Centers for Medicare & Medicaid Services

CY 2026 Part D

Bid Review Out-of-Pocket Cost Model

User Guide

April 2025

Table of Contents

Introduction

The Out-of-Pocket Cost (OOPC) Model is a set of programs used to calculate the estimated OOPC values using a beneficiary sample to determine the value of the benefits being offered by a Plan Benefit Package (PBP). The purpose of this User Guide is to provide Medicare Advantage Organizations (MAOs) and Prescription Drug Plan (PDP) sponsors with the technical information required to generate OOPC values in order to evaluate the changes to the OOPC model. Please note, MAOs will need to calculate their Part C & Part D OOPC values separately and combine them for their total OOPC value.

This version of the Part D OOPC model, referred to as the **CY 2026 Part D Bid Review OOPC Model,** uses a 0.1% sample of Part D beneficiary drug utilization. The OOPC Model is designed to enable plan organizations to review benefit structures using the software and data that will be used by the Center for Medicare and Medicaid Services (CMS) to evaluate annual bid submissions. The OOPC Model reports OOPC values at the plan level. The section, **Development of the Part D OOPC Data**, summarizes CMS' process to produce the OOPC values. Plan Sponsors are encouraged to review the more comprehensive Methodology document located in the most recently posted Plan version OOPC Model package at

https://www.cms.gov/medicare/coverage/prescription-drug-coverage/out-of-pocket-costs

Organizations use their own completed 2026 PBP data as input to the software (a given organization may have multiple plans for a given contract). After the user has successfully input their data for a particular contract/plan, and completed the PBP Rx section on Health Plan Management System (HPMS) PBP module page, the data is ready for use in the model. The model will use the JavaScript Object Notation (JSON) Files available for download from the HPMS PBP Interface as the input to the model, or JSON files created by the organizations that follow the HPMS PBP formatting guidelines. Users download the OOPC Model and follow the directions for where to copy the SAS programs and SAS data that serve as the other inputs. The user edits several small SAS programs and then executes them to produce OOPC estimates.

The CY 2026 Part D Bid Review OOPC Model package (**CY 2026 Bid Review OOPC Model.ZIP**) contains input datasets (SAS transport format) and a series of SAS programs. The programs import PBP, formulary, and utilization data; calculate person-plan-level costs for Part D benefits, summarize the costs to the plan level, and output the results at a plan-level to an Excel workbook.

Questions can be directed as follows:

- For Part C policy-related questions, such as Total Beneficiary Cost (TBC), please submit inquiries to the MA Benefits Mailbox at: mabenefitsmailbox.lmi.org
- For Part D policy-related questions about meaningful difference, please submit an email to PartDBenefits@cms.hhs.gov
- For technical questions about the Bid Pricing Tool, please submit an email to Actuarial-Bids@cms.hhs.gov

- For technical questions about the OOPC models, please submit an email to OOPC@cms.hhs.gov

Resource Requirements

Operation of the Model requires that the user be familiar with Personal Computer (PC) file management and operating SAS software.

Model Requirements: The Models have been tested on a variety of PCs. The user will need WINZIP (or similar software) to unzip the OOPC model package and storage space to accommodate the downloadable files that total over 5 MB zipped). A version of PC SAS with the built-in LIBNAME engine for JSON is needed to run the model. The model was developed and tested using SAS Version 9.4 TS Level 1M7 on 64-bit machines using Microsoft 365. Microsoft Excel is required for generating and using the model output. Testing has been done using the Microsoft 365 version of Excel.

Processing Time: The programs that import the various input files will run quickly. The other programs for the Part D calculations incorporate many different variables and combinations of covered/non-covered drugs, pricing structures, and formularies. Running a single or a few plans at a time will shorten the run time, especially when fewer drug formularies are involved.

