Orchestrating and triggering SAS process execution from custom process control tables

Angye C. Rivero, Osmel Brito, DatAnalítica, Santo Domingo, RD

ABSTRACT

When we have different processes in charge of feeding our Data Warehouse, commonly in the design a process control table is included, from which it must be read in order to generate a trigger that starts the execution of one or more programs that carry a logical sequence at the time of loading information, thus being able to integrate the execution of SAS processes to the execution of processes from different sources or technologies.

This paper will expose you to a way to generate this flag or trigger from the data loaded into a control table with a simple and simple design, as well as the implementation so that you can customize it according to your needs.

I will also show you a practical example of how you can usea control table, your SAS programs and schedule these executions to improve the automation of the execution of your processes.

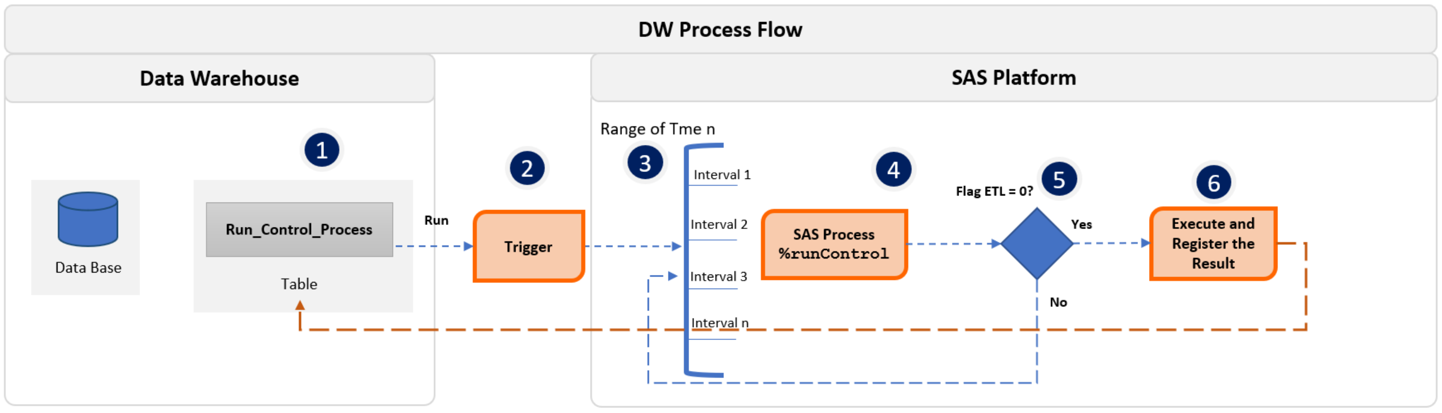
INTRODUCTION

There could be several ways to orchestrate the execution of various processes that participate in process flows both data management and business processes that take data and generate new ones where different platforms and technologies interact that in the end converge in a table that controls the order and give logical sense in how this data is generated.

That is why we present a design and approach that allows integrating different processes and technologies through the management of tools, tables and SAS programs.

So, we need to orchestrate the execution of various processes of our DW

For this we establish a design where several elements intervene that looks like this



1. La tabla **Run_Control_Process** contiene la traza de ejecución de un conjunto de procesos y sus resultados, esta tabla puede ser nombrada y customizada de acuerdo a las necesidades.
2. La ejecución del proceso SAS puede ser calendarizada. Se puede crear/editar una tarea en el SAS Management Console y/o en el Computer Management Console para configurar el trigger.
3. El proceso puede configurarse para que se ejecute dentro de un determinado rango de tiempo en un fecha específica, el programa esta diseñado para que este pueda repetirse por intervalos de tiempo hasta que consiga un registro de ejecución exitoso.
4. En este paso se ejecuta la macro `runControl`, que es la encargada de evaluar el registro de control de la tabla **Run_Control_Process** dentro del rango de tiempo indicado en el **Paso 3**, en este momento la macro genera el flag que condiciona la ejecución del siguiente proceso del negocio.
5. Se observa el valor del flag, si es cero se puede continuar con la ejecución, es decir, al **Paso 6**, en caso contrario, regresara al **Paso 3** en el siguiente intervalo de tiempo.
6. Se ejecuta el siguiente proceso del negocio y el resultado obtenido se escribe en la tabla **Run_Control_Process**. Si el resultado es satisfactorio la macro `runControl` impedirá que se repita la ejecución del proceso, por el contrario, si se registra un error, la macro `runControl` activará nuevamente el flag para que el proceso vuelva a repetirse hasta obtener el resultado deseado o que el rango de tiempo se agote.

The key question is, how do I generate the flag that is the key point in this whole process? Well, the one that does all that work is the runControl macro, which we'll explain below.

