



# **SAS<sup>®</sup> 9.2 Qualification Tools User's Guide**



## **Copyright Notice**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® 9.2 *Qualification Tools User's Guide*, Cary, NC: SAS Institute Inc., 2008.

### **SAS® 9.2 Qualification Tools User's Guide**

Copyright © 2008, SAS Institute Inc., Cary, NC, USA.

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, by any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc. Limited permission is granted to store the copyrighted material in your system and display it on terminals, print only the number of copies required for use by those persons responsible for installing and supporting the SAS programming and licensed programs for which this material has been provided, and to modify the material to meet specific installation requirements. The SAS Institute copyright notice must appear on all printed versions of this material or extracts thereof and on the display medium when the material is displayed. Permission is not granted to reproduce or distribute the material except as stated above.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

## Table of Contents

|  |           |
|--|-----------|
| <b>Introduction</b> .....  | <b>1</b>  |
| <b>SAS Installation Qualification Tool (SAS IQ)</b> .....                    | <b>3</b>  |
| <b>Overview</b> .....  | <b>3</b>  |
| <b>Windows</b> .....   | <b>3</b>  |
| Running the Installation Qualification Tool .....                            | 3         |
| Output .....   | 3         |
| OutputPath option .....  | 4         |
| <b>UNIX</b> .....  | <b>4</b>  |
| Running the SAS Installation Qualification Tool on UNIX Systems .....        | 4         |
| Reviewing the Results of the SAS Installation Qualification Tool .....       | 5         |
| <b>VMS Itanium</b> .....   | <b>6</b>  |
| Running the SAS Installation Qualification Tool on VMS Itanium Systems ..... | 6         |
| <b>SAS Operational Qualification Tool (SAS OQ)</b> .....                     | <b>9</b>  |
| <b>Overview</b> .....  | <b>9</b>  |
| <b>Windows</b> .....   | <b>9</b>  |
| Directory Structure .....  | 9         |
| Running the Operational Qualification Tool .....                             | 9         |
| Output .....   | 9         |
| Outdir Directory Structure .....   | 10        |
| Reviewing the Results of the SAS Operational Qualification Tool .....        | 10        |
| <b>UNIX</b> .....  | <b>10</b> |
| Directory Structure .....  | 10        |
| Running the SAS Operational Qualification Tool on UNIX Systems .....         | 11        |
| Output .....   | 11        |
| Outdir Directory Structure .....   | 11        |
| Reviewing the Results of the SAS Operational Qualification Tool .....        | 12        |
| <b>VMS Itanium</b> .....   | <b>12</b> |
| Directory Structure .....  | 12        |
| Running the SAS Operational Qualification Tool on VMS Itanium Systems .....  | 12        |
| Output .....   | 12        |
| Outdir Directory Structure .....   | 12        |
| Reviewing the Results of the SAS Operational Qualification Tool .....        | 13        |
| Additional Examples of Invocation .....                                      | 13        |
| Automation .....   | 14        |
| Windows .....  | 14        |
| Windows Example .....  | 15        |
| UNIX .....   | 15        |
| UNIX Example .....   | 15        |
| <b>SAS Operational Qualification Tool Table Language</b> .....               | <b>15</b> |
| Table Language General Rules .....   | 16        |
| <b>Block Identifier</b> .....  | <b>16</b> |
| &test .....  | 16        |
| <b>Command</b> .....   | <b>17</b> |
| &process .....   | 17        |
| &rc .....  | 17        |
| &run .....   | 18        |
| &set .....   | 18        |
| &unset .....   | 19        |

|   |           |
|---|-----------|
| <b>Macro .....</b>  | <b>20</b> |
| &infile .....   | 20        |
| &outfile .....  | 21        |
| &resfile .....  | 21        |
| Examples .....  | 22        |
| <b>SAS Operational Qualification Tool Command Line .....</b>    | <b>22</b> |
| <b>Perl Stream Editor Documentation .....</b>                   | <b>23</b> |
| What Can I Do with Perl REs That I Can't Do with sed REs? ..... | 23        |
| Filter File Syntax .....  | 24        |
| ped Command Line Syntax.....                                    | 24        |
| Substringing Substitute("S" command) Syntax .....               | 24        |
| Ped Filter File Examples .....                                  | 24        |
| Issues .....  | 24        |
| <b>IVFND Command Line.....</b>                                  | <b>25</b> |

# Introduction

There are two qualification tools available with SAS 9.2—the SAS Installation Qualification Tool (SAS IQ) and the SAS Operational Qualification Tool (SAS OQ). These tools will assist you in qualifying the use of the SAS software in regulated industries. Together, these tools support the qualification aspect of the essential migration, integration, and verification processes customers need to move from previous versions of SAS to later releases.

The SAS IQ assists you in demonstrating the SAS System has been installed and maintained to the manufacturer's specifications. SAS IQ verifies the integrity of each file in the SAS System 9 and provides the customer a set of reports detailing the results. SAS IQ is supported on all Windows, UNIX, and OpenVMS platforms for SAS 9.2. See “SAS Installation Qualification Tool (SAS IQ)” on page 3 for a detailed description.

*Note: The SAS Installation Qualification Tool will only validate SAS Foundation components. Java components associated with SAS Solutions are not validated by the SAS Installation Qualification Tool.*

The SAS OQ assists you in demonstrating the SAS System is operational. SAS OQ uses SAS programs provided by the component development groups and will execute, process, and report the program results. SAS OQ is supported on all Windows and UNIX platforms for SAS 9.2. See “SAS Operational Qualification Tool (SAS OQ)” on page 9 for more information.