Input Datasets Included in the Software Package

Drug Utilization Data Provided by CMS

The software includes four SAS transport datasets. The person-level (PERSON.XPT) file contains information on the cohort of beneficiaries drawn from the CMS Common Medicare Environment (CME) files. The BENE_SCRIPT.XPT file lists the drug names and scripts for each sample beneficiary extracted from the Prescription Drug Event (PDE) file. The RXCUI_REFERENCE.XPT file lists all of the RxCUIs used in the OOPC model, along with their associated Brand/Generic status, and average prices. The FRF_XWALK.XPT lists all the RxCUIs and their potential generic substitution RxCUIs. The datasets are used after they are converted to SAS datasets with a SAS program included in the package (CIMPORT.SAS).

Input Datasets Provided by the User

Plan List

The user will provide a text file list of the plans to be included in an OOPC run. This file (**PLANFILE.TXT**) will consist of a combined Contract/Plan/Segment identifier. Part D plans do not have segment identifier, but one is required to be added as “000” for this file. For example, PDP plan S9999 001 will appear as S9999001000.

Planfile.txt Record Layout

Required File Format = ASCII File - Tab Delimited
Do not include a header record
Filename extension should be ".TXT"

Field Name	Field Type	Field Length	Field Description	Sample Field Value(s)
Contract_Plan_Segment	CHAR	11	Unique Contract/Plan/Segment identifier	S9999001000

Parts of an example file look like:

S9999001000
S9998001000
S9997002000

Note: Only plans in the plan list will be run in the OOPC calculation, even if more plans exist in a user's PBP database.

PBP Data

Each year, plan personnel and other users are required to enter their benefit data into the PBP module located in HPMS in order to submit a bid. Plans are provided with instructions each year on how to enter data into the PBP module using the graphical user interface, and on how to directly upload data to HPMS via the PBP Application Programming Interface (API).

PBP Data Input to OOPC Tool: As part of this bid creation process, the PBP data that a user entered in HPMS is available for download from HPMS as a JSON file, which will be used as an input file for the OOPC calculation(s). Plans are cautioned that any changes subsequently made directly on HPMS will not be reflected in the OOPC model unless a new JSON file is downloaded from HPMS. The OOPC model requires JSON files that are properly formatted, and that reflect all benefits offered by a plan. A SAS program in the OOPC Model can read a plan's PBP data from the JSON file and convert it to a SAS data file.

Drug Formulary Data

For producing the Part D OOPC values, plan organizations with Part D benefits (PDPs and MA-PDs) will produce three files that describe the plan's formulary.

The first file, **FORMULARY.TXT**, needs to contain a tab-delimited list of the drugs for each formulary of the plans to be included in an OOPC calculation. This file and other .txt files described below should not contain header, or label rows, and should keep any leading zeroes. Each row in the file will contain, in this order: a formulary identifier, an Rx Norrm Concept Unique Identifier (RxCUI), and a tier-level identifier (1-7). This information can be obtained from the plan organization's formulary.

Formulary.txt Record Layout

Required File Format = ASCII File - Tab Delimited
Do not include a header record
Filename extension should be ".TXT"

Field Name	Field Type	Field Length	Field Description	Sample Field Value(s)
Formulary ID	CHAR	8	Unique Formulary Identifier	00026990
RxCUI	Number	Maximum of 8 digits	Rx Norm concept unique identifier from the active CY2024 Formulary Reference File	721775
Tier Level	Number	1	Defines the Cost Share Tier level associated with the drug	1 = Tier Level 1 2 = Tier Level 2 3 = Tier Level 3 4 = Tier Level 4 5 = Tier Level 5 6 = Tier Level 6 7 = Tier Level 7

Parts of an example file look like:

```
00026990   721775        1
00026991   721793        1
00026992   721795        2
00026993   721797        3
00026994   722113        2
```

Note: The Formulary ID needs to have an 8-digit field length. Any entries greater or less than 8 digits will not be read or used by the model.

The second file, **FFF.TXT**, contains a tab-delimited list of all plans and drugs (RxCUIs) for each plan that offers Free First Fill coverage. This information can be obtained from the plan organizations' supplemental formulary file submissions. The file will be submitted with a Contract identifier and a Plan identifier (no segment identifier required).

Note: If a plan has no partial tier coverage or no Free First Fill drugs, a blank version (i.e., no rows) of **the FFF.TXT** file needs to be created and saved.

