Configurable SAS® Framework for managing SAS® OLAP Cube based Reporting System

Ahmed Al-Attar, AnA Data Warehousing Consulting LLC, McLean, VA
Shadana Myers, US Census Bureau, Suitland, MD

Abstract

This paper illustrates a high-level infrastructure discussion with some explanation of the SAS codes used to implement a configurable batch framework for managing and updating SAS® OLAP Cube's data rows and row level permissions. The framework contains collection of reusable parameter driven SAS Base macros, SAS Base custom programs and UNIX/LINUX shell scripts. This collection manages typical steps and processes used for manipulating SAS files and executing SAS statements.

The SAS Base macro collections contains a group of Utility Macros that includes

- Concurrent /Parallel Processing Macros
- SAS Metadata Repository Macros
- SPDE Tables Macros
- Table Lookup Macros
- Table Manipulation Macros
- Other Macros

And a group of OLAP related Macros, that includes

- OLAP Utility Macros
- OLAP Permission Table Processing Macros

Introduction

The US Census Bureau selected SAS Enterprise BI Server as the reporting platform for their Unified Tracking System (UTS). UTS is a data warehouse providing one place to view, analyze, and make more efficient and effective decisions of Census data collected over time, from different data capture sources across Annual and Periodic surveys.

The SAS EBI Reporting System contains

- **Data:**
  - *Source Data*: ORACLE® data warehouse containing tables & views stored in denormalized and Star-Schema models.
  - *Reporting Data*: Mixture of SAS® Information Maps and SAS® OLAP Cubes with row level permissions controlling users' and groups' access to the underlying data records.
- **Visualization:**
  - SAS® Web Reports and SAS® BI Dashboards served via the SAS® Information Delivery Portal.

Background

As time went by, the SAS EBI Reporting System developers continued designing and building additional SAS® OLAP Cubes and SAS® Information Maps to satisfy the growing users' reporting needs. Gradually, the EBI Reporting System started to become a victim of its own success. The increase in the Data Warehouse's ETL window had negative impact on the time available for updating the SAS® OLAP Cubes in batch!

The batch process for a single data capture source system originally involved sequential execution of the commands/steps below within a single LINUX shell script

- Rebuilding the entire listed SAS® OLAP Cubes (Complete Refresh/Rebuild)
- Removing Interviewer Hierarchy Row Level Permissions from all defined SAS® OLAP Cubes
- Applying Interviewer Hierarchy Row Level Permissions to all defined SAS® OLAP Cubes
- Applying Location Hierarchy Row Level Permissions to all defined SAS® OLAP Cubes
- Applying CPS Hierarchy Row Level Permissions to all defined SAS® OLAP Cubes

The typical routine for developing/adding new SAS® OLAP Cube involved

1. Using SAS® OLAP Cube Studio to interactively create, and manage SAS® OLAP Cube
2. Exporting the PROC OLAP code to a physical code file (*.sas)
3. Editing the newly generated code file and inserting boiler code fragments (!!Repeated Code!!)
4. FTP the new code file to the "Compute" LINUX Server
5. Update the LINUX shell script to include a SAS command entry for executing the new code file
6. Update a text file containing a list of SAS® OLAP Cube names (Cubelist.txt)

Given what listed above, it was evident, supporting multiple data capture source systems within a shrinking batch processing time window was not possible, and we had to implement different approach, sooner than later!

In order to meet all the objectives listed below

1. Maintainability
2. Agility
3. Extendibility
4. Efficiency
5. Concurrency
6. Configurability

I developed the configurable SAS® Batch Framework discussed in this paper to replace initial programs and streamline the development and management of the SAS® EBI Reporting System.

Batch Framework Features

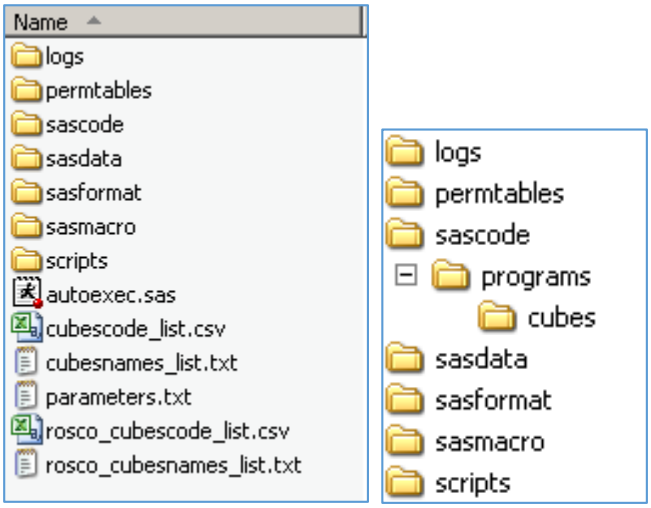
- Consistent Code: The Batch Framework utilizes parameter driven code for usability. SAS program file names and macro names have been aligned with one macro per program file. All code files have standard comments and header blocks, with consistent syntax where possible.
- Parallel/Sequential Execution: Built-in support for Parallel or Sequential execution based on supplied parameters.
- Configurable Code: The design limits the need to touch source code. For typical implementation, only the following files required modification:
    - parameters.txt: Contains collection of Key-Value pairs representing the environment configuration including Paths, Server Names, Ports, user-IDs, Librefs, etc.
    - set_batch_env.sh: Contains Unix/Linux Environment Variable settings required for Batch shell script execution.
    - autoexec.sas: Contains customer specific SAS environment settings including Librefs, Paths, and system options.
    - xxxx_list.csv: List of self-contained SAS code statement described via three comma-delimited attributes (Numeric ID, SAS Statement, and Description) used to generate execution queue dynamically.
    - xxxx_list.txt: List of Items/Objects to process stored in single column over multiple rows, where each row contains a single item/object.

