

SAS[®] Packages Framework an easy code sharing medium for SAS

Bartosz Jabłoński
Warsaw IT Days - March 31st, 2023



www.polsug.com
#PolSUG

- Comprehensive T_EX Archive Network
<https://ctan.org/pkg/>
- Comprehensive R Archive Network
<https://cran.r-project.org>
- Python Package Index
<https://pypi.org/>
- Why not in SAS?



SAS package¹

A **SAS package** is an automatically generated, single, stand alone zip file containing organised and ordered code structures, created by the developer and extended with additional automatically generated "driving" files (i.e. description, metadata, load, unload, and help files, etc.)

The purpose of a package is to be a simple, and easy to access, code sharing medium, which allows: on the one hand, to separate the code complex dependencies created by the developer from the user experience with the final product and, on the other hand, reduce developers and users unnecessary frustration related to a remote deployment process.

¹ The idea presented here should not be confused with other occurrences of "package" concept which can be found in the SAS ecosystem e.g., Proc DS2 packages, SAS/IML packages, SAS ODS packages, SAS Integration Technologies Publishing Framework packages, or even an Enterprise Guide *.egp file.

SAS Packages Framework

The **SAS Packages Framework** (SPF) is a "pack" of macros, which allows to *use* and to *develop* SAS packages.

- %listPackages(),
- %installPackage(),
- %verifyPackage(),
- %helpPackage(),
- %previewPackage(),
- %loadPackage(), %loadPackageS(),
- %loadPackageAddCnt(),
- %unloadPackage(),
- %extendPackagesFileref(),

and

- %generatePackage().

„The wolf is sated and the sheep is whole”

github.com/yabwon/SAS_PACKAGES



sas
packages

the way to share

„The wolf is sated and the sheep is whole”

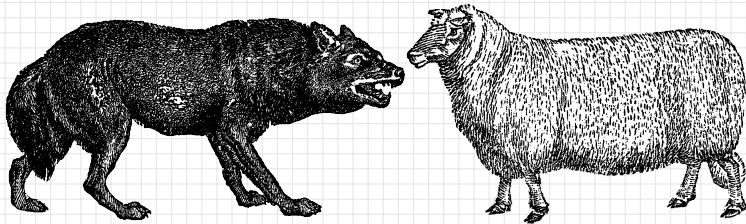
github.com/SASPAC

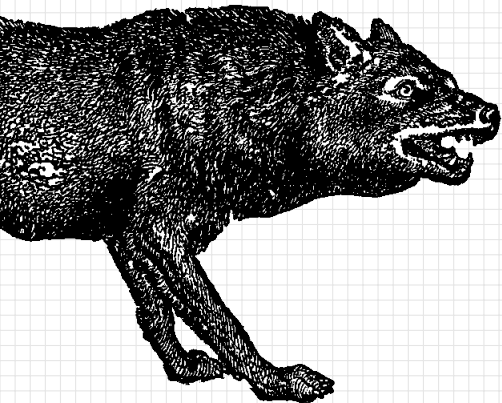


sas
packages

the way to share

The User & The Developer





Files and folders

- Create a folder for packages, e.g. C:/PCKG
- Install the framework: SPFinit.sas*

Code

```
filename packages "C:/PCKG";  
%include packages(SPFinit.sas);  
  
%installPackage(packageName)  
%<help/verify>Package(packageName)  
  
%loadPackage(packageName)
```

What next?

- Read the log
- Use the package ;-)

Files and folders

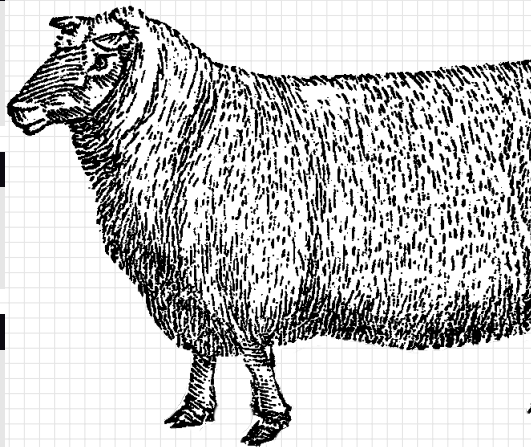
- Install SPFinit.sas* file, e.g. into ~/saspackages
- Create package folder, e.g. ~/saspackages/myPackage
- Prepare package files and the description file.