MACRO runControl.

As we said, it is in charge of generating the flag that will help us orchestrate our processes, which you can customize in line with your process control table.

%runControl

Syntax

`runControl(pLib= ,pt= ,pProcess= );`

Required Arguments

pLib

Librería donde se encuentre la tabla control

pt

Number of days back. This parameter is used to define whether the initial or previous process runs on the same day (pt=0), the day before (pt=1), or n days before (pt=n) as the next process. That is, how many days of difference there are between the initial process that generates the record in the control table that serves as the trigger for the next process.

pProcess

Identifica el proceso que se quiere ejecutar luego de activado el flag y define el nombre de la macro interna que inicializa los valores de las macro variables que sirven para personalizar la macro `runControl`.

Details

The Basics

The runControl macro generates a variable macro that represents the flag that will control the execution of the next process in your process flow, this variable macro is _vFlagETL and it only takes two values:

- _vFlagETL = 0 → this result indicates that all the conditions for the next process to run are satisfactorily met.
- _vFlagETL = 1 → this result indicates that not all the conditions for the next process to run are met and therefore cannot run.

Another important variable macro generated by the runControl macro is _run_process_date, this variable will save the value of the execution date of your initial process. For example, if your initial process is responsible for loading data into a Stage of your DW, the _run_process_date will contain the date to which the uploaded data corresponds. This variable will store the value contained in the RUN_PROCESS_DATE field of the process control table RUN_CONTROL_PROCESS which we will explain later.

Both variable macros must be declared global before running the process

```
%global _vFlagETL;
%global _run_process_date;
```

