



Customising SAS OQ to Provide Business Specific Testing of SAS Installations and Updates

Steve Huggins





Contents

- SAS Installation Qualification Tool (a brief note)
- SAS Operational Qualification Tool
- Understanding the OQ
- Customising the OQ
 - Creating a table file
 - Using the %rccheck macro
- Using the OQ across Business



The SAS Software Qualification Tools

- **SAS Installation Qualification Tool (SAS IQ)**
 - Checks the files installed as part of SAS Foundation

- **SAS Operational Qualification Tool (SAS OQ)**
 - Ensures that the SAS installation executes SAS code correctly



SAS Installation Qualification Tool

- Used to validate SAS after installation
- Validates the SAS executable files such as .dll and .exe
- Uses a hashing algorithm (MD4) to generate a check sum and compares this with expected values
- Detailed reports are delivered to a selected location
- Reports are produces as HTML and PDF files



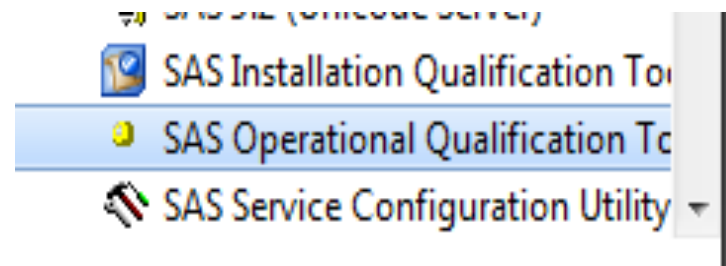
SAS Operational Qualification Tool

- Run after installing or updating SAS Software
- Ensure that SAS behaves correctly
- Executes test programs to check SAS functionality
- Identifies when errors are generated by SAS programs
- Can also be used to validate the generated outputs



Running the SAS OQ Tool

- Can execute from the command line
 - Allowing more control of the process
 - Can be run from a batch file
- `sasoq.exe -tables *:custom -outdir c:\test`
- Can be run from the start menu (SAS 9.2 onwards)



- This executes “ `sasoq.exe -tables *.*` ”



SAS OQ Results

- Detailed reports are delivered to the selected location
- A time stamped folder is created each time
- Reports are produces as HTML and PDF files

Pass/Fail Results			
Component	Table	Test	Result
BASE	tstbase	tstbrs	Pass
	tstbase	tstcalen	Pass
	tstbase	tstchrt	Pass
	tstbase	tstcmp	Pass
	tstbase	tstcor01	Pass
	tstbase	tstdoc	Pass
	tstbase	tstedt	Pass
	tstbase	tstform	Pass



Understanding the OQ: Overview

- The OQ is launched via a single executable (sasoq.exe)
 - Requires a location for results report
- Test programs are executed independently
- Each test will either Pass or Fail – Results are captured
- Failures are identified from a system return code when:
 - Error are found in the SAS log
 - SAS Terminates with an error state
- Generate Output Reports



Understanding the OQ: Components

- The OQ components are located within the SAS Installation folder: <SASROOT>\sastest
- The SAS OQ consists of the following components
 - The Executable File - (sasok.exe)
 - Folder Structure - Several sub folders within sastest
 - Table Files (.tab) - Define the individual tests to be run
 - Test Programs - SAS programs that can run independently
- By Default there is a subfolder for each SAS component installed. Base, SAS/STAT, etc



Understanding the OQ: Process

- When the OQ (sasoq.exe) is run the control program will examine all subfolders within \sastest:
 - Check for the presence of a table (.tab) file and process it
- Each table file defines a number of tests to be run
 - These usually execute SAS programs located in the same folder
 - The tests within each folder are grouped together in the results
- Pass or Fail results are collated for each test and used to generate the final output reports



Customising the OQ

- New functional tests can be added to the OQ process
 - These can be added to an existing set of tests or a new set of tests can be created
- Add new folder to organise the new tests
- Create table file to define the new tests
- Select and modify the test programs to be used
- Once added new tests will automatically be executed and included reports



