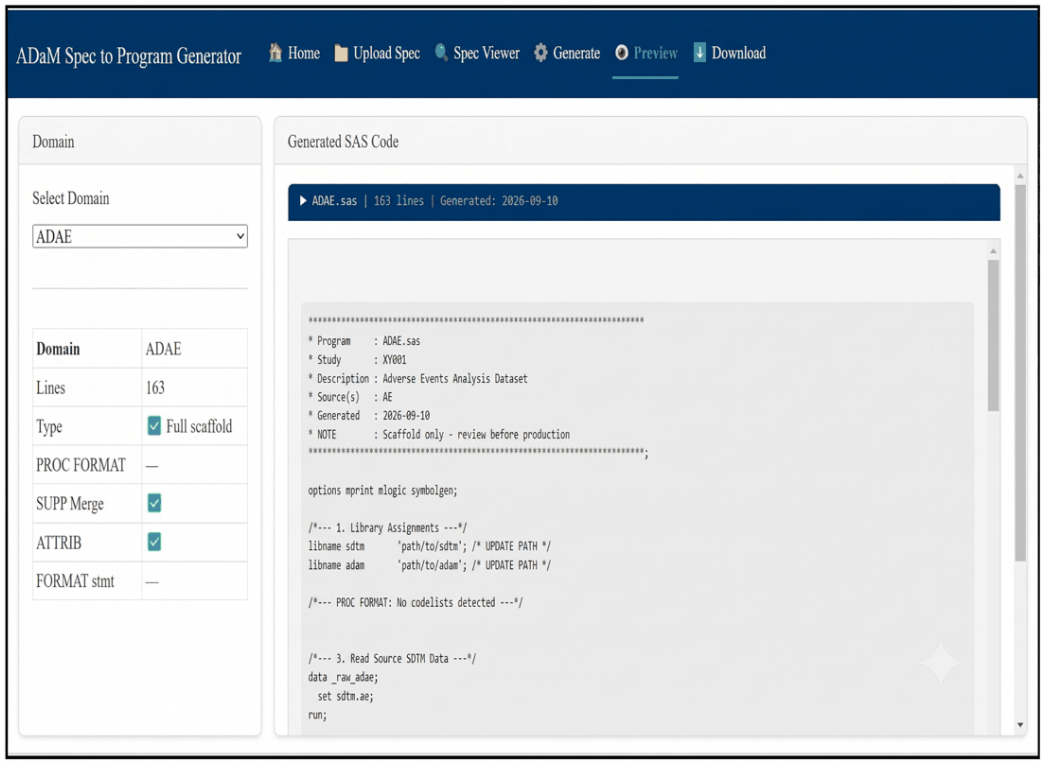
STEP 4: GENERATE SAS PROGRAM



- Lists the ADaM domains from the uploaded specification.
- Users can select a domain to generate the SAS program shell.

R Shiny Interface

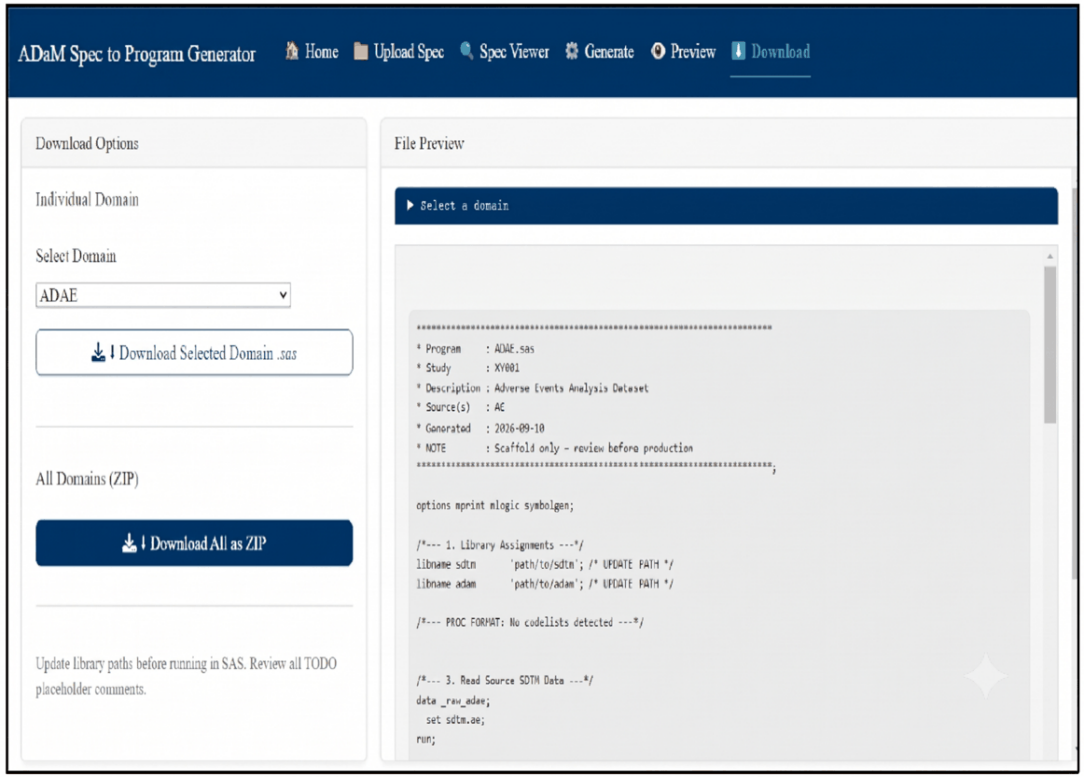
STEP 5: PREVIEW OF GENERATED SAS PROGRAMS