Implementation

```
/******************************************************************************************/
/*
/*  MACRO: runControl
/*
/*  USAGE: %runControl(pLib= , pt= , pProcess= );
/*
/*  DESCRIPTION:
/*    Esta macro controla el disparador desde la tabla de control del Cliente
/*
/*  NOTES:
/*    None.
/*
/*  SUPPORT: angye.r@datanalitica.com
/*
/* History :                                                                   */
/* ================                                                            */
/* version  autor        Fecha        Descripcion de Cambios, Mejoras y Adiciones  */
/* -------  -----------  ------------ -------------------------------------------- */
/* 1.0.0.0  ARR          31 Ago 2020  Versión Inicial                              */
/* 2.0.0.0  ARR          11 Ago 2022  Versión para el SESUG                         */

%macro runControl( pLib= /* Librería donde se encuentre tabla Control */
                  ,pt=     /* Cantidad Dias hacia atrás */
                  ,pProcess= /* Identificador del Proceso del Cliente*/
                  );

    %local _date _record_count _record_count _record_count_sas _sas_execution_control
           _result_code _control_code _result_code_sas _control_code_sas
           dTable dRun_Date dExecution_Control_CD vExecution_Control_CD vExecution_Control_CD_END
           dExecution_Result_CD dOrderDate OK vWhere vWhere_SAS
           ;

    %let _date = %sysfunc(INTNX(DAY, %sysfunc(today()), -&pt, S)); /* variable que toma la fecha actual y retrocede a la
fecha control especificada */
        %let _date_msg = %sysfunc(putn(&_date,date9.));
        /*%let _date_sas = %sysfunc(INTNX(DAY, %sysfunc(today()), &pt.-1, S)); /* fecha para evaluar si ya SAS ejecuto */
    %let _vFlagETL = -1; /* Flag de control de ejecución de proceso externo */
    %let _record_count = -1;
        %let _record_count_sas = -1;
    %let _sas_execution_control = 0;
    %let _result_code = ;
    %let _control_code = ;
    %let _result_code_sas = ;
    %let _control_code_sas = ;

    %macro RUN_CONTROL;
        %let dTable = RUN_CONTROL_PROCESS;
        %let dRun_Date = RUN_PROCESS_DATE;
        %let dExecution_Control_CD = EXECUTION_CONTROL_CD;
        %let vExecution_Control_CD = "STAGE";
        %let vExecution_Control_CD_END = "SAS";
        %let dExecution_Result_CD = EXECUTION_RESULT_CD;
        %let dOrderDate = EXECUTION_START_DTTM;
        %let OK = 0;
        %let vWhere = (&dRun_Date = &_date and &dExecution_Control_CD. = &vExecution_Control_CD.);
        %let vWhere_SAS = (&dRun_Date = &_run_process_date. and &dExecution_Control_CD. = &vExecution_Control_CD_END.);
    %mend;

    %&pProcess.;

    %put Variable Values &SYSDATE.: &=_vFlagETL. &=pt. &=_date.;

    /* Step 1: Create table with conditions and register of control date */
    /* crear tabla con registros de la fecha control y que cumplan con las condiciones de la variable vWhere */

    data work.EXECUTION_CONTROL;
        set &pLib..&dTable. (where = &vWhere.);
    run;

    /* Step 2: contar los registros para validar si ya existe un registro de control disparador creado */

    proc sql noprint;
        select count(*) into: _record_count
        from work.EXECUTION_CONTROL
```

```
quit;

%put &=_record_count;

/* Step 3: si ya existe un registro disparador creado proceder a la validación y generación del flag */

%if &_record_count. > 0 %then %do;

    /* Step 4: ordenar de manera descendente los registros para asegurarnos de tomar el último generado */

    proc sort data=work.EXECUTION_CONTROL;
        by descending &dOrderDate.;
    run;

    /* Step 5: tomar el último registro e inicializar las variables
      _result_code --> resultado de la ejecución
      _control_code --> valor del campo control
      _run_process_date --> fecha de ejecución o de los datos a procesar según lo definido por el negocio
    */

    data _null_;
        set work.EXECUTION_CONTROL;
            by descending &dOrderDate.;

        %put %str(NOTE: EN EL DATA _NULL_...);
        if _N_ = 1 then do;
            %put %str(NOTE: ENTRE if _N_ = 1....);
            CALL SYMPUT("_result_code", &dExecution_Result_CD.);
            CALL SYMPUT("_control_code", &dExecution_Control_CD.);
                            CALL SYMPUT("_run_process_date", &dRun_Date.);
        end; else stop;
    run;

    %put &=_result_code;
    %put &=_control_code;
        %put &=_run_process_date;


    /* Step 6: se debe evaluar si el registro disparador tiene registrado un resultado OK */

    %if &_result_code = &OK. %then %do;

        /* Step 6.1: Inicializar el flag en 0, esta variable debe ser declarada Global */
        %let _vFlagETL = 0;

        %put %str(NOTE: Registro de proceso &vExecution_Control_CD. exitoso en &dTable....);

        /* Step 6.2: Revisar si ya el proceso final escribió con resultado OK */

        /* Step 6.2.1: Crear dataset con todos los registros del proceso final para la fecha en ejecución */
                data work.EXECUTION_CONTROL_SAS;
            set &pLib..&dTable. (where = &vWhere_SAS.);
        run;

        /* Step 6.2.2: Contar los registros resultantes */
                    proc sql noprint;
            select count(*) into: _record_count_sas
            from work.EXECUTION_CONTROL_SAS
        quit;

        %put &=_record_count_sas;

        /* Step 6.2.3: Validar si se obtuvieron registros de ejecución de fin de proceso para la fecha */
        %if &_record_count_sas. > 0 %then %do;

            /* Step 6.2.4: Ordenar de manera descendente los registros para asegurarnos de tomar el último generado */
                            proc sort data=work.EXECUTION_CONTROL_SAS;
                by descending &dOrderDate.;
            run;

            /* Step 6.2.5: tomar el último registro e inicializar las variables de control de del proceso final
            _result_code_sas --> resultado de la ejecución
            _control_code_sas --> valor del campo control
            */

            data _null_;
```

```
                    set work.EXECUTION_CONTROL_SAS;
                    by descending &dOrderDate.;
                    if _N_ = 1 then do;
                        CALL SYMPUT("_result_code_sas", &dExecution_Result_CD.);
                        CALL SYMPUT("_control_code_sas", &dExecution_Control_CD.);
                    end; else stop;
                run;
                %put &=_result_code_sas;  %put &=_control_code_sas;

                /* Step 6.2.6: inicializar la variable _sas_execution_control según el resultado _result_code_sas */
                %if &_result_code_sas. eq &OK. %then %do;
                    %let _sas_execution_control = 1;
                %end;
                %else %do;
                    %let _sas_execution_control = 0;
                %end;

                        %end;
                        %else %do; /* _record_count_sas = 0 */
                /* Si el proceso final no ha ejecutado entonces se puede proceder a ejecutar */
                                %put %str(NOTE: No hay registro de proceso &vExecution_Control_CD_END. en &dTable.. Se
                                ejecutará el proceso &vExecution_Control_CD_END....);
                        %end;

            /* Step 6.3: Evaluar resultado de la variable _sas_execution_control */
            %put &=_sas_execution_control;
            %if &_sas_execution_control ne 0 %then %do;

                /* Step 6.3.1: Inicializar el flag en 1 para que no se vuelva a ejecutar el proceso, ya que se ejecuto con
                               éxito */
                %let _vFlagETL = 1;
                %put %str(NOTE: Se encontró registro de proceso &vExecution_Control_CD_END. ejecutado con éxito en
                          &dTable....);

            %end;
            %else %do;

                /* Step 6.3.2: Inicializar el flag en 0 para ejecutar el proceso, ya que se ejecuto con error y debe
repetirse */
                %let _vFlagETL = 0;
                %put %str(NOTE: No hay registro de proceso &vExecution_Control_CD_END. exitoso en &dTable.. Se ejecutará
                          el proceso &vExecution_Control_CD_END....);

            %end;

        %end;
        %else %do; /* &_result_code <> &OK. */

            /* Step 7: el registro disparador tiene registrado un resultado distinto a OK, es decir un error
                       Se inicializa el flag en 1 para que no se ejecute el proceso
            */
            %let _vFlagETL = 1;
            %put %str(WARNING: Registro del proceso &vExecution_Control_CD. fallido en &dTable.. Todavia no se puede
ejecutar el proceso &vExecution_Control_CD_END....);

        %end;
    %end;
    %else %do; /* Si no hay registro disparador, _record_count. = 0 */

        /* Step 8: no se ha generado el registro disparador en la tabla de control
                   Se inicializa el flag en 1 para que no se ejecute el proceso
        */
        %let _vFlagETL = 1;
        %put %str(NOTE: No hay registro del proceso &vExecution_Control_CD. en &dTable. para la fecha=&_date_msg....
Todavia no se puede ejecutar el proceso &vExecution_Control_CD_END....);

    %end;

    %put %str(NOTE: Se ejecuto macro runControl, el valor de _vFlagETL=&_vFlagETL....);

%mend runControl;
```