The Code

```
filename packages "~/saspackages";  
%include packages(SPFinit.sas);  
%generatePackage(~/saspackages/myPackage)
```

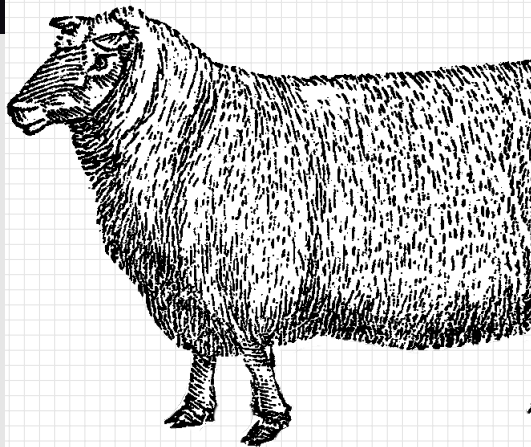
The Result

- Read the summary and the log
- Check tests results
- Share the package ;-)



Available types

```
libname,  
macro,  
functions,  
formats,  
imlmodule,  
proto,  
data, lazydata,  
exec, clean,  
casludf,  
addcnt,  
(test).
```



description.sas

```
Type: Package
Package: ThePackageName
Title: Short description, a single sentence.
Version: x.y.z
Author: Fname1 Lname1 (xxx1@yyy.zz), Fname2 Lname2 (xxx2@yyy.zz)
Maintainer: Fname3 Lname3 (xxx3@yyy.zz)
License: XYZ17
Encoding: UTF8
```

```
Required: "Base SAS Software", "SAS/ACCESS Interface to ABC"
ReqPackages: "somePackage (3.14)", "otherPackage (42)"
```

DESCRIPTION START:

```
  Lorem ipsum dolor sit amet, ThePackageName consec tetur
  adipis cingelit. Nullamdapibus lacus a elit congue
  elementum. Suspendisse iaculis ipsum nec ante luctus
  volutpat. Donec iaculis laoreet tristique.
```

DESCRIPTION END:

OK, but why...?

- Not only macros. You can use user defined functions (FCMP and CASL), IML modules, proc proto C routines, formats, and even data generating code in a package.
- Automatic update of the `cmplib=` and the `fmtsearch=` options for functions and formats.
- Loading order of the code is organised the way the developer want it to be.
- It is all in 1 (one) file - you won't forget to share "all that is needed" with your peers.
- Functionality over complexity - share one file and say, e.g. "Here is the macro %ABC(), you use it like this and that" and you don't have to say that there are 73 other utility macros working in the "background".
- A package contains additional metadata (e.g. version number or generation timestamp).
- Package genuineness can be verified and package content can be easily previewed.
- Help info is printed automatically in the log.
- A package can be loaded into the SAS session even if you don't have access to the SPF macros.
- "Shareable" between different OS.
- Package cleans after itself.
- Supports dependencies between packages.
- An additional content is also available.

thank you

✉ yabwon  gmail . com

- Bartosz Jabłoński, SAS Packages Framework, at:
https://github.com/yabwon/SAS_PACKAGES
- Bartosz Jabłoński, "My First SAS Package - a How To",
to be published in SAS Global Forum 2021 Proceedings, 1079-2021
available at:
<https://communities.sas.com/t5/SAS-Global-Forum-Proceedings/My-First-SAS-Package-A-How-To/ta-p/726319>
https://github.com/yabwon/SAS_PACKAGES/blob/main/SPF/Documentation
- Bartosz Jabłoński, "SAS Packages: The Way to Share (a How To)",
SAS Global Forum 2020 Proceedings, 4725-2020
<https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4725-2020.pdf>
extended version available at:
https://github.com/yabwon/SAS_PACKAGES/blob/main/SPF/Documentation
- The wolf and sheep are from *ClipArt ETC* archive, <https://etc.usf.edu/clipart/>