Batch Framework Directory Structure

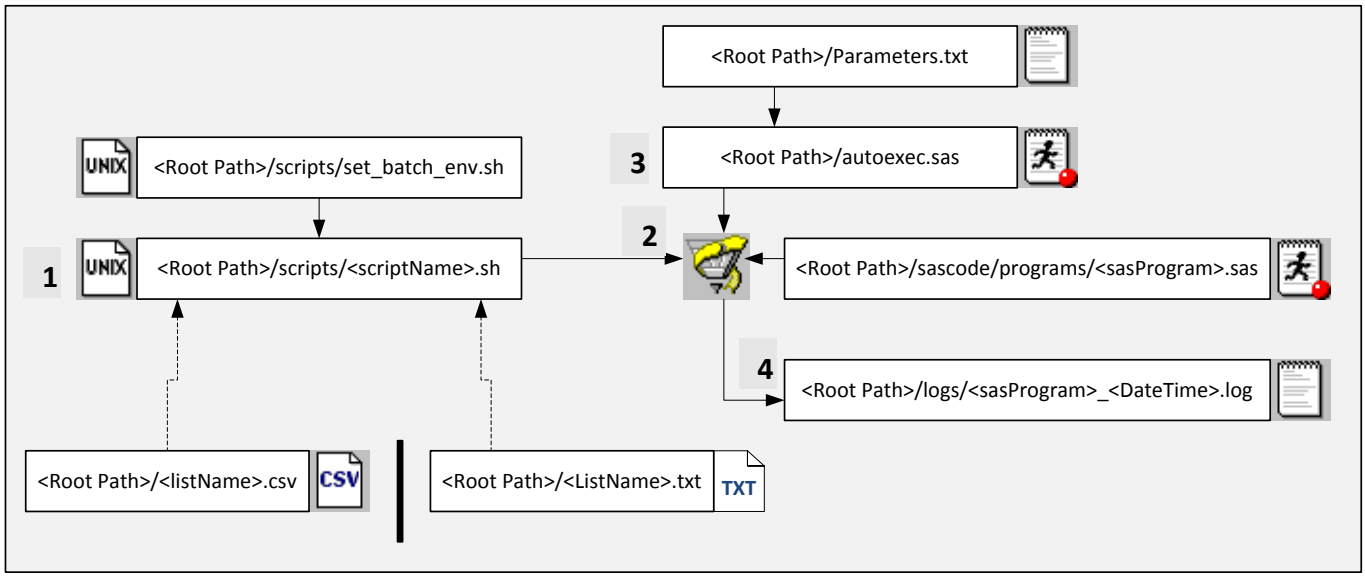
The Following table outlines the directory structure designed to support the Batch Framework. Create these directories on the SAS Compute server using the user ID (or assign appropriate ownership) that will execute the Batch process.

Directory	Description
Project Root (referred to as <projroot> in code file listing and throughout document): [ex : /data/batch/]	Top level Project directory
/logs	Location to store log files for batch process executions
/permtables	Location to store OLAP Cubes Permissions Table(s)
/sasdata	Location to store SAS Tables – Not used right now
/sasformat	Location to store SAS FORMATS catalog required for proper data manipulation within the SAS code modules
/sasmacro	Contains the reusable macros source code
/scripts	Location to store the Unix/Linux shell script files
/sascode	Custom SAS code files and sub-directories

Example Batch Framework directory structure in Windows



Batch Framework High Level Code Map & Flow



The typical flow of the execution always starts with

1. Shell script (*.sh) that has some predefined parameters/settings, this shell scripts calls the set_batch_env.sh to define prerequisite LINUX environment variables.
2. Once all prerequisite LINUX environment variables have been defined, SAS command is issued with predefined parameters for –autoexec (3), –sysin and –log (4) options, to say the least.
3. The autoexec.sas file would read-in the parameters.txt file and examine the values of the LINUX environment variables to create SAS macro variables out of it, and use them in SAS statements.
4. Each batch job would generate log file matching the SAS program it calls with DateTime suffix (_yymmdd.hhmm.log)

1.1. Batch Framework Macros

The Batch Framework utilizes a collection of reusable parameter driven SAS Base macro programs. These macros categorized based on their functionalities into