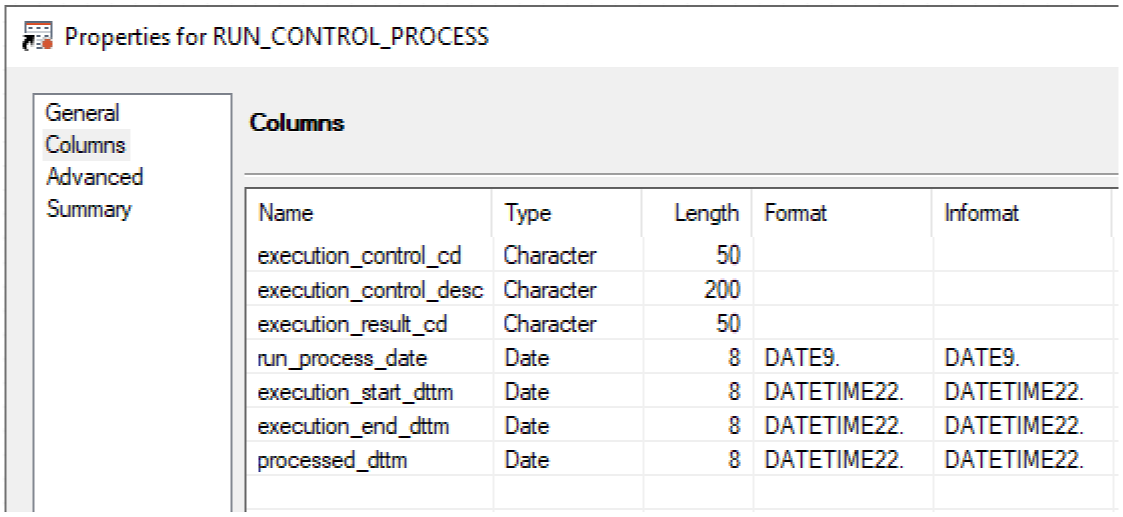
Example for this paper.

```
%runControl(pLib=sasctrl, pt=1, pPROCESS=RUN_CONTROL);
```

PROCESS CONTROL TABLE

This table is the one that contains the execution trace of several of the processes of your DW and their results, it can be named and customized according to the needs of your DW as standards and conventions established for names.

For the realization of this paper our table was created with the name **RUN_CONTROL_PROCESS** with the following fields and characteristics:



Name	Type	Length	Format	Informat
execution_control_cd	Character	50		
execution_control_desc	Character	200		
execution_result_cd	Character	50		
run_process_date	Date	8	DATE9.	DATE9.
execution_start_dttm	Date	8	DATETIME22.	DATETIME22.
execution_end_dttm	Date	8	DATETIME22.	DATETIME22.
processed_dttm	Date	8	DATETIME22.	DATETIME22.

CUSTOMIZATION OF THE RUNCONTROL MACRO

Name of the process

This is defined by the **pPROCESS** parameter of the **runControl** macro and must be the same as that used to define the internal macro responsible for initializing the variables through which the user customizes the program.

