How to use PROC COPY in SAS?

Authored by

stats writer

November 19, 2025

The PROC COPY in SAS is an indispensable utility procedure designed for efficient management of data sets and other members within or across SAS libraries. Its primary function is to create an exact replication of source members, ensuring that all associated metadata and attributes--such as labels, formats, indexes, and integrity constraints--are meticulously preserved in the destination location.

Beyond simple duplication, PROC COPY is a versatile tool utilized for various administrative and data management tasks. These tasks include renaming data sets during the copy process, migrating data from one SAS library to another (potentially involving a change in the underlying storage engine), and performing crucial backup and restoration operations for large collections of data sets. Understanding its parameters is essential for any advanced SAS programmer seeking efficient data migration capabilities.

You can use the **PROC COPY** statement in SAS to copy a data set from one library to another, managing your project resources effectively.

This statement employs a straightforward yet powerful syntax that requires the specification of source and destination libraries, as well as the members intended for migration.

Understanding the Core Syntax of PROC COPY

The fundamental structure of the PROC COPY command relies on clearly defining the environment, including the input and output locations, and specifying which types of members should be processed. This structured approach ensures data integrity and minimizes the risk of unintended actions when managing critical data sets.

The general syntax follows this pattern, utilizing global options defined on the procedure line and specific member selection within the block:

```
proc copy in=folder1 out=folder2 memtype=data;
select my_data;
run;
```

A detailed explanation of the essential components within this procedure block helps clarify the process:

IN: This option is mandatory and specifies the source SAS library (libref) where the desired member currently exists. All members referenced in the following `SELECT` or `EXCLUDE` statements must reside within this library.
OUT: This mandatory option defines the destination SAS library (libref) where the copies of the members will be stored. The specified output libref must have been previously defined using the

LIBNAME statement.
MEMTYPE: This optional parameter allows the user to specify the type of member(s) to be copied. By setting `MEMTYPE=DATA`, we ensure that only data sets (and not other member types like catalogs or views) are considered during the copy operation. Other common values include `ALL`, `CATALOG`, or `VIEW`.
SELECT: The `SELECT` statement is used to explicitly name the specific members (e.g., `my_data`) that should be copied from the `IN=` library to the `OUT=` library. Alternatively, omitting the `SELECT` or `EXCLUDE` statements results in copying all members present in the source library.

The subsequent step-by-step example demonstrates the practical application of **PROC COPY**, outlining the necessary environment setup, data creation, and final migration process between two distinct libraries.

Step 1: Setting Up the Environment and Creating the Source Data Set

Before initiating any copy operation, we must first ensure that the source data exists and is accessible. This step involves creating a temporary data set and then using standard SAS programming methods to visualize its structure and contents, confirming its readiness for the migration process.

We begin by creating a sample data set named **my_data**. This data set contains sample statistical information relevant to various basketball teams, including their scores and assists. This step leverages the standard SAS `DATA` step combined with the `DATALINES` statement for rapid data entry.

```
/*create dataset in WORK library*/
data my_data;
input team $ points assists;
datalines;
Mavs 14 9
Spurs 23 10
Rockets 38 6
Suns 19 4
Kings 30 4
Blazers 19 6
Lakers 22 14
Heat 19 5
Magic 14 8
Nets 27 8
;
```

run;

/*view dataset using PROC PRINT to verify creation*/
proc print data=my_data;
run;

Executing this code creates the **my_data** table in the temporary `WORK` SAS library. The `PROC PRINT` output confirms that the variables (`team`, `points`, `assists`) and all observations have been correctly loaded into the data structure, preparing it for permanent storage and eventual migration.

Obs	team	points	assists
1	Mavs	14	9
2	Spurs	23	10
3	Rockets	38	6
4	Suns	19	4
5	Kings	30	4
6	Blazers	19	6
7	Lakers	22	14
8	Heat	19	5
9	Magic	14	8
10	Nets	27	8

Step 2: Defining Libraries and Saving the Source Data to the Initial Location

In a production environment, data is rarely kept in the temporary `WORK` library. Therefore, the next step involves defining a permanent SAS library (a libref) that points to a specific physical directory on the operating system. This acts as our source location for the PROC COPY operation.