1.1.1. Utility Macros

1.1.1.1. Concurrent /Parallel Processing Macros

1.1.1.1.1. util_getidleremotesession.sas

Returns an Idle remote SAS session identifier

P_ACTIVESESSIONS	Space delimited list of active SAS/Connect MPConnect Sessions
P_RTRNSESSNAME	The name of a macro variable to hold the Idle Remote Session Identifier value

```
%util_getIdleRemoteSession(p_activeSessions=%str(RMT1 RMT2)
, p_rtrnSessName=l_idleSessn);
```

1.1.1.1.2. util_restartremotesession.sas

Restarts a remote SAS session

P_SESSNAME	The name of the SAS session.
P_ACTSESSMACVAR	The name of the macro variables holding the names/identifiers of the active
P_CREMOTE	The Name of the server to connect to
P_SASCMD	The command used to launch the SAS session

```
%util_restartRemoteSession(p_sessName=RMT1, p_actSessMacVar=g_activeSessions
, p_cRemote=local , p_sascmd=%nrstr(!sascmd));
```

1.1.1.1.3. util_startremotesessions.sas

Starts requested number of remote SAS sessions.

P_SESSCOUNT	The number of concurrent SAS sessions required.
P_SESSNAMEPREFIX	The Remote Session Name Prefix
p_activeSessionsMacVar	The Name of a macro variable to hold the identifiers of the active concurrent SAS sessions

```
%util_startRemoteSessions(p_sessCount=3
                , p_sessNamePrefix=RMT, p_activeSessionsMacVar=g_activeSessions);
```

1.1.1.1.4. util_stopremotesessions.sas

Stops all active remote SAS session passed in

P_ACTIVESESSIONS	Space delimited list of active SAS/Connect MPConnect Sessions

%util_stopRemoteSessions(p_activeSessions**=%str(RMT1 RMT2)**);

1.1.1.1.5. util_macrocallsprocessor.sas

Executes macro calls stored in a supplied data set. Depending on the supplied parameter, the execution can be sequential (Default) or parallel/concurrent using MPConnect.

P_QUERIESDSNAME	The Data Set name containing the macro calls to be executed.
P_DSNAMEWHERE	Optional Subsetting clause in case certain entries needed to be selected/excluded.
P_MPCONNECTYN	Flag to enable cocurrent Sessions
P_MPCONNECTCOUNT	Number of concurrent SAS sessions.
P_MPCONNECTAUTOEXECPATH	location of an Autoexe.sas file
P_LOGDIRPATH	Location of session log files

%util_macroCallsProcessor(
p_queriesDsName=**work.uts_createcubes**, p_DsNameWhere=
, p_mpConnectYn=**&g_useMpConnectYn**
, p_mpConnectCount=**&g_mpConnectSessnCnt**
, p_mpConnectAutoexecPath=**%str(&g_projRoot.&g_slash.autoexec.sas)**
, p_logDirPath=**%str(&g_projRoot.&g_slash.logs)**);

1.1.1.2. SAS Metadata Repository Macros

1.1.1.2.1. util_getregisteredgroups.sas

Populates the specified table name with all the registered User Groups in the Metadata Server.

p_outDsName	Output Table Name

%util_getRegisteredGroups(p_outDsName=**work.usrGrps**);

1.1.1.2.2. util_getcubeinputinfo.sas

Extracts selective SAS OLAP Cube properties for the specified Cube from the Metadata Server.

p_olapCube	SAS OLAP Cube Name including the Folder Path
p_olapSchema	SAS OLAP Schema Name
p_rtrnCubeDesc	SAS Macro variable to hold the OLAP Cube Description value. Optional
p_rtrnTblType	SAS Macro variable to hold the OLAP Cube Input Table Type value. Optional
p_rtrnTblName	SAS Macro variable to hold the OLAP Cube Input Table Name value. Optional
p_rtrnTblOptions	SAS Macro variable to hold the OLAP Cube Input Table Options value. Optional
p_rtrnDimensions	SAS Macro variable to hold the OLAP Cube Dimensions value. Optional
p_rtrnCubeFullName	SAS Macro variable to hold the OLAP Cube FullName value. Optional

```
%let l_tblType=;
%let l_tblName=;
%let l_cubeDesc=;
%let l_tblOptions=;
%let l_dimensions=;

%util_getCubeInputInfo(
        p_olapCube= CUBE_1000_PROGRESS_PERFORM_AHS
       ,p_olapSchema= SASApp-19 - OLAP Schema
       ,p_rtrnCubeDesc=l_cubeDesc
       ,p_rtrnTblType=l_tblType
       ,p_rtrnTblName=l_tblName
       ,p_rtrnTblOptions=l_tblOptions
       ,p_rtrnDimensions=l_dimensions
       ,p_rtrnCubeFullName=);
```

1.1.1.2.3. util_getcubeshortname.sas

Excludes the path from the specified OLAP Cube name and returns the short Cube name.

p_olapCubeLongName	SAS OLAP Cube Name including the Folder Path
p_rtrnCubeShortName	SAS Macro variable to hold the OLAP Cube Short Name (excluding the path) value.