FFF.txt Record Layout

Required File Format = ASCII File - Tab Delimited
Do not include a header record
Filename extension should be ".TXT"

Field Name	Field Type	Field Length	Field Description	Sample Field Value(s)
Contract ID	CHAR	5	Contract Number	S9999
Plan ID	CHAR	3	Plan Identifier	001
RxCUI	Number	Maximum of 8 digits	Rx Norm concept unique identifier from the active CY2024 Formulary Reference File	721775

Parts of an example file look like:

```
S9999      001      721797
S9999      001      722113
```

The third file, **PLAN_FORMULARY.TXT**, contains a tab-delimited list of all contracts, plan, and formulary identifiers that are to be run. The list of plans needs to correspond exactly with the list of plans in the **PLANFILE.TXT** file described above, although only the contract plan and plan identifiers are required.

Plan Formulary.txt Record Layout

Required File Format = ASCII File - Tab Delimited
Do not include a header record
Filename extension should be ".TXT"

Field Name	Field Type	Field Length	Field Description	Sample Field Value(s)
Contract ID	CHAR	5	Contract Number	S9999
Plan ID	CHAR	3	Plan Identifier	001
Formulary ID	CHAR	8	Unique Formulary	00026990

Parts of an example file look like:

```
S9999      001      00026994
S9998      001      00026990
S9997      002      00026991
```

Programs Included in the Software Package

The complete list of SAS Programs utilized by the model can be found in the Contents of the Zip File section below. The key programs that launch the computations are described below:

CIMPORT.SAS converts the SAS transport files supplied with this software into SAS datasets.

PARTD_FORM.SAS takes the Part D related formulary files described above and converts them into SAS format.

OOPCV1P.SAS supplies user-defined parameters needed to run the OOPC Model and calls the other SAS programs that carry out the calculations.

Instructions for Running the Model and Creating OOPC Values

Please read and follow the instructions carefully before running the software. Note that this assumes the PBP has already been completed for the plan(s) of interest.

Step 1: Create a text file (**JSONfile.TXT**) that has the name of the JSON file to be used in the OOPC run. Only **one** JSON file can be referenced in the JSONfile.TXT file. Note: The JSONfile.TXT should only have a single line denoting the PBP file to be read.

Step 2: Create a text file (**PLANFILE.TXT**) that lists the plans to be used in the OOPC run.

Step 3: Complete the PBP data entry for plans of interest using the HPMS PBP module and download the data. On the HPMS PBP page, click **Reports** from the left navigation menu. On this **Generate Report** screen, select **2026** for **Contract Year** and **JSON File** under **Report Type** drop-down box then all contracts and plans that have completed PBP data you wish to include in the OOPC model. In the **Section(s)** drop-down box, check **Both MA and Rx**, **MA**, or **Rx** check box, and then click on the green **Download** button. The file is saved (most configurations default to the "Downloads" folder) of a user's PC as a form of **JSON** file e.g., PBP_JSON_File_xxxxxxxxxxxxx.json, where the xxxx is a string of numbers that is unique to each download. Save this **JSON** file to a designated directory: e.g., **c:\oopc_d\pbp**; and rename the file to match the name inserted in the JSONfile.TXT file created in step 1. Note that the OOPC model will only read in a PBP JSON file with this name.

Step 4: Create text files for the formulary information of the plans to be run: **FORMULARY.TXT**, **PLAN_FORMULARY.TXT**, and **FFF.TXT,** and copy them to a created formulary directory: e.g., c:\oopc_d\formulary. Make a note of the location of these files.

Step 5: Set up directory locations for all files.

a. Copy the **CY 2026 Bid Review OOPC Model.ZIP** to a working directory (e.g., **c:\oopc_d**) and unzip its contents to that directory
b. In the working directory, extract the contents of **programs.zip** to create the **c:\oopc_d\programs** directory for the SAS programs modified by the user.
c. In the working directory, extract the contents of **input.zip** to create the **c:\oopc_d\input** directory for the input files and the programs that are not changed by the user.
d. In the input directory, create a jsonlib folder (e.g., **c:\oopc_d \input\jsonlib**).

e. In the working directory, set up a directory for the json temporary files (e.g., **c:\oopc_d\pbptemp**).

f. Set up a directory for the output spreadsheet file (e.g., **c:\oopc_d\output**).

g. Copy the **PLANFILE.TXT** file to the newly created programs file directory. (e.g., **c:\oopc_d\programs**).

h. Copy the **JSONfile.TXT** file to the newly created programs file directory. (e.g., **c:\oopc_d\programs**).

