# PP06 SAS Performance Qualification For Clinical Programmers

**Authors: Tatiana Volodina** volodina.tatyana@gmail.com, **Andrey Myslivets** andrewmyslivets@gmail.com, **Data MATRIX Ltd.**

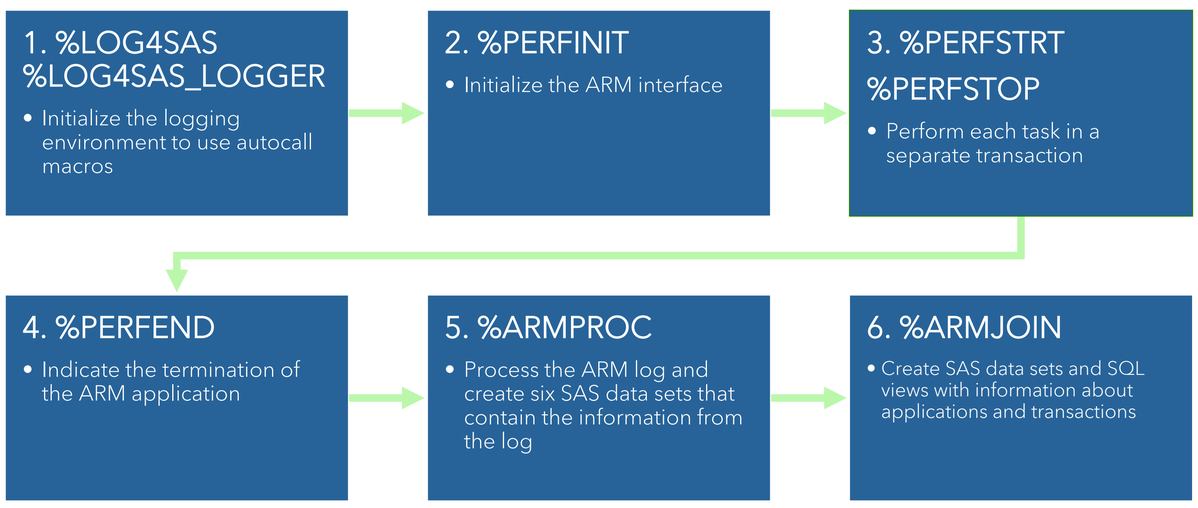
## Introduction

When starting to use a new system, tool or software, certain qualification checks ought to be performed to ensure its correct work, such as design qualification (DQ), installation qualification (IQ), operational qualification (OQ), process qualification (PQ) and performance qualification (P1Q).
SAS provides sufficient tools for performing most of these checks, such as SAS Installation Qualification Tool (SAS IQ) and the SAS Operational Qualification Tool (SAS OQ). These tools support the qualification aspect of the essential migration, integration, and verification processes users need to move from previous versions of SAS to later releases. However, performance qualification cannot be unified in the same way. It implies field-specified tests that check the system performance in real-world scenarios.
This poster presents a program code designed to run performance qualification tests that have been selected taking into account the specifics of clinical programming. Methods used to perform these tests are described, as well as the produced results.
The presented program code cannot only be used for performance qualification checks, but it presents an opportunity to compare the efficiency of different programming approaches and SAS procedures, which can be useful in everyday work.

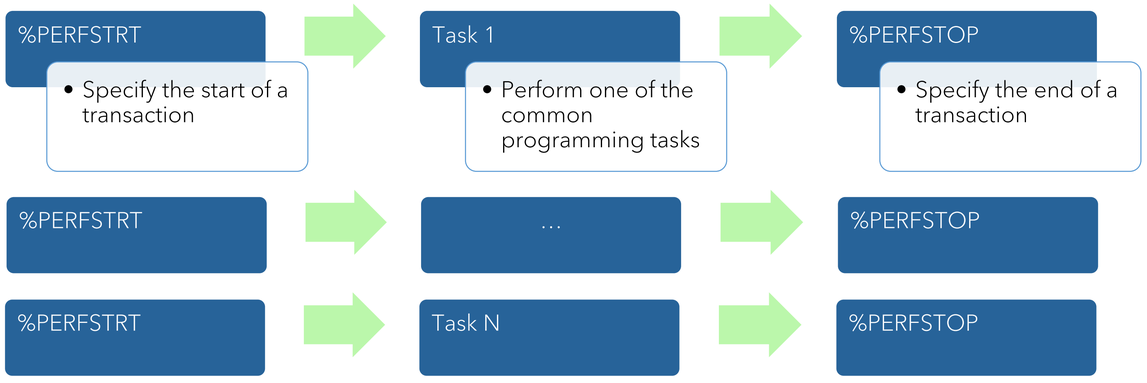
## Objective

The program code performs qualification tests that check calculations, procedures, methods, etc. that are used the most by a programmer in order to create statistical outputs for Clinical Study Reports (CSRs).