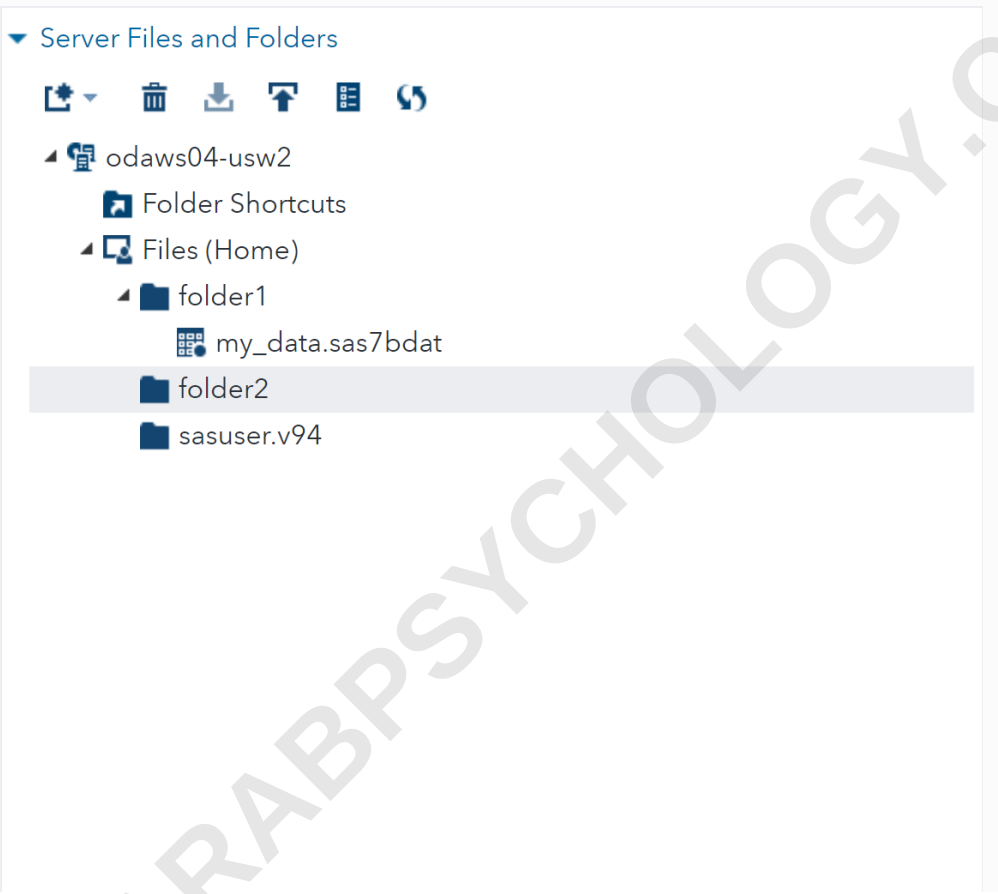
We use the critical LIBNAME statement to assign the alias **folder1** to the source directory path. Following this definition, we use a subsequent `DATA` step with the `SET` statement to copy the temporary **my_data** from `WORK` into this newly defined permanent storage location, `folder1`.

/*define source library path*/
libname folder1 '/home/u13181/folder1/';

/*save dataset to the permanent library called folder1*/
data folder1.my_data;

```
set my_data;
run;
```

This process successfully establishes `folder1` as the primary source SAS library containing the **my_data** data set. It is essential that the physical path specified in the LIBNAME statement (`/home/u13181/folder1/`) exists and that the SAS session has the necessary read/write permissions to interact with that directory. Verification via the operating system's file navigation confirms the successful creation and placement of the data set in the defined path.