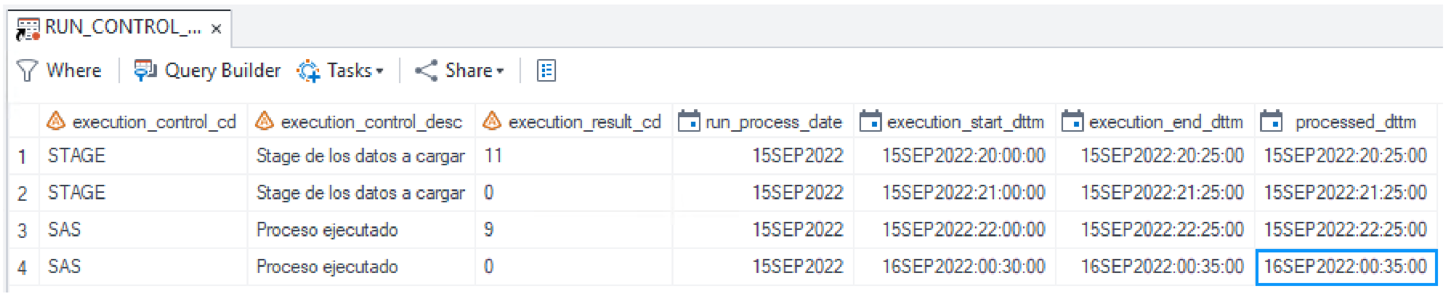
```
%macro runControl( pLib= ,pt= ,pProcess= );
…

    %macro RUN_CONTROL;
        %let dTable = RUN_CONTROL_PROCESS;
        %let dRun_Date = RUN_PROCESS_DATE;
        %let dExecution_Control_CD = EXECUTION_CONTROL_CD;
        %let vExecution_Control_CD = "STAGE";
        %let vExecution_Control_CD_END = "SAS";
        %let dExecution_Result_CD = EXECUTION_RESULT_CD;
        %let dOrderDate = EXECUTION_START_DTTM;
        %let OK = 0;
        %let vWhere = (&dRun_Date = &_date and &dExecution_Control_CD. = &vExecution_Control_CD.);
        %let vWhere_SAS = (&dRun_Date = &_run_process_date. and &dExecution_Control_CD. =
                           &vExecution_Control_CD_END.);
    %mend;

    %&pProcess.;
…

%mend runControl;


%runControl(pLib=sasctrl, pt=1, pPROCESS=RUN_CONTROL);
```

Internal Macro for Variable Initialization

This macro indicates relevant information from our process control table, values and specific conditions for the execution of the processes.

Let's make an analogy of our process control table to make the assignment of values of our variables and customize the macro.

We show table **Run_Control_Process** with two records of our initial process, which is defined as STAGE and two records of our final process, defined as SAS.



RUN_CONTROL_... ×

Where | Query Builder | Tasks | Share

	execution_control_cd	execution_control_desc	execution_result_cd	run_process_date	execution_start_dttm	execution_end_dttm	processed_dttm
1	STAGE	Stage de los datos a cargar	11	15SEP2022	15SEP2022:20:00:00	15SEP2022:20:25:00	15SEP2022:20:25:00
2	STAGE	Stage de los datos a cargar	0	15SEP2022	15SEP2022:21:00:00	15SEP2022:21:25:00	15SEP2022:21:25:00
3	SAS	Proceso ejecutado	9	15SEP2022	15SEP2022:22:00:00	15SEP2022:22:25:00	15SEP2022:22:25:00
4	SAS	Proceso ejecutado	0	15SEP2022	16SEP2022:00:30:00	16SEP2022:00:35:00	16SEP2022:00:35:00

La macro interna con sus variables quedaría de la siguiente manera:

```
%macro RUN_CONTROL;
    %let dTable = RUN_CONTROL_PROCESS;
    %let dRun_Date = RUN_PROCESS_DATE;
    %let dExecution_Control_CD = EXECUTION_CONTROL_CD;
    %let vExecution_Control_CD = "STAGE";
    %let vExecution_Control_CD_END = "SAS";
    %let dExecution_Result_CD = EXECUTION_RESULT_CD;
    %let dOrderDate = EXECUTION_START_DTTM;
    %let OK = 0;
    %let vWhere = (&dRun_Date = &_date and &dExecution_Control_CD. = &vExecution_Control_CD.);
    %let vWhere_SAS = (&dRun_Date = &_run_process_date. and &dExecution_Control_CD. =
                       &vExecution_Control_CD_END.);
%mend;
```

the **dtable** variable must contain the name of the process control table

```
%let dTable = RUN_CONTROL_PROCESS;
```

The variable **dRun_Date**, corresponds to the name of the field of the date of execution of our initial process, of the load of the data or date through which we are interested in spinning the processes.

```
%let dRun_Date = RUN_PROCESS_DATE;
```

That is, the **RUN_PROCESS_DATE** field in the table **RUN_CONTROL_PROCESS**.