# SAS Installation Qualification Tool (SAS IQ)

## Overview

The SAS Installation Qualification Tool (SAS IQ) validates a SAS System installation by verifying that each installed file is correct. A report is generated detailing all file results. This determination is made using the md5 algorithm to create a value for each file.

*Note: The SAS Installation Qualification Tool will only validate SAS Foundation components. Java components associated with SAS Solutions are not validated by the SAS Installation Qualification Tool.*

## Windows

### Running the Installation Qualification Tool

There are two ways to run the SAS IQ on a successful SAS 9.2 installation:

- From the start menu as follows:

```
Start → Programs → SAS → SAS 9.2 Utilities → SAS Installation Qualification Tool
```

or

- Open a command line in DOS and type the following command:

```
c:\><SASROOT>\sastest\sasiq.exe -OutputPath <dir>
```

In this command, <SASROOT> is the location where the SAS System was installed, and <dir> is an empty directory.

Both of these methods require an output directory, which must be empty.

### Output

The SAS IQ generates XML data files to hold the raw results of the validation process. The SAS System is used to translate the XML data into PDF and HTML reports. The PDF and HTML reports are identical. The PDF can be used to create hardcopies of the resulting validation. The HTML is easier to browse.

For the Windows platform, XSL translation files are also provided. These allow another means of generating HTML from the XML data file. However, the XSL translation is done dynamically.

In all cases, the output report is organized first at the SAS System component level, and then by each component's files.

## OutputPath option

The following table describes the structure of the directory used in the -OutputPath option:

| File Name      | Description   |
|----------------|---|
| Data           | Directory containing the raw validation data for each component in XML format |
| html           | Directory containing HTML files for each component                            |
| autoexec.sas   | SAS file used in producing PDF and HTML output from XML data                  |
| index.xml      | XML index file for all component summary                                      |
| index.xsl      | XML style sheet   |
| media.xsl      | XML style sheet   |
| sasiq.htm      | HTML file containing index to summary and component information               |
| sasiq.log      | SAS log file created from running the autoexec.sas program                    |
| sasiq.pdf      | PDF file containing all summary and component information                     |
| SASIQindex.map | Intermediate file used to output XML to PDF and HTML                          |
| SASIQprod.map  | Intermediate file used to output XML to PDF and HTML                          |
| sasiqtoc.htm   | HTML file used to display component information                               |

## UNIX

### Running the SAS Installation Qualification Tool on UNIX Systems

To run the SAS IQ on a successful SAS 9.2 installation, do the following:

From a console or a terminal window opened on the UNIX system, type the following command:

```
<SASROOT>/sastest/sasiq.sh -OutputPath <dir>
```

This invokes SASIQ and the following lines should be displayed:

```
SAS Installation Qualification Tool for the SAS System
Parsing input data files... Done.
```

After these two lines are displayed, the product-specific validation lines will be displayed. This will vary based on what products you have licensed and installed on your system. The following is a sample of this output and is not intended to represent what you will see depending on the products you have installed on your system:

```
Validating SAS/AF Software... Passed.
Validating SAS/ASSIST Software... Passed.
Validating Base SAS Software... Passed.
Validating SAS/CONNECT Software... Passed.
Validating SAS/ACCESS DB2 Software... Passed.
Validating SAS Data Quality Server Software... Passed.
Validating SAS/EIS Software... Passed.
Validating SAS/ETS Software... Passed.
Validating SAS/FSP Software... Passed.
```



```

Validating SAS/GENETICS Software... Passed.
Validating SAS/GIS Software... Passed.
Validating SAS/GRAPH Software... Passed.
Validating SAS/IML Software... Passed.
Validating SAS/ACCESS INFORMIX Software... Passed.
Validating SAS/INSIGHT Software... Passed.
Validating SAS Install Qualification Tool Software... Passed.
Validating SAS/IntrNet Software... Passed.
Validating SAS Integration Technologies Software... Passed.
Validating SAS/LAB Software... Passed.
Validating SAS OLAP Server Software... Passed.
Validating SAS/ACCESS Microsoft SQL Server Software... Passed.
Validating SAS/ACCESS MySQL Software... Passed.
Validating SAS/ACCESS Netezza Software... Passed.
Validating SAS/ACCESS ODBC Software... Passed.
Validating ODS Templates Software... Passed.
Validating SAS/OR Software... Passed.
Validating SAS/ACCESS Oracle Software... Passed.
Validating SAS/ACCESS PC File Formats Software... Passed.
Validating SAS/QC Software... Passed.
Validating SAS/ACCESS R/3 Software... Passed.
Validating SAS HPF Software... Passed.
Validating SAS/Secure 168-bit Software... Passed.
Validating SAS/SHARE Software... Passed.
Validating SAS/STAT Software... Passed.
Validating SAS/ACCESS SYBASE Software... Passed.
Validating SAS/ACCESS Teradata Software... Passed.
Validating SAS/TEST Software... Passed.
Validating SAS/TOOLKIT Software... Passed.
Validating Additional Fonts Software... Passed.
Generating output...
Generating reports...
SASCMD: <SASROOT>/9.2/sas -sysin <SASROOT>/sastest/sasiq.sas -autoexec
<OutputPath>/validate_YYYYMMDD.000/autoexec.sas
=====

Detailed PDF and HTML reports have been generated:
  <OutputPath>/validate_YYYYMMDD.000/sasiq.pdf
  <OutputPath>/validate_YYYYMMDD.000/sasiq.htm

39 SAS components validated: 12229 files validated, 0 failed.