- Allows programmers to review the generated SAS program.
- If corrections are needed, update the ADaM specification and repeat Steps 1–4.

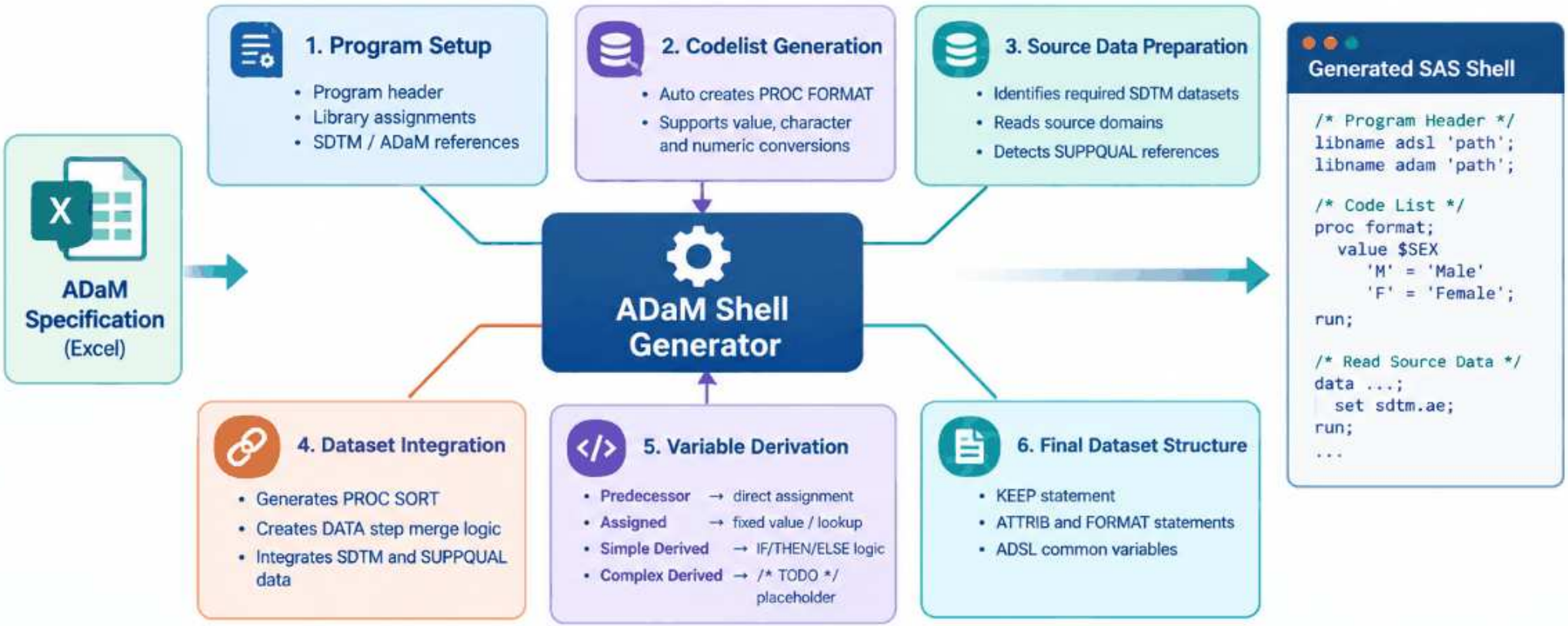
R Shiny Interface

STEP 6: GENERATE SAS PROGRAMS SHELL

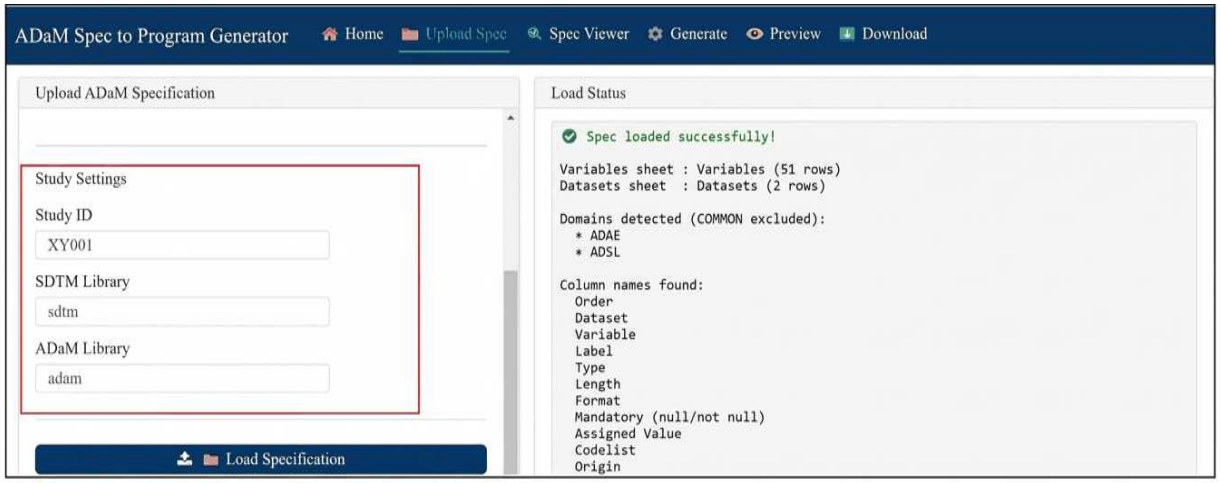


- Generate the final SAS program once the review is complete.

ADaM Shell Generator Workflow



Program Setup



SAS Snippet:

```
************************************************************************
* Program      : ADSL.sas
* Study        : XY001
* Description  : Subject Level Analysis Dataset
* Source(s)    : DM
* Generated    : 2026-09-10
* NOTE         : Scaffold only review before production
************************************************************************;

options mprint mlogic symbolgen;

/*--- 1. Library Assignments ---*/
libname sdtm       'path/to/sdtm'; /* UPDATE PATH */
libname adam       'path/to/adam'; /* UPDATE PATH */
```

Program headers are generated based on the study details entered in the Upload Specification tab.

Codelist Creation

Specification:

Order	Dataset	Variable	Label	Type	Length	Format	Mandatory (null/not null)	Assigned Value	Codelist	Origin	Derivation / Method	Comment
11	ADSL	AGEGR1	Age Group 1 Reporting	Char	50		Req	1='< 18 years' 2='>= 18 years'	AGEGR1CF F	Derived	ADSL.AGEGR1N	Type 2 → Numeric → Character (value NAME)
13	ADSL	SEX	Sex Character	Char	100	$SEXF.	Req	M='Male' F='Female' U='Unknown'	$SEXF	Assigned	DM.SEX	Type 3 → Character → Character (value NAME)