Customising the OQ : Table File

- The table (.tab) file defines the tests that will be run by the OQ tool
- Written using SAS Table Language statement
- Each test is defined within a “test block”
- Each test block will invoke SAS and execute a test program
- Each test block will capture the system return code and check it against the expected



Table Language Statements

- &set – Defines variables (resolved using @ sign)
- &test – Defines a test block, parameters are in braces { }
- &run – Specify a program to be run
- &infile – Defines a file shortcut (usually a SAS program)
- &rc – Validates the return code (logs a pass or fail)
- Note: &s do not denote SAS macro variables



Customising the OQ : Typical table file

- Create a table file variable:

```
&set( sasopts: -autoexec &infile(sas:assert:base)  
      -nosyntaxcheck -nostimer)
```

- Define a test block:

```
&test test_name {  
&run(sas:(-sysin &infile(sas:tstsql:base) @sasopts) )  
&rc(0) }
```

Example taken from the base SAS table file



Customising the OQ : Test Programs

- Any SAS program can be used
 - Standard macros (with known results)
 - Tests using specific procedures
 - Can test results against standard benchmark data
- Tests can be as simple as, “no errors = Pass”
- Test programs can validate generated results
 - Using the %rccheck macro
 - Single values
 - SAS datasets containing results



Using the %rccheck Macro

- Created by the assert program in the sastest\base folder
 - This can be defined in the autoexec for each test
- Use within test programs to flag failures when a SAS error is not generated
- Forces SAS to terminate with a “failed” return code
- Checks a macro variable against the expected value
 - This can be an automatic macro variable such as &sysinfo or one specifically created within a data step
- Used when checking results against benchmark tables



Using the %rccheck Macro: Example

Create test data;

```
data test_data;
```

```
  input a b c;
```

```
cards;
```

```
1 2 3
```

```
run;
```

Test functions and put the results into macro vars;

```
data _null_;
```

```
  set test_data;
```

```
  test1 = sum(a,b,c);
```

```
  * Test SUM function;
```

```
  call symputx('sum_test',test1);
```

```
  * Copy result to a macro var;
```

```
  test2 = mean(a,b,c);
```

```
  * Test MEAN function;
```

```
  call symputx('mean_test',test2);
```

```
  * Copy result to a macro var;
```

```
run;
```

```
%rccheck(&sum_test,6);
```

```
  * Check the SUM result:  Pass;
```

```
%rccheck(&mean_test,3);
```

```
  * Check the Mean result:  Fail;
```



Using Proc Compare with %rccheck

- Use Proc Compare to check the outputs generated in a test program against benchmark results
- The benchmark data can be hard coded into the test program or held in a separate dataset
- The **&sysinfo** automatic macro variable will be set to zero if two compared datasets are found to be identical
- Use the %rccheck macro after Proc Compare to check the value of &sysinfo (zero = Pass)

%rccheck(&sysinfo,0);



Using the OQ across Business

- 1) Select the Test Programs (must be fixed)
- 2) Create the Infrastructure
- 3) Capture Input and Benchmark Data (must be fixed)
- 4) Configure the Table File(s)
- 5) Edit the Test programs
- 6) Deploy Test Programs



Summary

- The IQ and OQ tools allow you to confirm that SAS runs consistently
- The OQ tool is customisable:
 - Custom test programs
 - Different tools allow you to devise a variety of test
 - Integrated into the normal process, execution and reporting
- Test programs and benchmark data can be stored in a central location and used across a business



Summary

- Any Questions?

- For Further Reading: SAS Documentation
 - SAS® 9.1.3 Qualification Tools. User's Guide
 - SAS® 9.2 Qualification Tools. User's Guide
 - SAS® 9.3 Qualification Tools. User's Guide

Steve Huggins
Amadeus Software Ltd.

Tel: +44 (0)1993 848010
e-Mail: steve.huggins@amadeus.co.uk
Web: www.amadeus.co.uk