%let g_shrtName=;
%util_getCubeShortName(p_olapCubeLongName=**</Folder Path/><Cube Name>**
,p_rtrnCubeShortName=**g_shrtName**);

1.1.1.3. SPDE Tables Macros

1.1.1.3.1. util_calcspdepartsize.sas

Calculates SPDE table partition size based on the size of the specified dataset and sets the value of supplied macro variable accordingly

p_libRef	SAS Library Reference
p_dsName	SAS Table name
p_numOfPaths	Number of Data Paths. Default:1
p_numOfPartitions	Number of Data partitions per Data Path. Default:4
p_minPartSize	Minimum partition size in MB. Default:128
p_DsNlObs	Number of Observations in the Table
p_rtrnMacName	Macro Variable containing returned values

%util_calcSpdePartSize(p_libRef=**dmLib**, p_dsName=**largeTable**
, p_numOfPaths=**1**, p_numOfPartitions=**4**
, p_minPartSize=**128**, p_DsNlObs=, p_rtrnMacName=**g_myMacVar**);

1.1.1.4. Table Lookup Macros

1.1.1.4.1. util_hashlkup.sas

Generates the necessary code to perform lookups using the Hash Object safely.
i.e. setting all variables to missing if a match is not found in the lookup data set

p_libName	The Library reference containing the lookup table
p_dsName	The lookup table name
p_hashObjName	The Name of the Hash Object instant
p_keyVars	Space separated list of variables to be used as lookup key
p_dtlVars	Space separated list of variables to be brought over from the lookup table
p_hashSize	The hash object's internal table size. Default=8

```
Data work.master(drop=rc);
        Set work.master;
        %util_hashLkup(p_libname=work, p_dsName=table1
        , p_hashObjName=h1, p_keyVars=var, p_dtlVars=%str(var2 var3 var4) );
        rc=h1.check();
        if (rc=0) then put 'Found';
        else put 'Not Found';
Run;
```

1.1.1.5. Table Manipulation Macros

1.1.1.5.1. util_locktable.sas

Places a lock upon a table so that you can modify its contents. If the table does not exist, a dummy table of the same name will be created and then locked. The optional parameter p_action can be set to 'unlock' to clear a tables lock.

p_table	table to be locked
p_action	lock or unlock. Default=lock
p_retryTimeout	If the table is already locked, how long will the process sleep before trying the lock again
p_retryLimit	If the table is already locked, how many times will the process retry the lock (between sleeps)
p_abendOnFailure	If Y, %abort return will be used to stop processing. Otherwise, g_lockrc is set to 2 and control is returned to the calling process. Default=N

```
%util_locktable(p_table=mylib.tablexyz, p_retryTimeout=1, p_retryLimit=5);
```

1.1.1.5.2. util_numobs.sas

Determines number of observations in the specified dataset

p_dsName	SAS Table name
p_rtrnMacName	Macro Variable containing returned values

%let carsObs=;
%util_numobs(p_dsName=**sashelp.cars**, p_rtrnMacVarName=**carsObs**);

1.1.1.6. Other Macros

1.1.1.6.1. util_logmsg.sas

Formats application messages in SAS log output. Displaying of the message is conditional on the value of g_logMessageLevel

p_msg	Required - Message to write in SAS log
p_msgType	Required - Message type to write in SAS log, Possible values are N (Note)/W (Warning)/E (Error) DH (diagnostic high-lvl)/DD (diagnostic detailed)
p_rc	Optional - Return code to write in SAS log
p_callingSource	Optional - Name of the module or program
p_dataStep	Optional - Is this being called from a data step? Y/N. Default=N
p_displayMsgDlm	Optional - Add the "**********" and blank line before and after each message. Default=Y

%util_logMsg(p_msg=**"Your message here" var1= var2= var2=var2fmt.** , p_msgType=**DD** , p_callingSource=yourProgramName, p_dataStep=**Y**);

Note that the text in the message must be quoted properly. p_msg is echoed 'as is' to a putlog statement

1.1.1.6.2. util_sortdlmtdvalues.sas

Populates the specified Return Macro Variable Name with the sorted values of the supplied delimited values.

p_values	delimited list of values to be sorted
p_dlm	Delimiter character separating the input values. Default (,)
p_valType_cn	C/N value indicator. Default:C
p_snglValLen	The maximum length of a single character value
p_keepDup_YN	Keep Duplicate Y/N Flag. Default:N [yes|y / no|n]
p_sortOrder	Sort Order direction [ascending|a|y / descending|d / no|n]
p_rtrnMacVarName	Macro variable name to hold the sorted values

%let l_srtdVals=;
%util_sortDlmtdValues(p_values=**%str(BF:AA:AF:CC:ZU:MN:LK)**, p_dlm=**%str(:)** ,
p_valType_cn=**C** , p_snglValLen=**2** , p_keepDup_YN=**N** , p_sortOrder=**A** ,
p_rtrnMacVarName=**l_srtdVals**);

1.1.1.6.3. util_getdynamicfilterclause.sas

Populates the specified Return Macro Variable Name with Filtering Clause that could be used for filtering table records.