```

## Reviewing the Results of the SAS Installation Qualification Tool

After you run the SAS IQ, a `validate_date.version` directory is created in the output directory you specified as `-OutputPath <dir>`. You can review the results of the SAS IQ by opening the `sasiq.htm` file in a Web browser or by opening the `sasiq.pdf` file in Adobe Reader.

The contents of the validate directory are described in the following table:

| File Name      | Description  |
|----------------|--|
| Data/          | Directory containing XML files for each component                                |
| html/          | Directory containing HTML files for each component                               |
| autoexec.sas   | SAS file used in producing PDF and HTML output from XML data                     |
| index.xml      | XML index file containing installation validation summary for installed software |
| index.xsl      | XML style sheet  |
| media.xsl      | XML style sheet  |
| sasiq.htm      | HTML file containing installation validation summary for installed software      |
| sasiq.pdf      | PDF file containing installation validation summary for installed software       |
| SASIQindex.map | Intermediate file used to output XML to PDF and HTML                             |
| SASIQprod.map  | Intermediate file used to output XML to PDF and HTML                             |
| sasiq.log      | SAS log file created from running the autoexec.sas program                       |
| sasiqtoc.htm   | File containing HTML output table of contents                                    |

The SAS IQ validates a SAS System installation by verifying that each installed file is correct. A report is generated detailing all file results. The status determination is made using the md5 algorithm to create a value for each file.

*Note: The SAS Installation Qualification Tool will only validate SAS Foundation components. Java components associated with SAS Solutions are not validated by the SAS Installation Qualification Tool.*

## VMS Itanium

### Running the SAS Installation Qualification Tool on VMS Itanium Systems

To run the SAS IQ on a successful SAS 9.2 installation, do the following:

From a console or a terminal window opened on the VMS Itanium system, type the following commands:

```
set default sas$root[.sastest]
@sasiq.com
```

This will result in the following output and you will enter an output directory to continue:

```
SAS Installation Qualification Tool for the SAS System
Please specify a directory where the output files should go
[.]:<OutputPath>
```

Then the following should be displayed:

```
Parsing input data files... Done.
```

After this line is displayed, the product-specific validation lines will be displayed. This will vary based on what products you have licensed and installed on your system. The following is a sample of this output and is not intended to represent what you will see depending on the products you have installed on your system:

```
Validating SAS/AF Software... Passed.
Validating SAS/ASSIST Software... Passed.
Validating Base SAS Software... Passed.
Validating SAS/CONNECT Software... Passed.
Validating SAS/EIS Software... Passed.
Validating SAS/ETS Software... Passed.
Validating SAS/FSP Software... Passed.
Validating SAS/GENETICS Software... Passed.
Validating SAS/GIS Software... Passed.
Validating SAS/GRAPH Software... Passed.
Validating SAS/IML Software... Passed.
Validating SAS/INSIGHT Software... Passed.
Validating SAS Install Qualification Tool Software... Passed.
Validating SAS/IntrNet Software... Passed.
Validating SAS/LAB Software... Passed.
Validating SAS OLAP Server Software... Passed.
Validating ODS Templates Software... Passed.
Validating SAS/OR Software... Passed.
Validating SAS/ACCESS Oracle Software... Passed.
Validating SAS/QC Software... Passed.
Validating SAS/SHARE Software... Passed.
Validating SAS/STAT Software... Passed.
Validating SAS/TEST Software... Passed.
Validating Additional Fonts Software... Passed.
Generating output...
Generating reports...
SASCMD: sas92 /sysin=sas$root:[.sastest]sasiq.sas
/autoexec=<OutputPath>[.validate_YYYYMMDD_000]autoexec.sas
/log=<OutputPath>[.validate_YYYYMMDD_000]sasiq.log
=====

Detailed PDF and HTML reports have been generated:
  <OutputPath>[.validate_YYYYMMDD_000]sasiq.pdf
  <OutputPath>[.validate_YYYYMMDD_000]sasiq.htm

24 SAS components validated: 6665 files validated, 0 failed.

Exiting from sasiq.
```

### Reviewing the Results of the SAS Installation Qualification Tool

After you run the SAS IQ, a `validate_date.version` directory is created in the output directory you specified as `<OutputPath>`. You can review the results of the SAS IQ by opening the `sasiq.htm` file in a Web browser or by opening the `sasiq.pdf` file in Adobe Reader.

The contents of the validate directory are described in the following table:

| File Name      | Description  |
|----------------|--|
| DATA.DIR       | Directory containing XML files for each component                                |
| HTML.DIR       | Directory containing HTML files for each component                               |
| autoexec.sas   | SAS file used in producing PDF and HTML output from XML data                     |
| index.xml      | XML index file containing installation validation summary for installed software |
| index.xsl      | XML style sheet  |
| media.xsl      | XML style sheet  |
| sasiq.htm      | HTML file containing installation validation summary for installed software      |
| sasiq.pdf      | PDF file containing installation validation summary for installed software       |
| SASIQindex.map | Intermediate file used to output XML to PDF and HTML                             |
| SASIQprod.map  | Intermediate file used to output XML to PDF and HTML                             |
| sasiq.log      | SAS log file created from running the autoexec.sas program                       |
| sasiqtoc.htm   | File containing HTML output table of contents                                    |

The SAS IQ validates a SAS System installation by verifying that each installed file is correct. A report is generated detailing all file results. The status determination is made using the md5 algorithm to create a value for each file.

*Note: The SAS Installation Qualification Tool will only validate SAS Foundation components. Java components associated with SAS Solutions are not validated by the SAS Installation Qualification Tool.*

# SAS Operational Qualification Tool (SAS OQ)

## Overview

The following tools are used to run operational tests of an installed image of SAS 9.2 at your site. These tools and testware are shipped on SAS 9.2 media.

- SAS Operational Qualification Tool (SAS OQ)—the program that invokes the operational tests. It accepts various command-line arguments (documented below). It runs the tests, validates the output from the test, and produces the report.
- IVFND—the “Filter 'N' Diff” program. The test output and benchmarks need to be filtered before they are compared in order to remove spurious differences such as date/time values and pathnames. IVFND runs the appropriate filters and difference tool to validate an output file.
- PED—a Perl version of the UNIX `sed` command that provides the filtering capability to IVFND.
- IVDIFF (`sasdiff.exe`)—a difference tool that has some special options to control the difference.

## Windows

### Directory Structure

SAS Operational Qualification Tool - `<SASROOT>\sastest\sasq.exe`

SAS Operational Qualification Tool support files - `<SASROOT>\sastest\`

Component content - `<SASROOT>\sastest<component>`

### Running the Operational Qualification Tool

To run SAS OQ on a successful SAS 9.2 installation, do the following:

- Open a command line in DOS and type the following command:

```
c:\><SASROOT>\sastest\sasq.exe -tables *:<component> -outdir <dir>
```

In this command, `<SASROOT>` is the location where the SAS System was installed, `<component>` is the installed component in the `sastest` directory for which you want to run tests, and `<dir>` is a directory to store the results.

### Output

The SAS OQ generates XML data files to hold the raw results of the validation process. The SAS System is used to translate the XML data into PDF and HTML reports. The PDF and HTML reports are identical. The PDF can be used to create hardcopies of the resulting validation. The HTML is easier to browse.

In all cases, the output report is organized first at the SAS System component level, and then by each component's files.

## Outdir Directory Structure

The following table describes the structure of the directory created with the `-Outdir` option:

| File Name      | Description  |
|----------------|--|
| <Component>    | When tests are run, a directory is created for each to store files created by the tests. |
| Data           | Directory containing the raw validation data for each component in XML format            |
| html           | Directory containing HTML files for each component                                       |
| autoexec.sas   | SAS file used in producing PDF and HTML output from XML data                             |
| index.xml      | XML index file for all component summary   |
| index.xsl      | XML style sheet  |
| sasoq.htm      | HTML file containing index to summary and component information                          |
| sasoq.log      | SAS log file created from running the autoexec.sas program                               |
| sasoq.pdf      | PDF file containing all summary and component information                                |
| SASOQindex.map | Intermediate file used to output XML to PDF and HTML                                     |
| SASOQtest.map  | Intermediate file used to output XML to PDF and HTML                                     |
| sasoqtoc.htm   | HTML file used to display component information  |
| Test.xsl       | XML style sheet  |

## Reviewing the Results of the SAS Operational Qualification Tool

After you run the SAS OQ, a result directory is created in the output directory you specified with `-Outdir <dir>`, with the format `ftt_<YYYYMMDD>.000`. The trailing `.000` would be incremented for tool invocations on the same day. You can review the results of the SAS OQ by opening the `sasoq.htm` file in a Web browser or by opening the `sasoq.pdf` file in Adobe Reader.

## UNIX

### Directory Structure

SAS Operational Qualification Tool - `<SASROOT>/sastest/sasoq.sh`

SAS Operational Qualification Tool support files - `<SASROOT>/sastest/`

Component content - `<SASROOT>/sastest/<component>`

## Running the SAS Operational Qualification Tool on UNIX Systems

To run the SAS OQ on a successful SAS 9.2 installation, do the following:

From a console or a terminal window opened on the UNIX system, type the following command:

```
<SASROOT>/sastest/sasoq.sh -tables *:<component> -outdir <dir>
```

In this command, *<SASROOT>* is the location where the SAS System was installed, *<component>* is the installed component in the *sastest* directory for which you want to run tests, and *<dir>* is a directory to store the results.

### Output

The SAS OQ generates XML data files to hold the raw results of the validation process. The SAS System is used to translate the XML data into PDF and HTML reports. The PDF and HTML reports are identical. The PDF can be used to create hardcopies of the resulting validation. The HTML is easier to browse.

In all cases, the output report is organized first at the SAS System component level, and then by each component's files.

### Outdir Directory Structure

The following table describes the structure of the directory created with the `-Outdir` option:

| File Name      | Description  |
|----------------|--|
| <Component>    | When tests are run, a directory is created for each to store files created by the tests. |
| Data           | Directory containing the raw validation data for each component in XML format            |
| html           | Directory containing HTML files for each component                                       |
| autoexec.sas   | SAS file used in producing PDF and HTML output from XML data                             |
| index.xml      | XML index file for all component summary   |
| index.xsl      | XML style sheet  |
| sasoq.htm      | HTML file containing index to summary and component information                          |
| sasoq.log      | SAS log file created from running the autoexec.sas program                               |
| sasoq.pdf      | PDF file containing all summary and component information                                |
| SASOQindex.map | Intermediate file used to output XML to PDF and HTML                                     |
| SASOQtest.map  | Intermediate file used to output XML to PDF and HTML                                     |
| sasoqtoc.htm   | HTML file used to display component information  |
| Test.xsl       | XML style sheet  |

## Reviewing the Results of the SAS Operational Qualification Tool

After you run the SAS OQ, a result directory is created in the output directory you specified with `-Outdir <dir>`, with the format `ftt_<YYYYMMDD>.000`. The trailing `.000` would be incremented for tool invocations on the same day. You can review the results of the SAS OQ by opening the `sasoq.htm` file in a Web browser or by opening the `sasoq.pdf` file in Adobe Reader.

## VMS Itanium

### Directory Structure

```
SAS Operational Qualification Tool - <SASROOT>:[sastest]sasoq.com
SAS Operational Qualification Tool and support files - <SASROOT>:[sastest] Component content
<SASROOT>:[sastest.<component>]
```

### Running the SAS Operational Qualification Tool on VMS Itanium Systems

To run the SAS OQ on a successful SAS 9.2 installation, do the following:

From a console or a terminal window opened on the VMS Itanium system, type the following commands:

```
<SASROOT>:[sastest]sasoq.com -tables *:<component> -outdir <directory for
results>
```

In this command, `<SASROOT>` is the location where the SAS System was installed, `<component>` is the installed component in the `sastest` directory for which you want to run tests, and `<dir>` is a directory to store the results.

### Output

The SAS OQ generates XML data files to hold the raw results of the validation process. The SAS System is used to translate the XML data into PDF and HTML reports. The PDF and HTML reports are identical. The PDF can be used to create hardcopies of the resulting validation. The HTML is easier to browse.

In all cases, the output report is organized first at the SAS System component level, and then by each component's files.

### Outdir Directory Structure

The following table describes the structure of the directory created with the `-Outdir` option:



| File Name      | Description  |
|----------------|--|
| <Component>    | When tests are run, a directory is created for each to store files created by the tests. |
| Data           | Directory containing the raw validation data for each component in XML format            |
| html           | Directory containing HTML files for each component                                       |
| autoexec.sas   | SAS file used in producing PDF and HTML output from XML data                             |
| index.xml      | XML index file for all component summary   |
| index.xsl      | XML style sheet  |
| sasoq.htm      | HTML file containing index to summary and component information                          |
| sasoq.log      | SAS log file created from running the autoexec.sas program                               |
| sasoq.pdf      | PDF file containing all summary and component information                                |
| SASOQindex.map | Intermediate file used to output XML to PDF and HTML                                     |
| SASOQtest.map  | Intermediate file used to output XML to PDF and HTML                                     |
| sasoqtoc.htm   | HTML file used to display component information  |
| Test.xsl       | XML style sheet  |

## Reviewing the Results of the SAS Operational Qualification Tool

After you run the SAS OQ, a result directory is created in the output directory you specified with `-outdir <dir>`, with the format `ftt_<YYYYMMDD>.000`. The trailing `.000` would be incremented for tool invocations on the same day. You can review the results of the SAS OQ by opening the `sasoq.htm` file in a Web browser or by opening the `sasoq.pdf` file in Adobe Reader.

## Additional Examples of Invocation

- To run multiple components:

```
<SASROOT>\sastest\sasoq.exe -tables *:base *:ets *:or *:qc *:stat
-outdir c:\public\mydata
```

This command line, for Windows, will run the tests in the `base`, `ets`, `or`, `qc`, and `stat` directories provided by SAS and put the data file in the directory `c:\public\mydata`. If the directory does not exist, it will be created.

- To run all tests:

```
<SASROOT>\sastest\sasoq.exe -tables **: * -outdir <directory for results>
```

This command line, for Windows, will run the tests provided by SAS and put the data file in the directory specified.

- To specify the testware location for customer tests:

```
<SASROOT>\sastest\sasq.exe -testware <path to testware> -tables
*:<subdirectory> ...
```

This command line would only run the tests in the -testware path.

- The tool can provide more detail in the screen output by using the verbose option:

```
<SASROOT>/sastest/sasq.sh -tables *:ets -outdir /users/guest/mydata -
verbose
```

This UNIX command line will run the ETS tests, put the data files in a local mydata directory, and provide the sas command line used to run the test, any processing statements, and the results of the test, such as:

```
Running test ets:tstets:tstari01...
Running: sas -sysin "/usr/local/sas/sastest/ets/tstari01.sas" -autoexec
"/usr/local/sas/sastest/base/assert.sas" -nodate -nostimer -ls 78 -ps 60
-noovp -nosyntaxcheck
sasq: The command returned the proper value (0).
Processing Results:
```

- The quiet option will suppress all non-error output.

```
<SASROOT>/sastest/sasq.sh -tables *:ets -outdir /users/guest/mydata -quiet
```

This UNIX command line will produce no output from the test execution until all tests are run. The tool will continue to produce the message directing the user to the appropriate path for the data files.

## Automation

The SAS OQ can be run in an automated manner to allow for scheduled operational tests on a production or test SAS 9.2 image.

### Windows

To establish the SAS OQ as a scheduled task on SAS 9.2-supported Windows operating systems, use the following steps:

1. Create a script similar to the example below.
2. To create a scheduled task:
  - Find the Scheduled Task interface. On newer Windows systems this is found under Control Panel
  - Use the wizard to create the scheduled task
  - In the Task panel, execute the script in the example using the wscript interface:  
wscript.exe "c:\sasiq\runoq.vbs"
  - In the Schedule panel select the appropriate frequency
  - Press OK to create the Scheduled Task
3. Review the tool output after each invocation.

## Windows Example

```
'* runoq.vbs - run the SAS Operational Qualification Tool using all the
   component tests
'* First edition

'* run this script only on the local machine
strComputer = "."

oqpath = "\sasq\mydata"

'* create the string to use for executing the tool
initoq = "c:\program files\sas\sas 9.2\sastest\sasoq.exe -tables *:base *:dmine
*:ets *:graph *:hpf *:iml *:insight *:irp *:lab *:or *:qc *:stat -outdir " &
oqpath

'* execute the tool
Set WshShell = WScript.CreateObject("WScript.Shell")
Set oExec = WshShell.Exec(initoq)
```

## UNIX

To establish the SAS OQ as a crontab task on SAS 9.2 supported UNIX operating systems, use the following steps:

1. Create a script similar to the UNIX example below.
2. To execute the UNIX example enter the following into a crontab entry:

```
# run SAS 9.2 SAS Operational Qualification Tool process
# at 09:05am Weekdays
05 09 * * 1,2,3,4,5 "/users/guest/bin/runoq > /users/guest/mydata/runoq.log"
```

3. Review the tool output after each invocation

## UNIX Example

```
#!/bin/sh
#runoq - execute the SAS Operational Qualification Tool

/usr/local/sas/sastest/sasoq.sh -tables *:base *:dmine *:ets *:graph *:hpf
*:iml *:insight *:irp *:lab *:or *:qc *:stat -outdir /users/guest/mydata
```

## SAS Operational Qualification Tool Table Language

The primary input files for the SAS OQ are test tables (stored in files with a .tab extension). These files define which tests to run, how to run the tests, and how to determine if the tests worked correctly.

## Table Language General Rules

The following are basic rules for test table files:

- The SAS OQ generally ignores white space (spaces or line breaks).
- Comments are delimited by `/*` and `*/`. They can span multiple lines and appear anywhere in a line. These comments work the same as C comments.
- Statement arguments fall into one of the following groups:
  - A **list** is a group of items in any order separated by commas. An example of a list is the set of options passed to the program being tested. The order of such options is irrelevant to the operation of the SAS OQ. (The program being tested might require them to be in a particular order, but the SAS OQ does not.)
  - A **structure** is a group of items in a specific order separated by colons. An example of a structure is a file specification. The elements must be in a particular order, so that the SAS OQ can tell which part is the file name, which part is the component name, etc.
- Statements are processed in the order in which they occur in the test block, so the statement to run the test must be before the statements to process the results.
- Tables are made up of comments, `&set` commands, `&unset` commands, and test blocks. Each test block starts with the block identifier, `&test`, and contains a set of commands—a single `&run` followed by `&process` and/or `&rc`. Within the commands you can optionally use the macros `&infile`, `&outfile` and `&resfile`.

## Block Identifier

The only valid block identifier is `&test`.

### **`&test`**

Usage:

```
&test testname { test_stmt_list }
```

Arguments:

- *testname* is the name of this test.
- *test\_stmt\_list* is the list of statements that make up the test.

Description:

The `&test` statement defines a test block. There must be one test block for each test in a test table file.

Example:

See the complete example below.

## Command

The commands are `&process`, `&rc`, `&run`, `&set` and `&unset`,

### **&process**

Usage:

```
&process ( verb : ( arguments ) [ , verb : ( arguments ) ... ] )
```

Arguments:

- o *verb* is the name of the command to run. In the current version, the only verb supported is `ivfnd`.
- o *arguments* are the arguments to the command.

Description:

The `&process` statement processes the results of the test run. It normally runs the IVFND program, which performs any filtering of the output and benchmark files and then compares the benchmark files. The return code from IVFND indicates the success or failure of the benchmark comparison. A return code of zero is success.

The use of the `&process` command is optional. However, if you choose to use the `&process` command you may not use it more than once per test block. If you do not use `&process` then you **must** use the `&rc` command in your test block.

Example:

```
&process( ivfnd : (-filter &infile(prs:logfilter) -result &resfile(log) ) ,
          ivfnd : (-filter &infile(prs:lstfilter) -result &resfile(lst) ) )
```

### **&rc**

Usage:

```
&rc ( value )
```

Arguments:

*value* is the expected value of the return code for this test.

Description:

The `&rc` statement tests the return code from the command specified in the `&run` statement. If the return code does not match the specified value, the test is marked as a failing test.

The use of the `&rc` command is optional. However, if you choose to use the `&rc` command you may not use it more than once per test block. If you do not use `&rc` then you **must** use the `&process` command in your test block.

Example:

```
&rc( 0 )
```

## **&run**

Usage:

```
&run ( verb : (arguments) )
```

Arguments:

- o *verb* is the name of the program to run. In the initial version, the only verb allowed is "sas".
- o *arguments* is the list of options and arguments to the program being run.

Description:

The `&run` statement runs the actual test command. There must be **one** `&run` command in each test block. The `&run` must be followed by `&process` and/or `&rc`.

Example:

```
&run( sas : (-sysin &infile() -ls 78) )
```

This command runs the SAS system. The `&infile` statement (see below) passes an input file with the same base name as the test name. The other options are passed through unmodified.

## **&set**

Usage:

```
&set ( name : value )
```

Arguments:

- o *name* is the name of the variable you are defining consisting only of alphanumeric characters.
- o *value* is the any sequence of tokens. If the sequence contains parentheses, they must be balanced or the unbalanced parenthesis needs to be escaped (see example).

**Description:**

This command allows you define variables and assign values to them. The names and values you define using `&set` are global and remain in effect for the remainder of the current.

You may change the value of a variable by assigning a new value later using another `&set` statement.

Once you define a variable, you can use it anywhere in the table (as long as the `&set` statement precedes its use). To use the variable you created, use `@name` to insert the value of the variable. If you use a variable that has not been defined, an empty string is used for the value.

All parentheses inside the variable value must be balanced. If you require unbalanced parentheses, use the `'\'` escape character to escape the parenthesis.

The `&set` statement may not be used inside of a block.

The `testname` and `tablename` variables are predefined for you and are set to the name of the test (or table) at the time the variable is expanded.

If the `@` sign is needed in the value of the variable but should not be treated as a variable expansion, then escape the `@` sign. See the examples below for a sample use of this.