The produced results show how much time the system uses for performing each task, so that the user can identify the system capability and its weak points in order to take corrective actions to improve it.




## Tasks




The following tasks have been identified as the most common and which performance is crucial for a clinical programmer's routine work:

- Sample size analysis;
- Creating a randomization list for 4000 subjects;
- Deleting 10 datasets;
- Creating a dataset with several variables for 4000 subjects;
- Merging datasets;
- Calculating descriptive statistics using *proc means*;
- Calculating frequency statistics using *proc freq*;
- Performing t tests using *proc ttest*;
- Performing nonparametric tests using *proc npar1way*;
- Performing ANCOVA with random factor using *proc mixed*;
- Creating a table using *proc report*;
- Creating plots using *proc gplot*.

## Results

The %ARMPROC and %ARMJOIN macros produce the results that are usually used for writing P1Q documentation. Results are attached as appendixes to P1Q documentation.

Performance Qualification Results 1
17:27 Wednesday, August 16, 2017

Txn Name=Tab_01_Power_SampleSize

| txname | Obs | ARM Call datetime | Delta Elapsed Time | Delta User CPU Time | Delta System CPU Time | Non-CPU Time |
|---|---|---|---|---|---|---|
| | 1 | 16AUG2017:14:27:29.298 | 0:00:00.000 | 0:00:00.000 | 0:00:00.000 | 0:00:00.000 |
| | 2 | 16AUG2017:14:27:31.174 | 0:00:01.876 | 0:00:00.296 | 0:00:00.094 | 0:00:01.486 |
| txname | | | 0:00:01.876 | | 0:00:00.094 | 0:00:01.486 |

| txname | Obs | Call Sequence Identifier | App ID | Txn Class ID | Start Handle | Txn Detail | Metric 1 Type | Metric 2 Type | Metric 3 Type | Metric 4 Type | Metric 5 Type |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | UPD00001 | 1 | 1 | 1 | | Count64 | Gauge64 | Gauge64 | Gauge32 | Gauge32 |
| | 2 | UPD00002 | 1 | 1 | 1 | | Count64 | Gauge64 | Gauge64 | Gauge32 | Gauge32 |

| txname | Obs | Metric 6 Type | Metric 1 Description | Metric 2 Description | Metric 3 Description | Metric 4 Description | Metric 5 Description | Metric 6 Description |
|---|---|---|---|---|---|---|---|---|
| | 1 | | _IOCOUNT_ | _MEMCURR_ | _MEMHIGH_ | _THREADCURR_ | _THREADHIGH_ | |
| | 2 | | _IOCOUNT_ | _MEMCURR_ | _MEMHIGH_ | _THREADCURR_ | _THREADHIGH_ | |

| txname | Obs | String Type | String Description | Txn User CPU Time | Txn System CPU Time | Metric 1 Value | Metric 2 Value | Metric 3 Value |
|---|---|---|---|---|---|---|---|---|
| | 1 | | | 0:00:01.622410 | 0:00:01.388408 | 4719357 | 10723328 | 10989568 |
| | 2 | | | 0:00:01.918812 | 0:00:01.482009 | 5834891 | 17539072 | 17805312 |

| txname | Obs | Metric 4 Value | Metric 5 Value | Metric 6 Value | String Value | Format 2 Data Buffer | Txn Status | updtdata |
|---|---|---|---|---|---|---|---|---|
| | 1 | 6 | 6 | | | | . | START |
| | 2 | 6 | 6 | | | | 0 | STOP |

## Methods

1. Initializing the logging facility for SAS programs is necessary if you use the logging facility autocall macros. The %LOG4SAS macro initializes the logging environment, and the %LOG4SAS_LOGGER macro defines a logger. We use the option "level = info" in order to get information that highlights the progress of an application.

2. ARM (Application Response Measurement) is an application programming interface which is used to monitor the availability and performance of software applications. ARM allows to measure application availability, performance, usage, and transaction response time.
The %PERFINIT macro names the application instance and initializes the ARM interface.

3. The %PERFSTRT macro specifies the start of a transaction.
Within each transaction, one of the programming tasks is performed.
The %PERFSTOP macro specifies the end of a transaction.

4. The %PERFEND macro indicates the termination of the application. The %PERFEND means that the application does not issue any more ARM calls.