The Time Interval value dictates the Filtering Clause format and components
p_interval = S (Survey) → depends on the value of p_tableType
p_tableType = BASE → SURVEY_ACRONYM IN ('[p_surveyFamily]') AND SURVEY_DISPLAY_NAME IN ('SurveyDispName1',' SurveyDispName2',..,' SurveyDispNamex')
p_tableType=STARFACT → FK_SURVEY_ID IN (SurveyId1, SurveyId 2,.., SurveyId x)

p_interval=D/M (Day/Month) → [p_dateVarName] BETWEEN [start_date] AND [end_date]

p_interval	Time Interval. D:Day/M:Month/S:Survey
p_sliceSize	Olap Cube Time Slice Size
p_dateVarName	Table's Date Filtering Variable Name
p_surveyOprName	Survey Operation Name
p_surveyMode	Survey Period Mode
p_surveyFamily	Survey Family Acronym
p_tableType	Table Type (STARFACT/BASE)

p_today	SAS Date value representing the End-Date Value
p_rtrnFilterVar	Macro Var name to hold the Dynamic Filtering Clause value.

```
/* Daily Increment */
%let g_dynFilterClause=;
%util_getDynamicFilterClause(p_interval=D, p_sliceSize=1, p_dateVarName= calendar_date ,
p_surveyOprName=, p_surveyMode=, p_surveyFamily=, p_tableType=, p_today= ,
p_rtrnFilterVar= g_dynFilterClause);
```

```
/* Survey Period Increment */
%let g_dynFilterClause=;
%util_getDynamicFilterClause(p_interval=S,p_sliceSize=3,p_dateVarName= ,
p_surveyOprName=ACS, p_surveyMode=CAPI, p_surveyFamily=ACS , p_tableType=BASE ,
p_today=, p_rtrnFilterVar= g_dynFilterClause);
```

1.1.1.6.4. util_sendemail.sas

Sends an email to an address (optional CC recipient). Multiple messages lines are possible through one parameter.

p_to	Specifies the primary recipients of the email. Required.
p_cc	Specifies the recipients to receive a copy of the email. Optional.
p_bcc	Specifies the recipients to receive a blind copy of the email. Optional.
p_subject	Specifies the subject of the message.Required.
p_attachFile	Specifies the physical name of the files to be attached to the message. Optional
p_attachContentType	Option to modify attachment specifications. Optional.
p_attachExtension	Option to modify attachment specifications. Optional.
p_msgText	Message line(s) to send in email. Multiple lines delimited by (|).

```
%util_sendEmail(p_to=email1 , p_cc=%str(email2,email3), p_bcc=
, p_subject=%str(Test message from SAS utility macro) , p_attachFile= , p_attachContentType= ,
p_attachExtension= , p_msgText=%str(This is a test|of multiple lines|message));
```

1.1.2. OLAP related Macros

1.1.2.1. OLAP Utility Macros

1.1.2.1.1. olap_closesessoins.sas

Lists and Closes All/Cube Specific Active Session on the specified OLAP Server.

p_logfile	Alternative log file physical location
p_host	Host name or network IP address of the computer hosting the SAS OLAP Server
p_userid	User ID to be used to connect to the SAS OLAP Server.
p_password	The password that corresponds to the user ID.
p_port	TCP port to which the SAS OLAP Server listens for connections. Default:5451
p_cube	SAS OLAP Cube Name being used

%olap_closeSessions(p_logfile=**%str(/tmp/tmpsaslog.log)** , p_host=**& g_olapSrvrHost** , p_userid=**& g_olapUserId** , p_password=**& g_olapUserPw**, p_port=**& g_olapPort** , p_cube=**_ALL_**);

1.1.2.1.2. olap_tmpltprocolap.sas

Provides Template Proc Olap Code shell to **rebuild** supplied OLAP Cube Name/Path

p_schema	SAS OLAP Schema Name
p_host	SAS Metadata Server Host name
p_delete_YN	Y/N Flag to Physically delete the specified OLAP Cube before re-building it. Default: Y
p_oraLibRef1	ORACLE Library Reference 1
p_port	SAS Metadata Server Port Number
p_olapCube	SAS OLAP Cube Name including the Folder Path
p_oraLibRef2	ORACLE Library Reference 2

This macro calls the following macros

- %util_getCubeShortName
- %olap_closeSessions

%olap_tmpltProcOlap(p_olapCube=, p_host=**&g_metaSrvrHost**, p_port=**&g_metaPort** , p_schema=**&g_olapSchema**, p_delete_YN=**Y**, p_oraLibRef1=**&g_oraLibRef1** , p_oraLibRef2=**&g_oraLibRef2**);