The variable **dExecution_Control_CD**, corresponds to the name of the field that contains the control code established to identify each process.

```
%let dExecution_Control_CD = EXECUTION_CONTROL_CD;
```

The variable **vExecution_Control_CD**, is the value that the field must have **EXECUTION_CONTROL_CD**. It corresponds to the code of the initiation process, in this case STAGE.

```
%let vExecution_Control_CD = "STAGE";
```

The variable **vExecution_Control_CD_END**, is the value that the **EXECUTION_CONTROL_CD** field must have. It corresponds to the code of the end process, in this case SAS.

```
%let vExecution_Control_CD_END = "SAS";
```

The variable **dExecution_Result_CD** that corresponds to the name of the field that stores the result of the execution of the processes.

```
%let dExecution_Result_CD = EXECUTION_RESULT_CD;
```

A variable through which to order in a descending way the control records, **dOrderDate**, which for the case raised was taken the field of the date recorded time of start of execution of the process.

```
%let dOrderDate = EXECUTION_START_DTTM;
```

And finally, the variable **OK**, to which the value that is considered as the satisfactory result of the execution of the process must be assigned.

```
%let OK = 0;
```

The **vWhere** variable is used to identify the initial process and the variable **vWhere_SAS** to identify the final process.

Attention: the values of the **vWhere** and **vWhere_SAS** condition variables could also be adjusted according to the rules that need to be considered for a process to be identified, but care must be taken because it could affect the proper functioning of the **runControl** macro.

End the process

Every time a process is executed, a record must be added to the process control table that contains the results of the execution, in this case, SAS. It will indicate that the SAS process has completed its execution correctly and that the DW processes can continue when a record of the SAS process with satisfactory results is produced.

When inserting the table RUN_CONTROL_PROCESS, you must use the value stored in the variable macro **_run_process_date** as a value for the **RUN_PROCESS_DATE** field..

```
%macro InsertControlSAS;

    proc sql noprint;
        insert into sasctrl.run_control_process
                    (execution_control_cd, execution_control_desc,
                     execution_result_cd, run_process_date,
                     execution_start_dttm, execution_end_dttm, processed_dttm)
              values("SAS", "Result of SAS Process",
                     "&vResult.", "&_run_process_date."d,
                     "&vExecution_Start_Dttm."dt, "&vExecution_End_Dttm."dt, "&etls_endTime."dt)
                     ;
    quit;
    %put NOTE: "Macro InsertControlSAS: Se inserto en &vControlLib..&vTabla.";

%mend;
```