Step 6: Edit the program **CIMPORT.SAS** as necessary so that the location **(in bold below)** of the input data is specified for all of the .XPT files. The programs provided in the model package contain, as defaults, the directory locations listed above. The user can change these locations, as desired.

```
* PROGRAM: CIMPORT.SAS;
* DESCRIPTION: IMPORT THE INPUT FILES TO THE OOPC PROCESS;

%LET DATALOC = %str(c:\oopc_d\input);
```

Then run CIMPORT.SAS.

For all SAS runs, check the SAS Log to make sure the text string **ERROR** does not appear anywhere. (In the **Troubleshooting** section below are noted several sources of problems when setting up and running the programs).

Note: The SAS CIMPORT program only needs to be run once, prior to the first OOPC run.

Step 7: Import **FORMULARY.TXT**, **PLAN_FORMULARY.TXT,** and **FFF.TXT,** files by editing the provided **PARTD_FORM.SAS** program, as necessary, for the correct directory locations and files.

```
*PROGRAM: PARTD_FORM.SAS;
*DESCRIPTION: CREATES SAS FILES FOR FIVE TAB DELIMITED FILES;
%LET DIR = C:\OOPC_D\formulary;
%LET FORMFILE = FORMULARY.TXT;
%LET PLANFORM = PLAN_FORMULARY.TXT;
%LET FFF = FFF.TXT;
```

Then run PARTD_FORM.SAS.

Note: Once this step is finished, and if there is no change in the formulary data, the user does not need to redo this step for subsequent runs. As necessary, the user may create and use different formulary text files and rerun **PARTD_FORM.SAS**.

Step 8: Edit the program **OOPCV1P.SAS** as shown below to indicate the directories (**in bold**) where the SAS programs and input files are stored. The programs provided in the model package contain, as defaults, the directory locations listed above. Also, edit the program to indicate where the PBP data are stored. And finally, edit the program to identify the location and name of the output spreadsheet file. The output spreadsheet name can be changed as necessary.

Note: The model has been updated to allow plans to run the formulary tied to non-DS plans through a DS benefit design. To produce these results change “**DSFORM_FLAG**” to 1. The flag defaults to a value of 0, and will not produce results with a formulary tied to non-DS plans with a DS benefit design in that state.

```
* PROGRAM: OOPCV1P.SAS;
* DESCRIPTION: MAIN OOPC PROGRAM;

%LET INPUTDIR          = c:\oopc_d\input;
%LET PROGDIR           = c:\oopc_d\programs;
%LET FORMDIR           = c:\oopc_d\formulary;
%LET PLANFILEDIR       = c:\oopc_d\programs;
%LET JSONDIR           = c:\oopc_d\programs;
%LET OUTPUT            = c:\oopc_d\output;
%LET JSONFILES         = c:\oopc_d\pbp;
%LET JSONTEMP          = c:\oopc_d\pbptemp;
%LET INPUTJSONDIR      = c:\oopc_d\input\jsonlib;
%LET DSFORM_FLAG       = 0;


%OOPCV1M(RUNYEAR              =2026,
      INP                     =IN1.PERSON,
      INPUTJSONDIR            =&INPUTJSONDIR,
      FORMULARY               =&FORMDIR);
```

Then run **OOPCV1P.SAS.**

When checking the SAS Log for the run, the run time is determined by looking at the last few lines of a successful run. For example,

```
NOTE: The SAS System used:
    real time        1:36.67
    cpu time         43.10 seconds
```

The resulting Excel spreadsheet file (.xlsx) will be written to the designated output file directory when the program finishes running successfully. The field **PartD** displays the Part D OOPC.