Step 3: Utilizing PROC COPY to Migrate the Data Set to a New Library

With the source data permanently stored in `folder1`, we can now define the destination library and execute the core migration procedure. This step showcases the power of PROC COPY in replicating data structures without requiring complex data step programming.

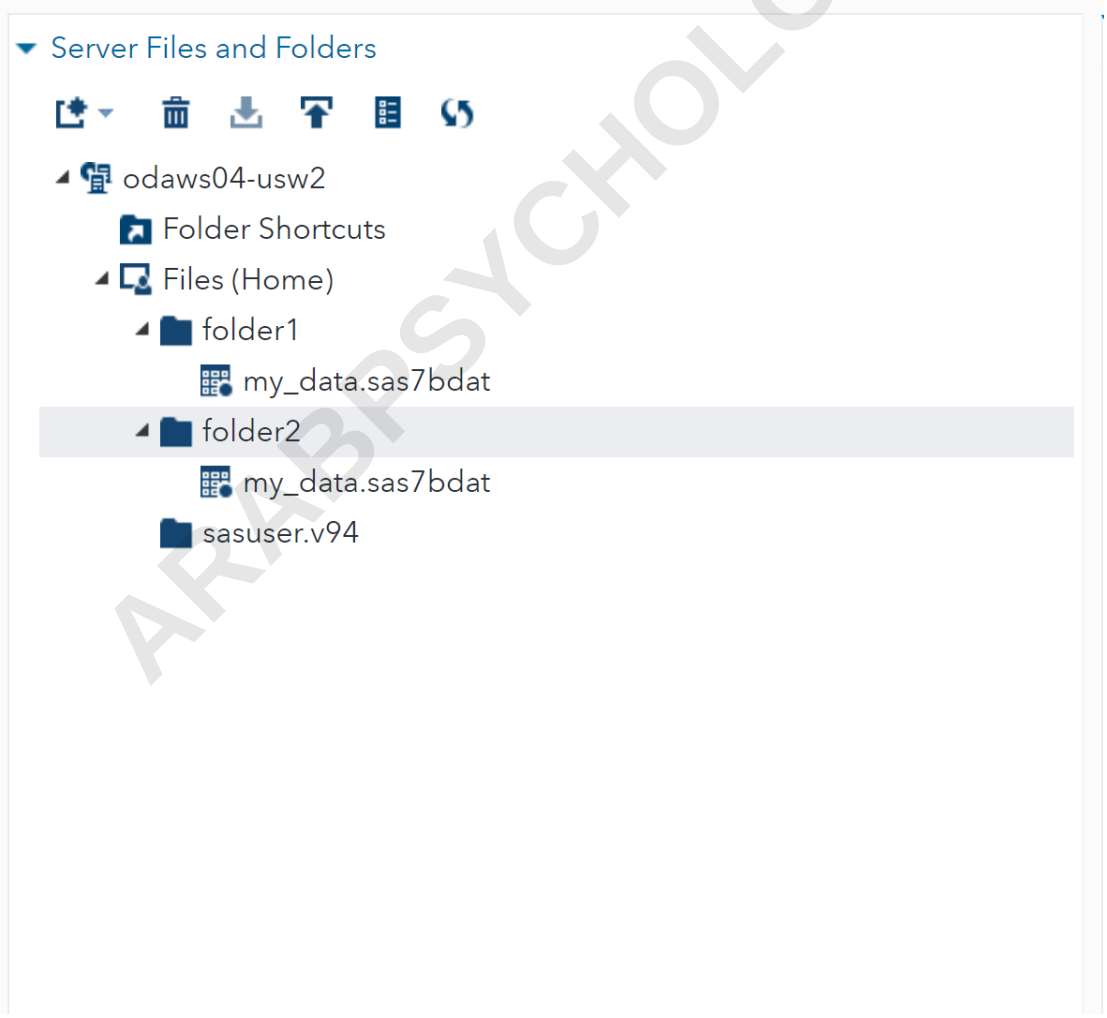
First, we must define the destination library, **folder2**, using another LIBNAME statement that points to a separate physical location. This ensures that the original and copied versions of the data set are stored independently. Once both `IN=` (folder1) and `OUT=` (folder2) are correctly defined, the

PROC COPY block is executed.

