m_utl_ds2xml.sas File Reference

Utilities

Utility macro to export a SAS dataset or table into a XML file

---

Description

This program converts a SAS dataset or database table into one extensible markup file with the XML extension. The program uses the XML or XMLV2 libname engine and works much like any other SAS engines. A libname statement is executed to assign a libref named MyXML and specified by the XML or XMLV2 engine. The XMLV2 engine is available since SAS 9.2 on all supported operating environments. For previous SAS versions and/or using MSACCESS as as xmltype or providing a xmlmap file, the basic XML engine is used for exporting a SAS dataset or database table to a xml file.

Autors

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

Date

2022-11-08 00:00:00

Version

22.1.11

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	base_ds	Full LIBNAME.TABLENAME name of the SAS dataset. The default value is: _NONE_.
Input	xls_file	Specifies the output XML file name and path where the formatted output is to be written. If the file that you specify does not exist, then its created for you. The default value is: _NONE_.
Input	engine	Indicator [XML|XMLV2] parameter to specify the engine that is to be used to import the XML file. The default value is: XMLV2.
Input	xmltype	Indicator [GENERIC|MSACCESS|XMLMAP] to specify the XML markup type. When set to GENERIC, the XML file consists of a root (enclosing) element and repeating element instances. It will determine the variable attributes from the data content. When set to MSACCESS, then the XML markup supports a Microsoft Access database. If the Microsoft Access file contains metadata-related information, then the XMLMETA=SCHEMADATA option is to be set to obtain variable attributes from the embedded schema. If there is not an embedded schema, MSACCESS uses default values for attributes. When set to XMLMAP, the XML markup is determined by an XMLMap, which is an XML document that contains specific XMLMap syntax. The XMLMap syntax tells the XML engine how to map the specific XML document structure back into SAS data. To specify the XMLMap in the LIBNAME statement, use the XMLMAP= option. The default value is: GENERIC.
Input	xmlmeta	Indicator [DATA|SCHEMA|SCHEMADATA] to specify whether to export metadata-related information that is included in the input XML document. The Metadata- related information is metadata that describes the characteristics (types, lengths, levels, and so on) of columns within the table markup. When set to DATA, the metadata-related information is ignored and only data content is exported into the input XML document. When set to SCHEMA, data content is ignored and only metadata-related information is included into the the input XML document. When set to SCHEMADATA, both data content and metadata-related information is exported into the output XML document. The default value is: SCHEMADATA.
Input	xmlmap	Optional parameter to specify name and location of the file that contains specific XMLMap syntax with an MAP extension. The syntax tells the XML engine how to interpret the XML markup for exporting. The XMLMap syntax itself is XML markup.
Input	xmlschema	Optional parameter to specify an external file that contains metadata-related information. This option is only to be used for GENERIC and MSACCESS markup types with XMLMETA=SCHEMADATA. If XMLMETA=SCHEMADATA and XMLSCHEMA= is specified, the data is written to the physical location of the XML document specified in the LIBNAME statement. Separate metadata-related information is written to the physical location specified with the XMLSCHEMA parameter. If XMLSCHEMA is not specified, the metadata-related information is embedded with the data content in the document.
Input	xmldataform	Optional indicator [ATTRIBUTE|ELEMENT] to specify whether the xmltag for the element that contains variable information (name and data) is in an open element or enclosed attribute format. Use of this option is valid for GENERIC markup type only and where the XMLSCHEMA parameter is not specified.

Input	xmlencoding	Optional parameter to override the default SAS dataset encoding for the output XML file. If an encoding value contains a hyphen, enclose the value in quotation marks. Use this option with caution. If you are unfamiliar with character sets, encoding methods, or translation tables, do not use this option without proper technical advice.
Input	password	Specifies the that is needed to access a password-protected dataset. This argument is required if the dataset has READ or PW password. There is no need to specify this argument if the dataset has only WRITE or ALTER passwords set.
Input	varlist	Specifies the variables that are to be included in the file and the order in which they should be included. To include all of the variables in the data set, do not specify this argument. If you want to include only a subset of all the variables, then list each variable name and use single blank spaces to separate the variables. Do not use a comma in the list of variable names.
Input	where	Specifies a valid WHERE clause that selects observations from the SAS dataset. Using this argument subsets your data based on the criteria that you supply for where-expression.
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_create_dir.sas
- m_utl_delete_file.sas
- m_utl_print_message.sas
- m_utl_print_mtrace.sas

Usage

Example 1: Show help information:

```
%m_utl_ds2xml(?)
```

Example 2: Export a SAS dataset into a generic format XML file:

```
%m_utl_ds2xml(
   base_ds  = SASHELP.class
 , xml_file = %sysfunc(getoption(WORK))/class.xml
 , debug    = Y
   );

filename class "%sysfunc(getoption(WORK))/class.xml";

data _null_;
   infile class;
   input;
   put _infile_;
run;

filename class clear;
```

Example 3: Export a SAS dataset into a Ms. Access format XML file:

```
%m_utl_ds2xml(
   base_ds  = SASHELP.classfit
 , xml_file = %sysfunc(getoption(WORK))/classfit.xml
 , xmltype  = MSACCESS
 , debug    = Y
   );

filename classfit "%sysfunc(getoption(WORK))/classfit.xml";

data _null_;
   infile classfit;
   input;
   put _infile_;
run;

filename classfit clear;
```