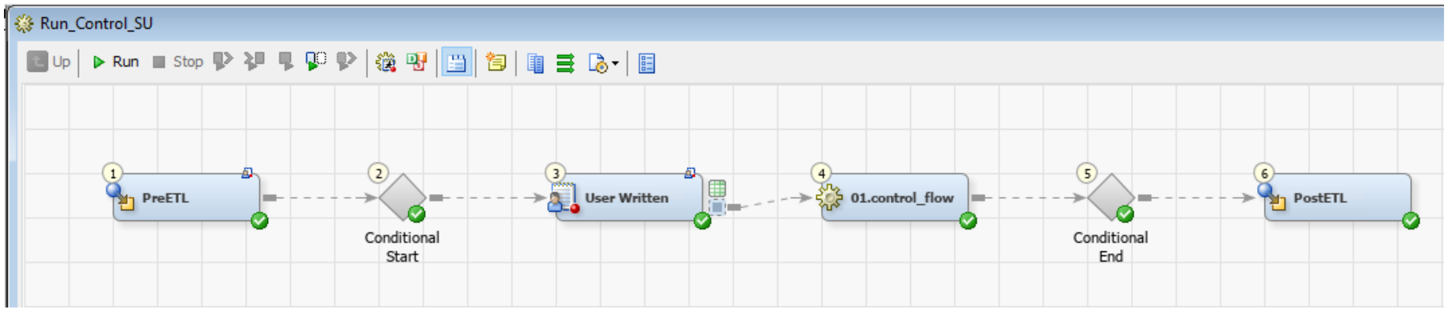
CONCLUSIONS

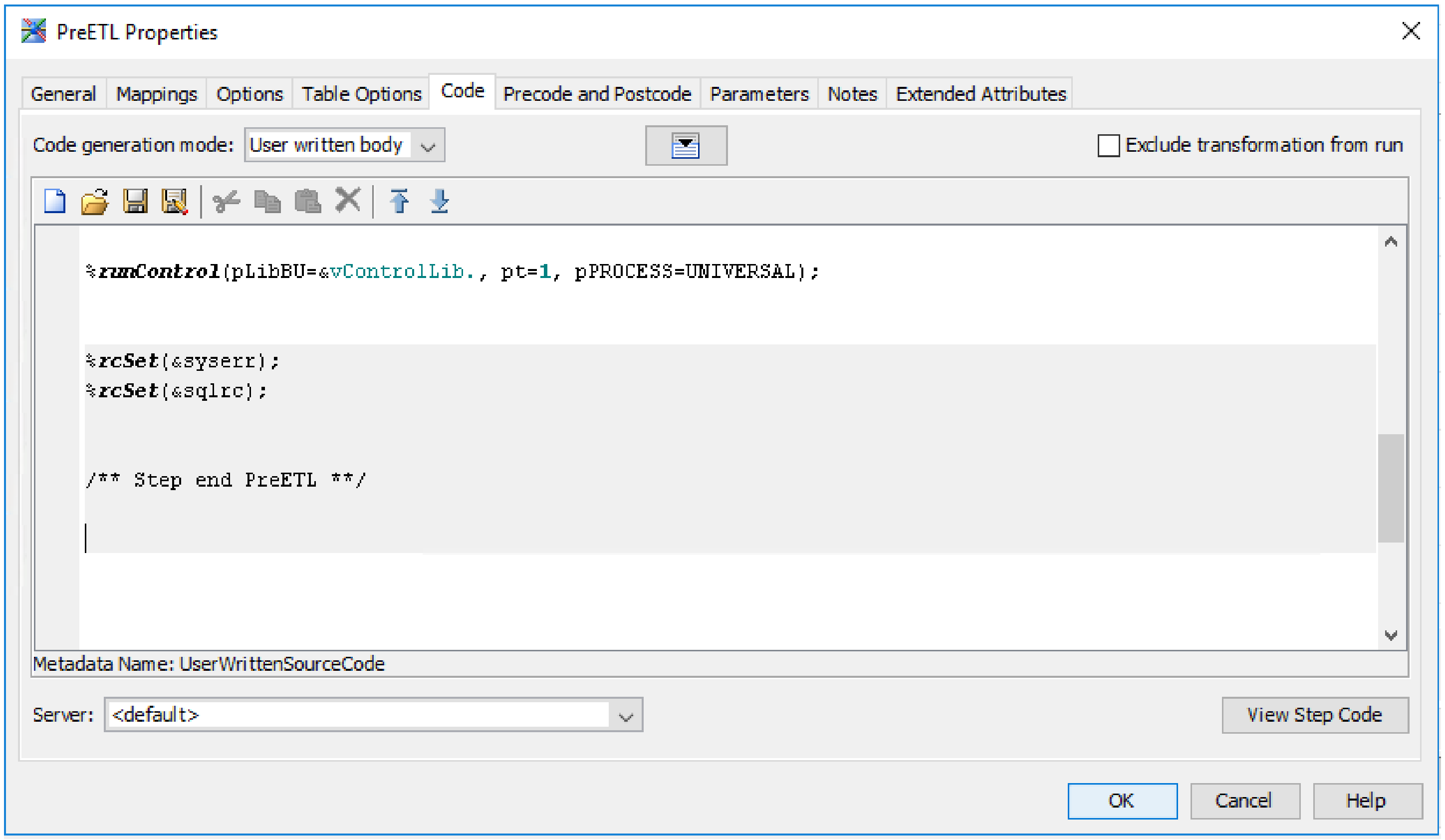
SAS programming tools and SAS code are used to orchestrate our processes regardless of what DataBase Management System our DW and business processes are in. It is important to note that the **runControl** macro can be adjusted to multiple processes simultaneously, you only need to add as many internal macros for initializing and configuring variables as you want to incorporate.

In this example, two processes are being controlled, and that is why we have a variable initialization macro for each, and the call to the **runControl** macro is made twice with the parameters **pPROCESS** corresponding to them:

```
%macro runControl( pLib= ,pt= ,pProcess= );
…

    %macro RUN_CONTROL;
        %let dTable = RUN_CONTROL_PROCESS;
        %let dRun_Date = RUN_PROCESS_DATE;
        %let dExecution_Control_CD = EXECUTION_CONTROL_CD;
        %let vExecution_Control_CD = "STAGE";
        %let vExecution_Control_CD_END = "SAS";
        %let dExecution_Result_CD = EXECUTION_RESULT_CD;
        %let dOrderDate = EXECUTION_START_DTTM;
        %let OK = 0;
        %let vWhere = (&dRun_Date = &_date and &dExecution_Control_CD. = &vExecution_Control_CD.);
        %let vWhere_SAS = (&dRun_Date = &_run_process_date. and &dExecution_Control_CD. =
&vExecution_Control_CD_END.);
    %mend;

    %macro UPDATE_CUSTOMER;
        %let dTable = RUN_CONTROL_PROCESS;
        %let dRun_Date = RUN_PROCESS_DATE;
        %let dExecution_Control_CD = EXECUTION_CONTROL_CD;
        %let vExecution_Control_CD = "STG_CUSTOMER";
        %let vExecution_Control_CD_END = "SAS_CUSTOMER";
        %let dExecution_Result_CD = EXECUTION_RESULT_CD;
        %let dOrderDate = EXECUTION_START_DTTM;
        %let OK = 0;
        %let vWhere = (&dRun_Date = &_date and &dExecution_Control_CD. = &vExecution_Control_CD.);
        %let vWhere_SAS = (&dRun_Date = &_run_process_date. and &dExecution_Control_CD. =
&vExecution_Control_CD_END.);
    %mend;


    %&pProcess.;

…

%mend runControl;


%runControl(pLib=sasctrl, pt=1, pPROCESS=RUN_CONTROL);

…

%runControl(pLib=sasctrl, pt=1, pPROCESS=UPDATE_CUSTOMER);
```