An example (truncated) of the resulting spreadsheet output is shown below (test data):

CONTRACT_NUMBER	PLAN_ID	SEGMENT_ID	PLAN_NAME	BENEFIT_YEAR	PARTD
HXXXX	001	000	RFB MA-PD A/B Full Network EA (PFFS)	2026	321.668
HXXXX	001	000	RFB MA-PD A/B DS (HMO-POS)	2026	178.0785887

CONTRACT_NUMBER	PLAN_ID	SEGMENT_ID	PLAN_NAME	BENEFIT_YEAR	PARTD
HXXXX	001	000	RFB MA-PD A/B Partial Network AE (PFFS)	2026	279.331
HXXXX	001	000	MA-PD A/B SNP Chronic/Disabling EA (HMO-POS SNP)	2026	129.008346
HXXXX	001	000	MA-PD A/B Full Network BA (PFFS)	2026	129.008346
HXXXX	001	000	RFB MA-PD A/B DS (PPO)	2026	178.0785887
HXXXX	015	000	MA-PD A/B EA (PPO)	2026	279.331
HXXXX	002	000	RFB MA-PD A/B DS (HMO)	2026	129.008346
RXXXX	001	000	MA-PD A/B BA (Regional PPO)	2026	156.4093657
SXXXX	008	000	PD-Only EA (PDP)	2026	178.0785887

Note: The PBP_Version_Date is displayed for reference purposes.

Rerunning the Model

Change Plan Benefits for a Plan: To change the plan benefit assumptions for the same plan(s), first modify the appropriate PBP data entry.

Change Plans: To change plans, you must modify the PBP data entry, change the PLANFILE.TXT and if necessary, the formulary .txt files. If the PBP data entry is in a different JSON file you will need to update the reference in JSONfile.TXT.

Change Formulary files/Same plan: To change formulary assumptions for the same plan(s), change the formulary.txt files.

For any of the above changes, after changing input files, and rerunning **PARTD_FORM.SAS**, as necessary, rerun **OOPCV1P.SAS**, while changing the Excel output file name.

Contents of the Output (Excel) File

The output from the OOPC Model is a single Excel file. The table below lists the labels as they appear in the output file and in the corresponding detailed heading.

Note: Labels used in the output file are restricted to no more than 32 characters by SAS.

OOPCS 2026 tab

Label Used in Output	Detailed Heading/Description
Contract_ID	Contract Number
Plan_ID	Plan ID
Segment_ID	Segment ID
Plan_Name	Plan Name
Benefit_Year	Benefit Year/PBP for Estimated OOPC Values
PartD	Part D OOPC Value

OOPC DS 2026 tab

Label Used in Output Files	Detailed Heading/Description
Contract_ID	Contract Number
Plan_ID	Plan ID
Segment_ID	Segment ID
Formulary_ID	Formulary ID
PartD	Part D OOPC Value

Contents of the ZIP File (CY 2026 Bid Review OOPC Model.ZIP)

1. **Input.zip**

AE_CALCS.SAS
BA_CALCS.SAS
BASEID_DRUGS.SAS
BENE_SCRIPT.XPT
BENE_RXCUI.XPT
BENEFIT_OOPC_COST_CALCULATION.SAS
BUILD_FFF_FORMAT.SAS
BUILD_LOOKUP_CMS_SUB.SAS
BUILD_LOOKUP_SUB.SAS
BUILD_LOOKUP_THERASUB.SAS
BUILD_NONFORM_FORMAT.SAS
BUILD_TAG_RANK_FORMAT.SAS
CLEANUP.SAS
CONVERT.SAS
DS_CALCS.SAS
EA_CALCS.SAS
EXCEPT_TA.SAS
FFS_BENEFIT_OOPC_COST_CALCULATION.SAS
FRF_XWALK.XPT
JSON_IMPORT.SAS
OOPCLIB_PARTD.CPORT
OOPCV1M.SAS
PARTD_AUTOMAP.TXT
PBP_DRUG_VARIABLES.SAS
PBP_IMPORT.SAS
PBP_IMPORT_CMS.SAS
PBP_IMPORT_PARTD_CMS.SAS
PERSON.XPT
PLAN_CATNAME_NEW.SAS
PLAN_LEVEL.SAS

RXCUI_REFERENCE.XPT
SHORTEN.TXT

2. **Programs.zip**
CIMPORT.SAS
OOPCV1P.SAS
PARTD_FORM.SAS

Development of the Part D OOPC Data

The CY 2026 Part D Bid Review OOPC Model was developed using the methodology summarized below.