Example 4: Export a selection of a SAS dataset into a XML file:

```
%m_utl_ds2xml(
   base_ds  = SASHELP.cars
 , xml_file = %sysfunc(getoption(WORK))/cars.xml
 , where    = %str(upcase(Make) = 'AUDI')
 , debug    = Y
   );

filename cars "%sysfunc(getoption(WORK))/cars.xml";

data _null_;
   infile cars;
   input;
   put _infile_;
run;

filename cars clear;
```

Example 5: Export a selection of a SAS dataset into a XML file:

```
%m_utl_ds2xml(
   base_ds  = SASHELP.cars
 , xml_file = %sysfunc(getoption(WORK))/cars.xml
 , varlist  = %str(Make Model Type Origin)
 , where    = %str(upcase(Make) = 'AUDI')
 , debug    = Y
   );

filename cars "%sysfunc(getoption(WORK))/cars.xml";

data _null_;
   infile cars;
   input;
   put _infile_;
run;

filename cars clear;
```

Example 6: Export an encrypted SAS dataset into a XML file:

```
data WORK.class(encrypt=aes encryptkey=aespasskey);
   set SASHELP.class;
run;

%m_utl_ds2xml(
   base_ds  = WORK.class
 , xml_file = %sysfunc(getoption(WORK))/class.xml
 , password = aespasskey
 , debug=Y
   );

filename class "%sysfunc(getoption(WORK))/class.xml";

data _null_;
   infile class;
   input;
   put _infile_;
run;

filename class clear;
```

Example 7: Export a SAS dataset into a XML data and a XSD schema file:

```
%m_utl_ds2xml(
   base_ds   = SASHELP.class
 , xml_file  = %sysfunc(getoption(WORK))/class.xml
 , xmlschema = %sysfunc(getoption(WORK))/class.xsd
 , debug     = Y
   );

filename class "%sysfunc(getoption(WORK))/class.xsd";

data _null_;
   infile class;
   input;
   put _infile_;
run;

filename class clear;
```

Example 8: Export a SAS dataset into a XML file with UTF-8 encoding:

```
%m_utl_ds2xml(
   base_ds     = SASHELP.class
 , xml_file    = %sysfunc(getoption(WORK))/class.xml
 , xmlencoding = %str('utf-8')
 , debug       = Y
   );

filename class "%sysfunc(getoption(WORK))/class.xml";

data _null_;
   infile class;
   input;
   put _infile_;
run;

filename class clear;
```

Example 9: Export a SAS dataset into XML by using a XML map file:

```
data WORK.teams ;
attrib
   NAME        length= $30  format=$30.  label='NAME'
   ABBREV      length= $3   format=$3.   label='ABBREV'
   CONFERENCE  length= $10  format=$10.  label='CONFERENCE'
   DIVISION    length= $10  format=$10.  label='DIVISION'
;
infile cards dsd delimiter=',' truncover;
input
   NAME        :$char.
   ABBREV      :$char.
   CONFERENCE  :$char.
   DIVISION    :$char.
;
datalines4;
"Thrashers","ATL","Eastern","Southeast"
"Hurricanes","CAR","Eastern","Southeast"
"Panthers","FLA","Eastern","Southeast"
"Lightning","TB","Eastern","Southeast"
"Capitals","WSH","Eastern","Southeast"
"Stars","DAL","Western","Pacific"
"Kings","LA","Western","Pacific"
"Ducks","ANA","Western","Pacific"
"Coyotes","PHX","Western","Pacific"
"Sharks","SJ","Western","Pacific"
;;;;
run;

filename nhl_map "%sysfunc(getoption(WORK))/nhl.map";

data _null_;
   file nhl_map;
   put '<?xml version="1.0" ?>';
   put '<SXLEMAP version="1.2">';
   put '   <TABLE name="TEAMS">';
   put '      <TABLE-PATH syntax="XPATH">/NHL/CONFERENCE/DIVISION/TEAM</TABLE-PATH>';
   put '         <COLUMN name="NAME">';
   put '            <PATH>/NHL/CONFERENCE/DIVISION/TEAM/@name</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>30</LENGTH>';
   put '         </COLUMN>';
   put '         <COLUMN name="ABBREV">';
   put '            <PATH>/NHL/CONFERENCE/DIVISION/TEAM/@abbrev</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>3</LENGTH>';
   put '         </COLUMN>';
   put '         <COLUMN name="CONFERENCE" retain="YES">';
   put '            <PATH>/NHL/CONFERENCE</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>10</LENGTH>';
   put '         </COLUMN>';
   put '         <COLUMN name="DIVISION" retain="YES">';
   put '            <PATH>/NHL/CONFERENCE/DIVISION</PATH>';
   put '            <TYPE>character</TYPE>';
   put '            <DATATYPE>STRING</DATATYPE>';
   put '            <LENGTH>10</LENGTH>';
   put '         </COLUMN>';
   put '   </TABLE>';
   put '</SXLEMAP>';
run;

data _null_;
   infile nhl_map;
   input;
   put _infile_;
run;

filename nhl_map clear;

%m_utl_ds2xml(
   base_ds  = WORK.teams
 , xml_file = %sysfunc(getoption(WORK))/nhl.xml
 , xmlmap   = %sysfunc(getoption(WORK))/nhl.map
 , debug    = Y
   );

filename nhl "%sysfunc(getoption(WORK))/nhl.xml";

data _null_;
```

```
   infile nhl;
   input;
   put _infile_;
run;

filename nhl clear;
```

Copyright