22	ADSL	TRT01PN	Planned Treatment for Period 01 (N)	Num	8		Perm	TRT A='1' TRT B='2' TRT C='3' TRT D='4'	TRT01PNF	Assigned	DM.ARM	Type 1 → Character → Numeric (invalue SNAME)

SAS Snippet:

```
/**** 2. PROC FORMAT (Auto-Generated from Spec) ****/
proc format;

  /* SEXC                    Sex Character */
  value  $SEXF
    'M'                              = 'Male'
    'F'                              = 'Female'
    'U'                              = 'Unknown'
  ;
  /* TRT01PN                 Planned Treatment for Period 01 (N) *
  invalue $TRT01PN
    'TRT A'  = 1
    'TRT B'  = 2
    'TRT C'  = 3
    'TRT D'  = 4
  ;
  /* AGEGR1R                 AgeGroup1 Reporting */
  value   AGEGR1CF
    1                                = '<18 years'
    2                                = '>=18 years'
  ;
```

Codelists are created based on the codelist columns and values specified in the ADaM specification.

Source Dataset & SUPPQUAL

- Default SDTM data sets will be used based on ADaM data set

R Code Snippet:

```
"ADSL" = "dm",  "ADAE" = "ae",  "ADCM" = "cm",
"ADLB" = "lb",  "ADVS" = "vs",  "ADEX" = "ex",
"ADMH" = "mh",  "ADEG" = "eg"
```

SAS Snippet:

```
/*=== 3. Read Source SDTM Data ===*/
data _raw_adsl;
  set sdtm.dm;
run;
```

- Derivations such as SUPPDM.PSUBJID instruct the code generator to transpose SUPPDM where QNAM = 'PSUBJID' and rename the variable to the corresponding ADaM variable name.