/*define library where dataset should be copied to*/
libname folder2 '/home/u13181/folder2/';

/*copy my_data from folder1 to library called folder2*/
proc copy in=folder1 out=folder2 memtype=data;
select my_data;
run;

The procedure executes rapidly, reading the specified member (`my_data`) from `folder1` and writing a bit-for-bit identical copy, including all attributes, to `folder2`. It is important to remember that PROC COPY is a non-destructive operation; the source data set remains fully intact in the original library (`folder1`) even after the copy is complete. This makes it ideal for backup, migration, and development workflows where the original source must be preserved.



Note: When using **PROC COPY**, the data set you are copying will still remain in the original library that it came from. This procedure creates a duplicate rather than performing a move operation.

Advanced Capabilities: Using SELECT and EXCLUDE Statements

For large SAS libraries containing dozens or hundreds of members, selectively copying specific items is crucial for efficiency. The `SELECT` and `EXCLUDE` statements provide granular control over which members are processed during the PROC COPY operation.

The `SELECT` statement, as seen above, specifies a list of members to be copied. If only a few members need to be transferred from a large source library, listing them explicitly is the most straightforward approach. Furthermore, SAS supports using `:` (colon) or `--` (double hyphen) notation within the `SELECT` statement to specify groups of members based on alphabetical or sequential order, respectively. For instance, `SELECT A:` would copy all members whose names begin with the letter 'A'.

Conversely, the `EXCLUDE` statement instructs PROC COPY to ignore certain members during the process. This is particularly useful when the goal is to copy almost all members from a source library, save for a few exceptions (e.g., test files or temporary data). When both `SELECT` and `EXCLUDE` statements are used simultaneously, SAS first applies the selection criteria and then removes any items specified by the exclusion list from the resulting set.

Managing Different Member Types with MEMTYPE=

A SAS library can contain various types of members besides standard data sets. These include catalogs (which store formats, informats, or screen applications), views (which are compiled query definitions), and stored programs. The `MEMTYPE=` option on the PROC COPY statement allows precise control over which of these types are considered for the copy operation.

By default, if `MEMTYPE=` is omitted, PROC COPY attempts to process all member types, which is equivalent to specifying `MEMTYPE=ALL`. However, setting `MEMTYPE=DATA` (as used in the example) ensures that only permanent data files are copied, ignoring ancillary files like catalogs, which might be specific to the original environment or engine. Programmers must carefully consider the required member types, especially when migrating complex applications that rely on formats or stored procedures, which reside in catalogs.

Performance Considerations and Alternatives to PROC COPY

While PROC COPY is highly effective for migrating entire libraries or specific members, its performance and usage context should be understood. PROC COPY performs a sequential copy of the file structure, ensuring that all attributes are maintained, making it ideal for moving data

between different operating systems or SAS engines.

For copying data within the same SAS library or between libraries using the same file system engine, performance is usually optimized. However, when migrating between vastly different engines (e.g., from a native SAS file format to a database engine via a LIBNAME engine connector), the procedure must translate the data structure, which can impact execution time for extremely large data sets.

Alternative methods exist for specific scenarios. For instance, the `DATA` step with a `SET` statement is often preferred if modifications (like column drops, additions, or transformations) are needed during the copy. If the goal is simply to rename a file within the same library without generating an exact duplicate file, the `PROC DATASETS` procedure often offers a more lightweight solution than PROC COPY.

Further Exploration of SAS Utilities

The effective management of data sets in SAS extends beyond just duplication. Understanding other utility procedures provides a comprehensive toolset for data manipulation and maintenance:

The following tutorials explain how to perform other common tasks in SAS, complementing the functionality of PROC COPY: