Getting started with the SAS I: installation, setup, and odfingest

Giuseppe Vacanti (XMM-Newton SOC, gvacanti@xmm.vilspa.esa.es)

November 22, 2001

Abstract

A quick look at how to install and use the SAS. Then, the first step before an observation can be analyzed: *odfingest*.

1 How to install the SAS

The SAS is distributed in binary form for a number of platforms. The list of supported platforms, and the most up-to-date instructions on how to install the SAS can be found on the SAS web pages (http://xmm.vilspa.esa.es/sas/).

What is important to remark here, is that we try to make the binary installation as trouble free as possible, by providing the user with all the libraries required to run the pre-compiled distributions.

2 SAS set up

Install the SAS in `/software/sas/`. Before you can start using the SAS you must:

- **Bourne shells**:
  - `SAS_DIR=/software/sas/`; export `SAS_DIR`
  - `SAS_PATH=$SAS_DIR`; export `SAS_PATH`
  - `. $SAS_DIR/sas-setup.sh`
- **C shells**:
  - `setenv SAS_DIR /software/sas/`
  - `setenv SAS_PATH $SAS_DIR`
  - `source $SAS_DIR/sas-setup.csh`

At this point you are ready to run any SAS task. Note that if you wish to make use of tasks like *xmmselect* and *calview*, you should make sure that `ds9` and `grace` are available in your `PATH`. Refer to the SAS web pages (http://xmm.vilspa.esa.es/sas/) to learn where to find them.

3 The graphical user interface

If you are new to the SAS, you should consider starting with the task *sas*. Before you do that, it is useful to set two other environment variables: `SAS_ODF`, and `SAS_CCF`. The former should point to the summary file created with *odfingest*, the latter should point to a CCF index file. (See section 5 and *gvacanti_2* for details on *odfingest* and the CCF index file.)

4 Useful things to know

4.1 What version of the SAS am I running?

Use *sasversion*. For example:

```
gv> sasversion
SAS release: xmmsas_20010917_1110-5.2.0
Compiled on: Mon Sep 17 11:22:53 MET DST 2001
Compiled by: saslib@xmmserver
Platform   : solaris2.6
```

Under `SAS release` you see the SAS build identifier (`xmmsas_20010917_1110`) and the SAS version identifier (`5.2.0`).

4.2 What does `task -v` show?

For instance: *emproc -v*

```
emproc (emproc-2.4.2) [xmmsas_20010917_1110-5.2.0]
```

These are:

- `emproc`: task name
- `emproc-2.4.2`: SAS package name
- `xmmsas_20010917_1110-5.2.0`: release identifier

Some packages contain more than one task. For instance:

- `tabcalc -v`

  ```
  tabcalc (dscalc-1.0.4) [xmmsas_20010917_1110-5.2.0]
  ```

- `phasecalc -v`

  ```
  phasecalc (dscalc-1.0.4) [xmmsas_20010917_1110-5.2.0]
  ```

4.3 What does `task -h` show?

All SAS tasks share a common *main* program, where command line options are processed. To see all of the available command line options invoke a task with the `-h` switch.

```
gv> odfingest -h
XMM-Newton Science Analysis System, task 'odfingest (odfingest-3.2.1) [xmmsas_20010917_1110-5.2.0]'

Usage:
  odfingest [options] [--]

Options:
  -a <dir1>[:<dir2>...] | --ccfpath <dir1>[:<dir2>...]
  -c | --noclobber
  -d | --dialog
  -f <f1> [<f2> ...] | --ccffiles <f1> [<f2> ...]
  -h | --help
  -i <cifname> | --ccf <cifname>
  -m | --manpage
  -o <odfname> | --odf <odfname>
  -p | --param
  -t | --trace
  -V <level> | --verbosity <level>
  -v | --version
  -w [code|n] | --warning [code|n]
  --<param>=<value>
  --

Parameters:
(list of parameters omitted)
```

For a detailed description of the options supported by each SAS task you should consult the documentation for the package *taskmain*. Note that some of these options are available only if the task makes use of some of the SAS infrastructure libraries. For instance, the option `--ccf` is available only in tasks that make use of the Calibration Access Layer (*cal*).

4.4 I need help!

There are several ways to reach the SAS documentation.

- `sashelp`
  This brings you to the SAS documentation entry page.
- `sashelp --doc=emproc`
  This brings you directly to the documentation of *emproc*.
- `emproc --manpage`
  This brings you directly to the documentation of *emproc*.
- `netscape $SAS_DIR/README.html`
  Should all of the above fail, then simply point your browser to the entry page of the SAS documentation, as shown above.

4.5 Lost in a very long command line ...

Many SAS tasks have many parameters, and this can make for a very long command line. The output of `task -h` will show you the complete list of available parameters, but still this can be hard to read[1].

Are you working on the command line but wish you had a *mini graphical user interface*? Try: `task --dialog` and you may be pleasantly surprised.

4.6 Verbosity

The amount of diagnostic output generated by SAS tasks (the so-called *verbosity*) can be modulated in two manners.

On the command line, you can use the `-V nn` switch. (See the documentation of *taskmain*.) The integer `nn` should be between 0 and 10. The higher the verbosity level, the more messages from the SAS infrastructure libraries you see. In practice you never want to run a task at a verbosity higher than 6, or you'll be swamped in low level messages that you are not interested in.

In a global manner, you can set the environment variable `SAS_VERBOSITY` to be an integer as above. This affects all tasks, and can be overridden on the command line, as explained above.

In the sas GUI (*sas*) you can set the verbosity via a menu.

Note: The SAS expands a command line to contain the complete lists of parameters passed to the task, whether they were explicitly set on the command line (or in the GUI) or not. The expanded command line is echoed back only if the verbosity level is 3 or higher.

4.7 XPROC?

If you ever looked at the header of one of the datasets created by the SAS, you may have noticed the presence of several `XPROC` attributes (keywords). What are they?

Each time that a dataset is created or modified by a SAS task, the command line that was used in the process is recorded in one `XPROC` entry.

At times it is useful to be able extract these records into a file, perhaps to re-execute them one at a time. This would be the case with the output of the procs. (See *gvacanti_3*.)

In order to do this, you can use the task *playback*. For example:

```
gv> playback --sets=eventlist.ds
playback:- # ---------------------------------------------------------------
playback:- # SAS commands from eventlist.ds
playback:- # ---------------------------------------------------------------
playback:- evselect attributestocopy='' blockstocopy='' cleandss=no etc.
```

[1] From the next public version of the SAS, the output of `task -h` will be improved, so that the parameters are no longer listed alphabetically.

4.8 Links

A few pointers to more detailed sources of information:

- *taskmain*: detailed description of the task command line, and the main environment variables used to affect the behavior of the SAS.
- *param*: the SAS parameter library, describing the meaning of the parameter types used in the SAS.
- *sas*: the SAS graphical user interface.

5 odfingest

The Observation Data File (ODF) distributed to the guest observer contains a summary file (SF) with the familiar name `rrrr_oooooooooo_SCX00000SUM.ASC`, where `rrrr` is the revolution number and `oooooooooo` is the observation identifier.

The SF does not contain all the information required to process the ODF with the SAS. The task *odfingest* extends the SF with data extracted from the instrument housekeeping data files and the calibration database. It creates a new summary file: the SAS ODF Summary File (SOSF). The name of the SOSF is the same as the name of the original summary file but for its ending: `rrrr_oooooooooo_SCX00000SUM.SAS`

By default *odfingest* writes the SOSF to the current directory. This can be changed via the parameter `outdir`.

If the output directory and the ODF directory are different it is useful to instruct *odfingest* to write in the SOSF the path to the ODF directory. This behavior is controlled via the parameter `writepath`, and is enabled by default.

5.1 The first five steps

Normally you don't need to re-process an ODF. You can use the calibrated event lists available in the pipeline products. (See *josborne_1*.)

Should you decide to re-process your data, these are the steps you have to make first (see also *gvacanti_3*):

1. Point the environment variable `SAS_ODF` to the directory where the ODF is stored. In the following assume that it is revolution `0123` and observation `0123456789`.
2. Get the calibration index file (CIF) and the current calibration file (CCF) constituents corresponding to your observation from the CCF web site (http://xmm.vilspa.esa.es/ccf/ follow the link to `cifbuild`). See also *gvacanti_2* for more details on *cifbuild*.
3. Store the CCF constituents in `/xmm/ccf/`, store the CIF in ccf.cif. Set the environment variable `SAS_CCFPATH` to be `/xmm/ccf`, and `SAS_CCF` to be `ccf.cif`.
4. Run: `odfingest`

   This will create a SOSF (`0123_0123456789_SCX00000SUM.SAS`) in the current directory.

5. Set the environment variable SAS_ODF to be 0123_0123456789_SCX00000SUM.SAS.

   Now you are ready.

   Note: If you want to be able to move the SOSF and still be able to access the data (that must not have changed location), make sure that you start by setting SAS_DIR

5.2 Links

- *cal*: See the documentation of the Calibration Access Layer for the meaning of SAS_CCF and SAS_CCFPATH.
- *oal*: See the documentation of the ODF Access Layer for the meaning of SAS_ODF.

References