1.1.2.1.3. olap_tmpltprocolap_iu.sas

Provides Template Proc Olap Code shell to rebuild supplied OLAP Cube Name/Path While supporting Daily Incremental Updates, and Periodic (Monthly, Survey based ranges) Cube Rebuild

p_olapCube	SAS OLAP Cube Name including the Folder Path
p_interval	Time Interval. M:Month/S:Survey. Default:M
p_timeSliceSize	Olap Cube Time Slice Size
p_dateVarName	Table's Date Filtering Variable Name
p_surveyOprName	Survey Operation Name
p_surveyMode	Survey Period Mode
p_surveyFamily	Survey Family Acronym
p_host	SAS Metadata Server Host name
p_port	SAS Metadata Server Port Number
p_schema	SAS OLAP Schema Name
p_olapSrvrHost	SAS OLAP Server Host Name
p_olapPort	TCP/IP Port used by the SAS OLAP Server Process. Default: 5451
p_olapUserId	Registered SAS Metadata User ID to use when logging into the OLAP Server
p_olapUserPw	Registered SAS Metadata Password. Please use Encrypted password.
p_delete_YN	Y/N Flag to Physically delete the specified OLAP Cube before re-building it. Default: N
p_oraLibRef1	ORACLE Library Reference 1
p_oraLibRef2	ORACLE Library Reference 2

This macro calls the following macros

- %util_getCubeShortName
- %util_getCubeInputInfo
- %util_getDynamicFilterClause
- %olap_closeSessions

When (&p_delete_YN = Y) OR (DayofMonth(sysdate) = 1)

→ Cube Rebuild Based on specified Interval and Time Slice Size settings.

Else

→ Single Day Incremental Update

```
/* Monthly Interval - Star Schema - Example */
%olap_tmpltProcOlap_iu(
p_olapCube=/ACS - Progress and Outcome/Cubes/CUBE_1000_PROGRESS_PERFORM_ACS
, p_interval=M, p_timeSliceSize=18
, p_dateVarName=%str(DW_CREATE_DATE,DW_UPDATE_DATE)
, p_surveyOprName=, p_surveyMode=, p_surveyFamily=
, p_delete_YN=N, p_host=&g_metaSrvrHost
, p_port=&g_metaPort, p_schema=&g_olapSchema
, p_olapSrvrHost=&g_olapSrvrHost, p_olapPort=&g_olapPort
, p_olapUserId=&g_olapUserId, p_olapUserPw=&g_olapUserPw
, p_oraLibRef1=&g_oraLibRef1, p_oraLibRef2=&g_oraLibRef2);


/* Monthly Interval - Details Table - Example */
%olap_tmpltProcOlap_iu(
p_olapCube=/CEQ - Data Collection Effort/Cubes/CUBE_350_COST_PROGRESS_CEQ
, p_interval=M, p_timeSliceSize=24
, p_dateVarName=%STR(update_date,create_date)
, p_surveyOprName=, p_surveyMode=, p_surveyFamily=
, p_delete_YN=N, p_host=&g_metaSrvrHost
, p_port=&g_metaPort, p_schema=&g_olapSchema
, p_olapSrvrHost=&g_olapSrvrHost, p_olapPort=&g_olapPort
, p_olapUserId=&g_olapUserId, p_olapUserPw=&g_olapUserPw
, p_oraLibRef1=&g_oraLibRef1, p_oraLibRef2=&g_oraLibRef2);


/* Survey Based Interval – Star Schema - Example */
%olap_tmpltProcOlap_iu(
p_olapCube=/All Surveys/Cubes/CUBE_1000_PROG_PERFORM_LASTDAY
, p_interval=S, p_timeSliceSize=24
, p_dateVarName=%str(DW_CREATE_DATE,DW_UPDATE_DATE)
, p_surveyOprName=, p_surveyMode=%str(%"CAPI%"),
p_surveyFamily=%STR(%"ACS%",%"AHS%",%"CEQ%",%"CPS%",%"NCVS%",%"NHIS%",%"SIPP
%",%"SOC%")
, p_delete_YN=N, p_host=&g_metaSrvrHost
, p_port=&g_metaPort, p_schema=&g_olapSchema
, p_olapSrvrHost=&g_olapSrvrHost, p_olapPort=&g_olapPort
, p_olapUserId=&g_olapUserId, p_olapUserPw=&g_olapUserPw
, p_oraLibRef1=&g_oraLibRef1, p_oraLibRef2=&g_oraLibRef2);
```

1.1.2.2. OLAP Permission Table Processing Macros

1.1.2.2.1. olap_removeperm.sas

Use batch aci dataset as input to remove cube permissions

p_aciOlapTable	ACI dataset used to remove cube permissions

%olap_removePerm(p_aciOlapTable=**lib.table**);

1.1.2.2.2. olap_permcode.sas

Use batch aci dataset as input to create cube permissions

p_aciOlapTable	ACI dataset used to create cube permissions

%olap_permCode(p_aciOlapTable=**lib.table**);

1.1.2.2.3. olap_permcdc.sas

Performs Change Data Capture for supplied Permission Tables (Master, Delta, and CubesList).

p_masterDsName	Name of the Master Permissions Table
p_deltaDsName	Name of the Delta Changes Permisisons Table
p_cubesDsName	Name of the Cubes List Table
p_applyPerms_YN	A Y/N Flag to indicate whether to apply the Permissions. Default:N