Average monthly Part D OOPC values are calculated for each health plan. CMS applies the data entered into the Plan Benefit Packages (PBPs) to compute the OOPC values based on formulary and co-payments/coinsurance for each drug.

CMS made the following basic assumptions related to the OOPC estimates for prescription drugs:

- PDE National Drug Codes (NDCs) are mapped into RxCUI codes to apply a particular plan's tier-formulary based cost sharing. The PDE drug events used by the CY 2026 Part D Bid Review OOPC model is based on a random 0.1% sample of 2024 Part D enrolled Medicare beneficiaries.

- PDE claims data (2024) were used to calculate average drug prices.

- All vaccines are excluded from the Beneficiary Script input file.

- Insulin cost sharing applies regardless of the phase until the out of pocket threshold is reached.

Part D OOPC

The estimated OOPC values are based upon the drug information found in the PDE file provided for the individual sample beneficiaries. The beneficiaries used to identify the drug utilization come from a random 0.1% sample of Part D enrolled Medicare beneficiaries in 2024. The data are used in conjunction with the Calendar Year (CY) 2026 Plan Benefit Packages submitted by plans that detail the drug benefit cost sharing and plan coverage as well as the CY 2026 plan-level formulary submissions. The NDC on each PDE record is mapped into a RxCUI using the appropriate CY 2026 CMS formulary reference file (FRF) released in March 2025. The model will be refreshed prior to the bid deadline to incorporate changes from the May 2025 FRF.

An average price for each RxCUI is calculated using the 2024 PDE claims data. The average price is calculated as the total gross expenditure (ingredient cost + dispensing fee + taxes + vaccination fee) divided by the number of 30-day equivalent prescriptions.

The average price for selected drugs subject to Maximum Fair Price (MFP) at the RxCUI level is calculated using the MFP for initial price applicability year 2026. More specifically, the average price is calculated using the single MFP NDC-9 price per unit applied to the quantity associated with a 30-day equivalent prescription.

Using each plan's drug coverage status and PBP-based cost-sharing information (deductible, copayments and/or coinsurance in the initial coverage phase, etc.), the beneficiary's OOPC values are calculated. The calculations are performed according to the type of Part D plan (Defined Standard, Basic Alternative, Actuarially Equivalent Standard, or Enhanced Alternative) and the associated cost-share structure. The calculations are based on the assumption that each prescription is for a one-month (30-day) supply of drugs (rather than a 60- or 90-day supply) from an In-Network Pharmacy. In the event that both a preferred and a standard pharmacy exist, the calculations are based on the preferred pharmacy cost sharing. If a particular PDE record in the 0.1% cohort reflects an extended day supply, this would be considered as multiple one-month fills.

Substitution is assumed such that when a generic version of a brand drug exists and is covered on the plan's formulary, the generic version is the one included in the calculations, provided it has lower cost sharing. Food and Drug Administration (FDA) application type is utilized to determine the applicable/non-applicable status of drugs.

In the instances when both a brand drug and its AB rated generic are not covered on a plan's formulary, the OOPC model uses an algorithm whereby cost sharing will be randomly treated as either:

1) Beneficiary pays the full retail cost, based on PDE average price [there is a 49% probability of this outcome],

2) Beneficiary pays the cost sharing for a potential therapeutic alternative covered on the formulary, where the model selects the cost sharing for one of the covered potential formulary alternatives [there is a 36% probability of this outcome], or

3) Beneficiary pays the cost sharing for the formulary exception tier(s) [there is a 15% probability of this outcome].