5. The %ARMPROC macro processes the ARM log and creates six SAS data sets that contain the information from the log.

6. The %ARMJOIN macro reads the six SAS data sets that are created by the %ARMPROC macro. It merges the information from those data sets to create data sets and SAS views for easier reporting of ARM data (SQL views).

## ARM Log

While each performance check is carried out, the result is written into the log.
The file with the ARM log is presented as a table on the picture below.
Metrics for each test are presented.

However, even using the description of each metric, the file is not easy to understand. For this reason, the %ARMPROC and %ARMJOIN macros are used. They create a more understandable output that is presented in the Results section.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | 1819528054.832000 | 1 | 2.808018 | 4.102826 | Perf_App | tvolodina | | | | | | | | | |
| G | 1819528054.882000 | 1 | 1 | Tab_01_Power_SampleSize | | _IOCOUNT_ | Count64 | _MEMCURR_ | Gauge64 | _MEMHIGH_ | Gauge64 | _THREADCURR_ | Gauge32 | _THREADHIGH_ | Gauge32 |
| S | 1819528054.882000 | 1 | 1 | 1 | 2.823618 | 4.102826 | 4613178 | 10985472 | 11251712 | 6 | 6 | | | | |
| P | 1819528060.294000 | 1 | 1 | 1 | 3.151220 | 4.243227 | 0 | 5728988 | 18329600 | 18595840 | 6 | 6 | | | |
| G | 1819528060.304000 | 1 | 2 | Tab_02_Randlist_for_4000_patients | | _IOCOUNT_ | Count64 | _MEMCURR_ | Gauge64 | _MEMHIGH_ | Gauge64 | _THREADCURR_ | Gauge32 | _THREADHIGH_ | Gauge32 |
| S | 1819528060.304000 | 1 | 2 | 2 | 3.151220 | 4.258827 | 5741276 | 18329600 | 18595840 | 6 | 6 | | | | |
| P | 1819528061.727000 | 1 | 2 | 2 | 3.276021 | 4.711230 | 0 | 11355360 | 19644416 | 23838720 | 6 | 6 | | | |
| G | 1819528061.737000 | 1 | 3 | Tab_03_Delete_10_DS | | _IOCOUNT_ | Count64 | _MEMCURR_ | Gauge64 | _MEMHIGH_ | Gauge64 | _THREADCURR_ | Gauge32 | _THREADHIGH_ | Gauge32 |
| S | 1819528061.737000 | 1 | 3 | 3 | 3.276021 | 4.726830 | 11355360 | 19644416 | 23838720 | 6 | 6 | | | | |
| P | 1819528062.020000 | 1 | 3 | 3 | 3.276021 | 4.758030 | 0 | 11363552 | 19644416 | 23838720 | 6 | 6 | | | |
| G | 1819528062.020000 | 1 | 4 | Tab_04_DS_creation_3_var_4000_patients | | _IOCOUNT_ | Count64 | _MEMCURR_ | Gauge64 | _MEMHIGH_ | Gauge64 | _THREADCURR_ | Gauge32 | _THREADHIGH_ | Gauge32 |
| S | 1819528062.020000 | 1 | 4 | 4 | 3.276021 | 4.758030 | 11363552 | 19644416 | 23838720 | 6 | 6 | | | | |
| P | 1819528062.300000 | 1 | 4 | 4 | 3.276021 | 4.773630 | 0 | 11627232 | 19644416 | 23838720 | 6 | 6 | | | |

## Metrics

| Metric Name | Metric Type | Description |
|---|---|---|
| _IOCOUNT_ | COUNT64 | The total number of disk, tape, or related input and output operations at each %PERFSTRT and %PERFSTOP event |
| _MEMCURR_ | GAUGE64 | The current value for memory used in the process at each %PERFSTRT and %PERFSTOP event |
| _MEMHIGH_ | GAUGE64 | The highest amount of memory used for the life cycle of the current process at each ARM event |
| _THREADCURR_ | GAUGE32 | The current thread count of the process at each ARM event |
| _THREADHIGH_ | GAUGE32 | The highest number of active threads for the life cycle of the current process at each ARM event |

## Possible uses

This technique is useful in all tasks related to measuring the performance of any SAS codes.
Except the main task (validation of the SAS platform or PQ check documentation) it can be successfully used in:

- Checking the SAS performance on different devices;
- Comparing different approaches of problem solving;
- Choosing the most effective algorithm for each programming task.

In case you need to produce P1Q documentation, assistance in writing your code or in interpreting the results, feel free to send your request to us via email.