**Example:**

```
&set( sasopts : -ls 78 -ps 60 )
```

This command sets the `sasopts` variable to “-ls 78 -ps 60”.

```
&set( infile : -sysin &infile( sas : @testname : base) )
```

This command sets the `infile` variable to “-sysin &infile( sas : @testname : base)”.

```
&set( infile1 : -sysin &infile\( sas : )
&set( infile2 : : base\) )
&set( run : sas : ( @infile1 @testname @infile2 \@noexpand) )
```

This command sets the `infile1` variable to `-sysin &infile( sas :, infile2 to : base)` and `run` to `sas : ( @infile1 @testname @infile2 @noexpand)`. Assuming a test name of `test1`, the `run` variable will expand to `sas : ( -sysin &infile( sas : test1 : base)`.

***&unset*****Usage:**

```
&unset ( name )
```

**Arguments:**

- o *name* is the name of a variable that may, or may not, have been set using the `&set` command. Note that the `@` is **not** included when specifying the variable name, as that would cause the variable to be replaced with its value before the `unset` processing occurs.

**Description:**

This command allows you to guarantee that a variable is no longer set and if used, will expand to nothing.

**Example:**

```
&unset( testopts )
```

This command clears the `testopts` variable even if it had not previously been defined.

**Macro**

The macros are `&infile`, `&outfile`, and `&resfile`.

***&infile*****Usage:**

```
&infile( ext [: name [: component]]) or
&infile([ext]: name [: component]) or
&infile([ ext ] : [name] : component)
```

**Arguments:**

- o *ext* is the file extension. If it is omitted, the default is "sas".
- o *name* is the file name. If it is omitted, the name of the test is used as the file name.
- o *component* is the component area where the file is installed. If it is omitted, the component area where the table file was found is used.

**Description:**

The `&infile` statement defines an input file. It is normally used as part of an option to a command in the `&run` statement. Note that it is allowed to omit all of the arguments to `&infile`.

**Example:**

```
&infile( sas : foobar : base )
    Specifies the file named foobar.sas from the base testware area.
```

```
&infile()
    Specifies the file testname.sas (where testname is the name of the current test) from the testware area of the component where the .tab file is located.
```



**&outfile**

Usage:

```
&outfile( ext [: name ] )
```

Arguments:

- o *ext* is the file extension. There is no default for the extension. It must be specified.
- o *name* is the file name. If it is omitted, the name of the test is used as the file name.

Description:

The `&outfile` statement defines an output file. It is normally used as part of an option to a command in the `&run` statement.

```
&outfile( lst : foo )
```

Specifies the file named `foo.lst`.

```
&outfile(log)
```

Specifies the file `testname.log`.

**&resfile**

Usage:

```
&resfile( ext [: name [: component]] ) or  
&resfile(ext: name [: component])
```

Arguments:

- o *ext* is the file extension. There is no default for the extension. It must be specified.
- o *name* is the file name. If it is omitted, the name of the test is used as the file name.
- o *component* is the component area where the corresponding benchmark file is installed. If it is omitted, the component area where the table file was found is used.

Description:

The `&resfile` statement defines a test result file and a corresponding benchmark. It is normally used as part of an option to the `IVFND` command in the `&process` statement. It actually specifies a pair of files. One file is the output file from the test. It is assumed to be written to the current working directory where the test was run. The other file is the shipped benchmark file. It is in the specified location. The base names and extensions of the files must be the same.

Example:

```
&resfile( log : foobar : test : base )
```

Specifies the file named `foobar.log` from the base testware area.

```
&resfile( lst )
```

Specifies the file `testname.lst` (where `testname` is the name of the current test) from the testware area of the component where the `.tab` file is located.

## Examples

```
&set ( sasopts : -sysin &infile -log &outfile( log : barfoo ) )
&test foobar
{
  &run( sas : (@sasopts) )
  &process( ivfnd : ( -filter &infile( prs : logfilter ) -result &resfile( log, barfoo ) ),
            ivfnd : ( -result &resfile( lst ) ) )
  &rc( 0 )
}
```

This test case runs the SAS system with an input file named `foobar.sas`. The file is in the same component area as the test table file. The test is expected to produce two output files, `foofoo.log` and `foobar.lst`. The log file and the corresponding benchmark are filtered using `ped` with the filter script named `logfilter.prs` (also in the testware area of the current component) before being compared. The `lst` file is not filtered before being compared. The SAS command is expected to exit with a return code of zero.

## SAS Operational Qualification Tool Command Line

The SAS OQ command line arguments are defined as follows:

```
-tables table_spec [table_spec ...]
```

The `-tables` option specifies which test tables to run. A `table_spec` has the form `name:component`. This is very similar to the specifications in the `&infile` and `&resfile` statements, except that the extension is always assumed to be `.tab`. At least one `table_spec` is required.

```
-help
```

The `-help` option requests the tool to print its usage.

```
-verbose | -quiet
```

These two arguments dictate the amount of STDOUT output the SAS OQ writes during execution. The `-verbose` option tells the SAS OQ to print extra output to STDOUT when executing. This is useful when setting up new tests. The `-quiet` option tells the SAS OQ to only report errors. This is useful when scripting SAS OQ commands.

```
-sasroot <path>
```

This argument tells the SAS OQ where to locate The SAS System installation. By default, the tool will attempt to locate the SASROOT path itself. If it is unable to, for some reason, this argument can be used to explicitly set that path. This can also be used to test different SAS System installations on the same machine.

```
-testware <path>
```

This argument specifies the location of the testware. By default SASROOT/sastest is used.

```
-outdir <path>
```

This argument specifies the location where all output files should be written. The default is to use the current directory where the tool execution took place.