%olap_permcdc(p_masterDsName=**libRef.unrgstrd_Interviewer_perms** , p_deltaDsName=**WORK.unrgstrd_Interviewer_perms** , p_cubesDsName=**WORK.LIST**, p_applyPerms_YN=**N**);

1.1.2.2.4. olap_loc_regperms.sas

Creates the Location Region permission entries based on the supplied Values table.

p_inDsName	Input Values table name
p_byVar	Space delimited "BY" clause variable(s)
p_outDsName	Output Permissions table name
p_cube	SAS OLAP Cube Name
p_olapSchema	SAS OLAP Schema. Default:%superq(g_olapSchema)
p_dimension	The Dimension name. Default:Interviewer
p_permType	The code that identifies the type of the ace. Default:GD (Grant Dimension)
p_removeAce	A Y/N flag that specifies if the ACE is to be added or simply to be removed from the OMR. Default:N

%olap_loc_regPerms(p_inDsName=**WORK.LOC_regList** , p_byVar=**region** ,
p_outDsName=**%bquote(WORK.cubeIntrvwrPermTblName)** ,
p_cube=**%superq(l_cubeToProcess)** , p_olapSchema=**%superq(g_olapSchema)** ,
p_dimension=**Interviewer**, p_permType=**GD**, p_removeAce=**N**);

1.1.2.2.5. olap_ssffs_fsperms.sas

Creates the SSFFS **FS** permission entries based on the supplied Values table.

p_inDsName	Input Values table name
p_byVar	Space delimited "BY" clause variable(s)
p_outDsName	Output Permissions table name
p_cube	SAS OLAP Cube Name
p_olapSchema	SAS OLAP Schema. Default:%superq(g_olapSchema)
p_dimension	The Dimension name. Default:Interviewer
p_permType	The code that identifies the type of the ace. Default:GD (Grant Dimension)
p_removeAce	A Y/N flag that specifies if the ACE is to be added or simply to be removed from the OMR. Default:N

%olap_ssffs_fsPerms(p_inDsName=**WORK.SSFFS_fsList**, p_byVar=**%str(fs ssf region)** ,
p_outDsName=**%bquote(WORK.cubeIntrvwrPermTblName)** ,
p_cube=**%superq(l_cubeToProcess)**, p_olapSchema=**%superq(g_olapSchema)** ,
p_dimension=**Interviewer**, p_permType=**GD**, p_removeAce=**N**);

1.1.2.2.6. olap_ssffs_ssfperms.sas

Creates the SSFFS **SSF** permission entries based on the supplied Values table.

p_inDsName	Input Values table name
p_byVar	Space delimited "BY" clause variable(s)
p_outDsName	Output Permissions table name
p_cube	SAS OLAP Cube Name
p_olapSchema	SAS OLAP Schema. Default:%superq(g_olapSchema)
p_dimension	The Dimension name. Default:Interviewer
p_permType	The code that identifies the type of the ace. Default:GD (Grant Dimension)
p_removeAce	A Y/N flag that specifies if the ACE is to be added or simply to be removed from the OMR. Default:N

%olap_ssffs_ssfPerms(p_inDsName=**WORK.SSFFS_ssfList** , p_byVar=**%str(ssf region)** ,
p_outDsName=**%bquote(WORK.cubeIntrvwrPermTblName)** ,
p_cube=**%superq(l_cubeToProcess)**, p_olapSchema=**%superq(g_olapSchema)** ,
p_dimension=**Interviewer**, p_permType=**GD**, p_removeAce=**N**);

1.1.2.2.7. olap_ssffs_regperms.sas

Creates the SSFFS Region permission entries based on the supplied Values table.

p_inDsName	Input Values table name
p_byVar	Space delimited "BY" clause variable(s)
p_outDsName	Output Permissions table name
p_cube	SAS OLAP Cube Name
p_olapSchema	SAS OLAP Schema. Default:%superq(g_olapSchema)
p_dimension	The Dimension name. Default:Interviewer
p_permType	The code that identifies the type of the ace. Default:GD (Grant Dimension)
p_removeAce	A Y/N flag that specifies if the ACE is to be added or simply to be removed from the OMR. Default:N

%olap_ssffs_regPerms(p_inDsName=**WORK.SSFFS_regList** , p_byVar=**region** ,
p_outDsName=**%bquote(WORK.cubeIntrvwrPermTblName)** ,
p_cube=**%superq(l_cubeToProcess)** , p_olapSchema=**%superq(g_olapSchema)** ,
p_dimension=**Interviewer**, p_permType=**GD**, p_removeAce=**N**);

1.1.2.2.8. olap_cpses_perms.sas

Creates the CPS_ES permission entries based on the supplied Values table.