This data creation process results in a file that includes the total cost of the drug for each sample beneficiary and prescription, as well as each plan's associated cost-sharing structure for that drug. The beneficiary level OOPC values are then averaged at the plan level (across all beneficiaries in the data sample) in order to yield nationally representative data.

Troubleshooting

Below are several areas where users may have problems running the model:

Wrong or Missing Directory Locations

If an “input” directory is empty, the following type of error can show up in the SAS log while attempting to run the **CIMPORT.SAS** or **Part D_FORM.SAS** programs.

```
NOTE: Library IN does not exist.
ERROR: Library IN does not exist.
NOTE: Library OUTPUT does not exist.
ERROR: Physical file does not exist, c:\oopc_d\input\person.xpt
```

If an incorrect directory name for input data is listed in the OOPCV1P.SAS program, the following type of error may be displayed in the SAS log.

```
%LET DIR  = c:\oopc_d\formulary (correct)
%LET DIR  = c:\oopc_d\form (incorrect)
```

```
ERROR: Library FORMULARY does not exist.
ERROR: Unable to open catalog FORMULARY.FORMATS.
```

Missing Plans in PBP data

If some of plans that are included in the PLANFILE.txt file but that are not included in the PBP data, the following type of error may appear in the SAS log while attempting to run the OOPCV1P.SAS program.

```
ERROR: Numeric length cannot be used with character variable
LAN_COINS_PRE.
ERROR: Numeric length cannot be used with character variable
PLAN_COPAY_PRE.
```

Problems with Output Files

Each new SAS run should have a new unique output file name designated in the **OOPCV1P.SAS** program. If you do not change the name from a previously created Excel file, the new SAS run will overwrite the old file contents, or if the current Excel file is open, will not produce output at all. An example error message is shown below:

```
ERROR: The MS  Excel  table  OOPCS_2026  has  been opened for OUTPUT.
This table already exists, or there is a name conflict with an existing object.
This table will not be replaced. This engine does not support the REPLACE
option.
ERROR: Export unsuccessful. See SAS Log for details.
```

Another message will be generated if you forget to create an output directory. For example,

```
ERROR: Connect: 'c:\oopc_d\output\OOPC_RUN20260415.xlsx' is not a valid
path. Make sure that the path name is spelled correctly and that you are
connected to the server on which the file resides.
```

```
ERROR: Error in the LIBNAME statement.
```

Also, you may submit a run, find no “Error” messages in the **OOPCV1P.SAS** program, and yet find no Excel output file. One way this can happen is if the plan identifiers in the PLANLIST.TXT file are filled out without the final 3 segment identifiers, e.g.,:

```
S9999001
```

Problems with Insufficient Hard Drive Space

If you have been running the model repeatedly, you may encounter the following error message:

```
WARNING: File 'WORK.xxxxxx.DATA' is shorter than expected.
ERROR: The file WORK.xxxxxx.DATA is shorter than expected.
ERROR: The file WORK.xxxxxx.DATA is shorter than expected.
ERROR: The file WORK.xxxxxx.DATA is shorter than expected.
WARNING: Data set WORK.yyyyyy was not replaced because this step was
stopped.
ERROR: The open failed because library member WORK.xxxxxx.DATA
is damaged.
ERROR: The open failed because library member WORK.xxxxxx.DATA
is damaged.
ERROR: The open failed because library member WORK.xxxxxx.DATA
is damaged.
```

This problem means that SAS does not have sufficient hard disk space for its temporary files. You can reboot your machine so that more memory is available to SAS. Also, check that you do not have 'leftover' SAS temporary directories. An example of SAS temporary directories that may remain from other sessions under 'My Computer' is:

```
c:\Documents and Settings\yourname\Local Settings\Temp\SAS Temporary
Files\
with subdirectories such as:
TD_xxxxx
SAS_util000100000150_machinename
```

Part D Output Expected, but Blank

When you have completed your PBP data entry, make sure you have exit/validated from the program. Also, output may not be produced if the formulary IDs are not formatted correctly (i.e., tab-delimited) or if they are formatted differently in the two input files:

FORMULARY.TXT and **PLAN_FORMULARY.TXT.**