Specification:

Order	Dataset	Variable	Label	Type	Length	Format	Mandatory (null/not null)	Assigned Value	Codelist	Origin	Derivation / Method	Comment
8	ADSL	PSUBJID	Previous Subject Identifier for the Study	Char	20		Perm			Predecessor	SUPPDM.PSUBJID	

SAS Snippet:

```
/*--- 3b. Transpose + Merge SUPP Dataset ---*/
/*--- SUPPDM Step A: Filter (RDOMAIN=ADSL) ---*/
data _suppdm_sub;
  set sdtm.suppdm;
  where  QNAM in (
              'PSUBJID'
    );
run;

/*--- No IDVAR conversion needed (subject-level SUPP) ---*/

/*--- SUPPDM Step C: Sort before Transpose ---*/
proc sort data=_suppdm_sub;
  by STUDYID USUBJID QNAM;
run;

/*--- SUPPDM Step D: Transpose tall at' wide ---*/
proc transpose data=_suppdm_sub
               out=_suppdm_wide
                   (drop=_NAME_ _LABEL_);
  by  STUDYID USUBJID;
  id  QNAM;
  var QVAL;
```

```
/*--- SUPPDM Step F: Sort transposed data ---*/
proc sort data=_suppdm_wide;
  by STUDYID USUBJID;
run;
```

Variable Derivation

- Autogenerated SAS shell code based on Origin in ADaM Specification

Origin Type	What Is Generated	Example SAS Output
Predecessor	Direct assignment from SDTM source variable	`STUDYID = STUDYID;` `/* PREDECESSOR */`
Assigned — Fixed Value	Hardcoded value assignment	`COUNTRY = 'GBR';` `/* ASSIGNED */`
Assigned — Codelist	Applies put() or input() with PROC FORMAT	`SEX = put(SEX, SEXFMT.);` `/* ASSIGNED */`
Derived — Simple	IF/THEN/ELSE logic from spec derivation text	`if DSSCAT='PROTOCOL' then EOSSTT='COMPLETED';`
Derived — Complex	TODO placeholder flagged for programmer	`/* TODO; AVAL = <implement logic>; */`
Date Conversion	ISO 8601 to SAS date using input()	`ASTDT = input(AESTOTC, yymmndd10.)`

Supporting Components

ATTRIB Statement

Defines variable name, leng, and label from the spec.

```
attrib STUDYID length=$20
label='Study Identifier';
```

FORMAT Statement

Applies display formats from the Format column of the spec.

```
format TRTSDT DATE9. AGE 8.
SEX $SEXFMT.;
```

KEEP Statement

Retains only spec-defined variables in their specified order.

```
keep STUDYID USUBJID AGE
SEX TRTSDT TRTEDI;
```

ADSL Common Variables

Merges shared ADSL variables flagged as Domain = “Common”.

```
merge adsl(keep=STUDYID
USUBJID TRT01P ...)
```

Performance Metrics

ADaM Shell Program Generator vs. Manual Programming

Metric	Manual Process	With Tool	Improvement
Time to Generate Shell	4-8 hrs per domain	**<1 hr** per domain	**~75-85%** reduction
Overall Programming Time	Baseline	Reduced	**~40%** savings
Human Error Rate	High — manual transcription	Near-zero for shell	**~80%** reduction
CDISC Compliance	Varies by programmer	100% spec-aligned	Standardized
Codelist Format Generation	30-60 min per domain	Fully automated	**~100%** automated
SUPPQUAL Merge Logic	Manual, error-prone	Auto-detected & generated	Eliminated manual step

40%
Overall Time Saved

75–85%
Faster Shell Generation

~100%
Codelist Automation

Conclusion

Automates ADaM Spec to SAS Shell

Eliminates repetitive manual coding from the spec workbook.

Covers All Key Components

ATTRIB, FORMAT, PROC FORMAT, SDTM merge, SUPPQUAL & derivations

Origin-Driven Code Generation

Handles Predecessor, Assigned, and Derived variables from the spec

Saves Time & Reduces Errors

Programmers focus on complex logic — routine coding is automated.

Lightweight & Accessible

Built in R Shiny — no proprietary software required.

Strong Foundation for Growth

Planned: R output, SDTM variable existence check, spec change detection

Name: Reshma Radhakrishnan
Organization: Ephicacy Lifescience Analytics
Address: No. 06, 2nd Main Rd, British Biologicals, Arekere City, Bengaluru, Karnataka 560076
E-mail: reshma.radhakrishnan@ephicacy.com
Web: www.ephicacy.com
LinkedIn: www.linkedin.com/in/reshma-radhakrishnan-374961118

Name: Dineshkumar Subbu
Organization: Ephicacy Lifescience Analytics
E-mail: dinesh.subbu@ephicacy.com
Web: www.ephicacy.com
LinkedIn: www.linkedin.com/in/dineshkumar-subbu