p_outDsName	Output Permissions table name
p_cube	SAS OLAP Cube Name
p_sPeriod	Latest Survey Period
p_olapSchema	SAS OLAP Schema. Default:%superq(g_olapSchema)
p_dimension	The Dimension name. Default:Interviewer
p_permType	The code that identifies the type of the ace. Default:GD (Grant Dimension)
p_removeAce	A Y/N flag that specifies if the ACE is to be added or simply to be removed from the OMR. Default:N

%olap_CpsEs_Perms(p_outDsName=**%bquote(WORK.cubeSrvyPrdPermTblName)** , p_cube=**%superq(l_cubeToProcess)** , p_sPeriod=**%superq(l_sPeriod)** , p_olapSchema=**%superq(g_olapSchema)** , p_dimension=**%str(Survey Period)** , p_permType=**GD**, p_removeAce=**N**);

1.1.2.2.9. olap_singlecubeperms.sas

SAS macro that updates SAS OLAP Cube Permissions tables for a single specified OLAP Cube.

p_cubeToProcess	SAS OLAP Cube Name
p_cubeNumber	Integer Sequential value
p_intPrmsDsName	Interviewer Permissions Table Name
p_spPrmsDsName	Survey Period Permissions Table Name

%olap_singleCubePerms(p_cubeToProcess=**%str(&l_cubeToProcess)** , p_cubeNumber=**&l_cubeNum** , p_intPrmsDsName=**WORK.Interviewer_perms** , p_spPrmsDsName=**WORK.SurveyPeriod_perms**);

1.2. Batch Framework Configuration & Customization

In order to integrate the Batch Framework into an environment, one must modify the following two files

- <Root Path>/parameters.txt
  The collection of Key-Value pair parameters listed below needs modification. These parameters are converted into SAS global macro variables by default inside the <Root Path>/autoexec.sas program.

Key	Value	Description
g_frmatlib		Path holding Custom Formats catalog
g_permlib		Path holding the OLAP Permissions Table(s)
g_metaSrvrHost		SAS Metadata Server Host Name
g_metaPort	8561	TCP/IP Port used by the SAS Metadata Server Process
g_metaUserId		Registered SAS Metadata User ID to use when logging into the Metadata Server
g_metaUserPw		Registered SAS Metadata Password. Please used Encrypted password.
g_olapSrvrHost		SAS OLAP Server Host Name
g_olapPort	5451	TCP/IP Port used by the SAS OLAP Server Process
g_olapUserId		Registered SAS Metadata User ID to use when logging into the OLAP Server
g_olapUserPw		Registered SAS Metadata Password. Please used Encrypted password.
g_olapSchema		SAS OLAP Schema name
g_oraLibRef1		ORACLE Library Reference 1
g_oraPath1		ORACLE Database Aliase
g_oraSchema1		ORACLE Schema name containing the desired tables and views
g_oraUserId1		ORACLE User name
g_oraUserPw1		ORACLE Password
g_oraBuffSize		Number of rows of ORACLE data to read into the buffer in each fetch operation
g_oraLibRef2		ORACLE Library Reference 2
g_oraPath2		ORACLE Database Aliase

g_oraSchema2		ORACLE Schema name containing the desired tables and views
g_oraUserId2		ORACLE User name
g_oraUserPw2		ORACLE Password

Note: *If needed, we can add additional parameters to this file. Tab separates the Key from the Value, and each line contains single Key-Value pair.*

- <Root Path>/scripts/set_env_batch.sh
  The collection of Unix/Linux Environment Variables listed below needs modification. These Environment Variables used by the different Shell Scripts under the <Root Path>/scripts directory.

Key	Default/Current Value	Description
SASROOT	/apps/SAS/v9.2/SASFoundation/9.2	Full directory path containing SAS executable
PROJROOT		Batch Framework installation directory
AUTOEXEC	$PROJROOT/autoexec.sas	Name of the SAS Autoexec files
SASCODEROOT	$PROJROOT/sascode	Location of UTS Final/Production Code Image root dir
PROGRAMROOT	$SASCODEROOT/programs	Location of SAS Program code Files
CUBEPROGRAMROOT	$SASCODEROOT/programs/cubes	Location of SAS Program code Files
SASMACROROOT	$PROJROOT/sasmacro	Location of custom SAS Macro Files
SASFMTROOT	$PROJROOT/sasformat	Location of custom SAS Format catalogs
PROJSCR	$PROJROOT/scripts	Location of the UTS Batch Jobs shell scripts
PROJLOG	$PROJROOT/logs	Location of the UTS Batch Jobs logs files
SASMEMSIZE		SAS session Memsize setting
SASSORTSIZE		SAS session Sortsize setting
SUPPORTEMAIL		UTS System Support Email Address
MSGFILE	/tmp/emailmsg	File Containing Shell Script latest email message text
SCRIPT		Absolute path to the current script
SASENVNAME		SAS EBI Environment (Dev, Beta, Int, Prod)

Note: *If needed, we can add additional parameters to this file.*

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Ahmed Al-Attar
AnA Data Warehousing Consulting, LLC.
McLean, VA 22101
Cell Phone: 703-477-7972
E-mail: ahmed.al-attar@anadwc.com
Web: www.anadwc.com

Trademarks