m_utl_xls2ds.sas File Reference

Utilities

Utility macro to import a XLS file into SAS datasets or tables

---

Description

This program converts an Microsoft Excelsheet into one or more SAS datasets or database tables. The macro will process the file looking for existing worksheets, and import each worksheet as a dataset or database table. The imported datasets or tables are loaded into the library identified by the OUT_LIB value. Furthermore it is possible to provide a worksheet as parameter. In this case only the provided worksheet will be imported.

Note

The engine parameter can be set to two values: XLSX or PCFILES. In most actual cases the engine set to XLSX will be sufficient to import Microsoft Excel files with .xlsx extension. For files created by an older version of Microsoft Excel or equivalents, or in case of having a mixed 32/64 bit environment, the engine parameter value should be set to PCFILES. This means that there must be a SAS PC Files Server running in the environment to be able to use the PCFILES engine setting.

Autors

Paul Alexander Canals y Trocha (paul.canals@gmail.com)

Date

2023-07-27 00:00:00

Version

23.1.07

Link

https://github.com/paul-canals/toolbox

Parameters

Input	help	Parameter, if set (or ?) to print the Help information in the log. In all other cases this parameter should be left out from the macro call.
Input	in_file	Specifies the name and location of the XLS(X) file. The default value for IN_FILE is: _NONE_.
Input	out_lib	Specifies the LIBNAME name in which the output SAS datasets or database tables will be stored. The default value for OUT_LIB is: _NONE_.
Input	engine	Indicator [XLSX|PCFILES] parameter to specify the engine that will be used to import Excel files with XLS or XLSX extension. In most cases the XLSX library engine will be sufficient to import Excel files. For files that are created by an older version of Excel and/or in case of a mixed 32/64 bit environment the PCFILES engine is to be used. The SAS PC-Files Server will take care of the 32/64 bit conversion. The default value for ENGINE is: XLSX.
Input	pcfport	Optional for PCFILES only. Specifies the port number or service name that SAS PC Files Server is listening to on the computer. The default value for PCFPORT is: 9621.
Input	pcfhost	Optional for PCFILES only. Specifies the name of the computer that is running the SAS PC Files Server. This name is required for Linux and UNIX users to connect to the SAS PC Files Server. The default value for PCFHOST is: localhost.
Input	pcfuser	Optional for PCFILES only. Specifies the domain name and user ID for the PC that is running as SAS PC Files Server. Always enclose the value in quotation marks. Otherwise, the domain backslash can be misinterpreted by the SAS parser. The default value for PCFUSER is: ''.
Input	pcfpass	Optional for PCFILES only. Specifies the password for the SAS PC Files Server for the User ID given. If the account has no password, omit this option. Always enclose the password in quotation marks in order to preserve the character case of the password. The default value for PCFPASS is: ''.
Input	sepchar	Specifies the separator character that is used as delimiter for the worksheet listing. In cases where the default SEPCHAR value collides with a worksheet name another character can be used as delimiter. The default value for SEPCHAR is: #.
Input	worksheet	Optional. Parameter to specify a specific worksheet that is to be imported for the Excel file. If this parameter is omitted all existing worksheets will be imported. The default value for WORKSHEET is: _ALL_.
Input	debug	Boolean [Y|N] parameter to provide verbose mode information. The default value is: N.

Returns

- Output SAS dataset or database table.

Calls

- m_utl_delete_file.sas
- m_utl_get_sashelp.sas
- m_utl_print_message.sas
- m_utl_print_mtrace.sas
- m_utl_printto.sas
- m_utl_valid_name.sas

Usage

Example 1: Show help information:

```
%m_utl_xls2ds(?)
```

Example 2: Step 1 - Export the CLASS table to a XLSX file named class.xlsx:

```
proc export
   data=SASHELP.class(where=(SEX='M'))
   dbms=xlsx
   outfile="%sysfunc(getoption(WORK))/class.xlsx"
   replace;
   sheet='Male';
run;

proc export
   data=SASHELP.class(where=(SEX='F'))
   dbms=xlsx
   outfile="%sysfunc(getoption(WORK))/class.xlsx"
   replace;
   sheet='Female';
run;
```

Example 2: Step 2 - Import the class.xlsx file with all existing worksheets:

```
%m_utl_xls2ds(
   in_file = %str(%sysfunc(getoption(WORK))/class.xlsx)
 , out_lib = WORK
 , engine  = xlsx
 , debug   = Y
   );

proc print data=WORK.male;
run;

proc print data=WORK.female;
run;
```

Example 3: Step 1 - Export the CLASSFIT table to a XLS file named classfit.xls:

```
*proc export
*   data=SASHELP.classfit
*   dbms=excelcs
*   outfile="%sysfunc(getoption(WORK))/classfit.xls"
*   replace;
*   sheet='Fitness';
*run;
```

Example 3: Step 2 - Import the class.xls file with worksheet Class:

```
*%m_utl_xls2ds(
*   in_file   = %str(%sysfunc(getoption(WORK))/classfit.xls)
* , out_lib   = WORK
* , engine    = pcfiles
* , pcfport   = 9621
* , pcfhost   = localhost
* , pcfuser   = "" %* user always in quotes, even if not set ;
* , pcfpass   = "" %* pass always in quotes, even if not set ;
* , worksheet = Fitness
* , debug     = Y
*   );

*proc print data=WORK.fitness;
*run;
```

Example 4: Step 1 - Create a table EXAMINED with a date format column:

```
data WORK.examined;
   set SASHELP.class;
   attrib Date format=ddmmyyp10.;
   if Name ne 'John' then
   date=today();
run;
```

Example 4: Step 2 - Export the EXAMINED table to a XLS file named examined.xlsx:

```
proc export
   data=WORK.examined
   dbms=xlsx
   outfile="%sysfunc(getoption(WORK))/examined.xlsx"
   replace;
   sheet='Class';
run;
```

Example 4: Step 3 - Import the examined.xlsx file with date column:

```
%m_utl_xls2ds(
   in_file   = %str(%sysfunc(getoption(WORK))/examined.xlsx)
 , out_lib   = WORK
 , engine    = xlsx
 , worksheet = Class
 , debug     = Y
   );

proc print data=WORK.examined;
run;
```

Copyright