## **Perl Stream Editor Documentation**

Perl Stream Editor (`ped`) is designed to be an alternative to using `sed` (the original UNIX SysV Stream Editor). Although based on `sed`, the filter expressions that `ped` recognizes are different than those recognized by `sed`. Some filter expressions that are valid for `sed` are still valid for `ped`; but some are not! `ped` is not intended to translate existing `sed` filter files.

The biggest benefit to using `ped` is that you now have the power of Perl Regular Expressions at your disposal rather than the useful, but limited set of `sed` regular expressions. Another benefit to using `ped` is that it allows a syntax for inclusion of other filter files, similar to the inclusion process in many other languages.

`ped` also defines an extension to the Perl-provided set of regular expression commands. This command is called Substringing Substitution. It allows you to write a regular expression that identifies a substring of the input stream and then apply a Substitute regular expression command on only that substring. This allows you to be very specific about what you are filtering and it makes the actual search pattern regular expressions much, much less complicated. Briefly, the command syntax is: `S/<substring-RE>/<substring-Options>/<search-RE>/<replacement-RE>/<substitute-Options>`

The `substring-Options` are the same set of options available to the Perl-provided Substitute command. The Substringing Substitution command is implemented as a pair of Substitute commands.

### **What Can I Do with Perl REs That I Can't Do with sed REs?**

Perl REs have the ability to match patterns over any number of newline characters. It has the ability to match even deeply nested patterns, this is very useful for markup languages like HTML, XML, and SGML. It has a much richer set of meta-sequences (white-space, word-characters, octal- and hex- characters, digits, etc.). You can capture matched sub-strings in memory buffers and substitute them back into the output string. You can do "look-aheads" and "look-behinds". You can easily express quantifiers and define new character classes. It has an extended syntax that allows you to add human-readable comments directly into complicated RE's thus allowing other folks to understand them and maintain them.

Also, `ped` has no limit on the number of filter expressions that it can accept. `sed` limits you to 100 expressions.

## Filter File Syntax

- Lines that begin with a pound-sign (#) are ignored. This is the ped comment, it is the same for sed.
- Lines that contain nothing but white space are ignored.
- Lines that begin with %include are interpreted as include commands and not as REs. ped looks for a token following the %include ; it interprets that token as a file path. It attempts to read that file and process its contents as additional REs.

## ped Command Line Syntax

```
ped [-e <'|"><filter RE><'|">] [-f <filter file path>] [-n [<file path 1> [<file path n>]]
ped <'|"><filter RE><'|"> [<file path 1> [<file path n>]]
```

## Substringing Substitute("S" command) Syntax

```
S/<substring-RE>/<substring-Options>/<search-RE>/<replacement-RE>/<substitute-Options>
```

Where substring-Options are defined to be the same set of options available to the Perl regular expression Substitute("s") command. The substring-RE and the search-RE are required. The substring-Options, replacement-RE, and substitute-Options are not required.

The "S"-command works by applying the substring-RE to the input stream; when a match is found, a common "s" command is applied to only the matched portion at portion of the input stream.

## Ped Filter File Examples

```
mylogfilter.prs:
#-----
# Purpose: To filter out the common, non-component-specific stuff
#         from SAS .log files.
# Filter: ped
#-----
# Get rid of the page headers
s/^\f*\d{1,6}\s+The SAS System\s+$/gmo
# Get rid of the copyright notice.
s/^\s*note:\s*copyright\ (c\).*$/imo
# Get rid of the "proprietary/licensed" note.
s/^\s*note:\s*sas\s*(r)\s*proprietary.+?\n\s*licensed.+?$/imos
# Get rid of the Institute address note.
s/^\s*note:\s*sas\s+institute\s+inc\.\.\s+sas\s+campus\s+drive,\s+cary,\s+nc.+?$/imo
# Get rid of the "SAS used" message.
s/^\s*note:\s+the\s+sas\s+system\s+used:\s*\n\s+real\s+time\s+d+\.\d\d\s+seconds\s*\n\s+cpu\s+time\s+d+\.\d\d\s+seconds\s+$/imos
```

## Issues

Since ped reads in the entire input stream into memory before it begins to operate on it, the size of the file that ped can filter is limited to the amount of available memory. Fortunately, memory is large today and no file that is too large to filter has yet been reported. sed only reads a few lines at a time, so it can process files of virtually any size.

## IVFND Command Line

IVFND (Install Validation Filter 'N' Diff) performs filtering and differencing operations for the SAS OQ. It is normally invoked from the `&process` statements in the test tables.

Arguments:

`-filter ext:name:component`

The `-filter` option specifies the name and location of the filter input file. If this option is present, IVFND executes PED using the specified filter script to filter both the test output file and the corresponding benchmark file before comparing them. Note that all of the fields must be present.

`-result ext:name:component`

The `-result` option specifies the name and location of the result file and of the corresponding benchmark file. The result and benchmark are filtered (if the `-filter` option is present) and then compared using IVDIFF.

Return codes:

Zero - All input files were found, filtering (if any) was successful, and the filtered files compared cleanly.

Non-zero - Something went wrong. Messages are written to `stdout/stderr`. If the files did not compare cleanly, difference files are left in the current working directory so the user can see exactly what failed. If any of the IVFND commands returns a non-zero return code, the test is marked as failing.







THE  
POWER  
TO KNOW.

### **support.sas.com**

SAS is the world leader in providing software and services that enable customers to transform data from all areas of their business into intelligence. SAS solutions help organizations make better, more informed decisions and maximize customer, supplier, and organizational relationships. For more than 30 years, SAS has been giving customers around the world The Power to Know®. Visit us at **www.sas.com**.