You can control theproblems to be executed either using a conditional transformation in a SAS Data Integration Studio ETL or a conditional one in SAS Enterprise Guide, as well as schedule executions using SAS Management Console and Windows Computer Management, for example:





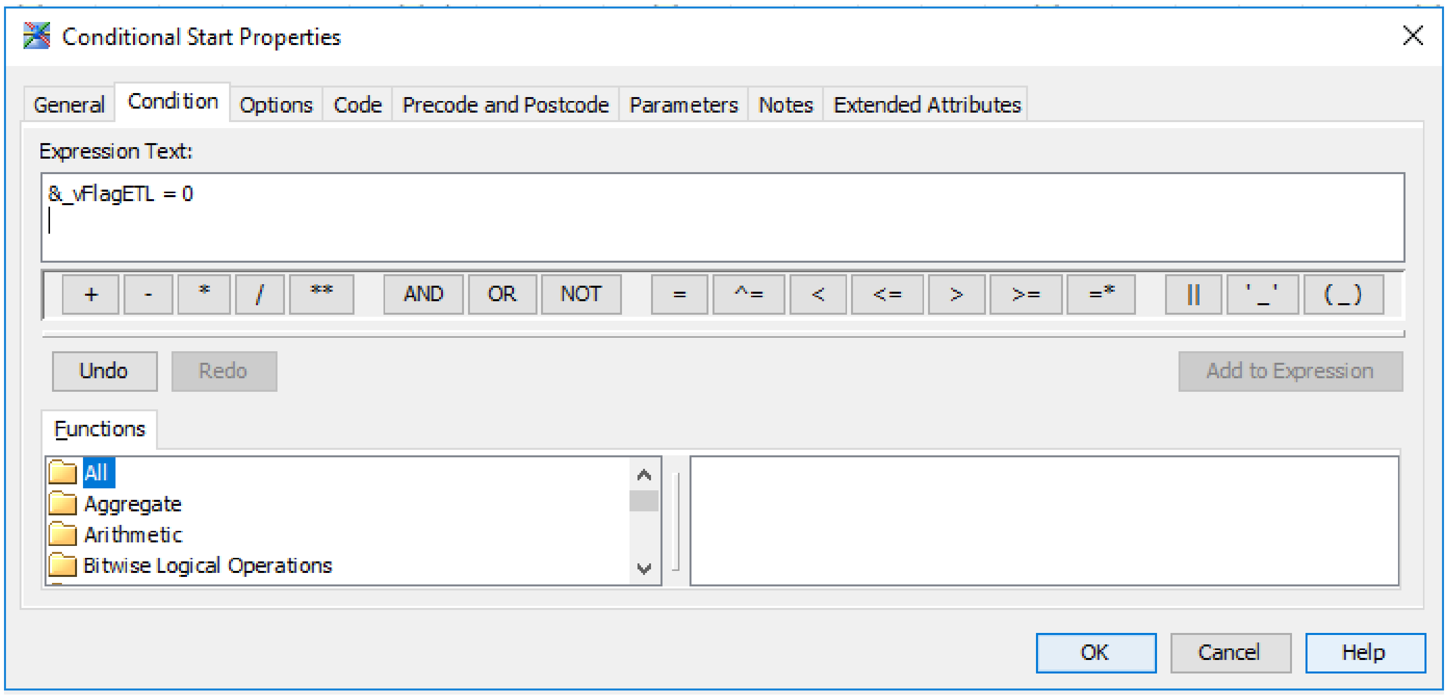
Code generation mode: User written body

Exclude transformation from run

```
%runControl(pLibBU=&vControlLib., pt=1, pPROCESS=UNIVERSAL);


%rcSet(&syserr);
%rcSet(&sqlrc);


/** Step end PreETL **/
```

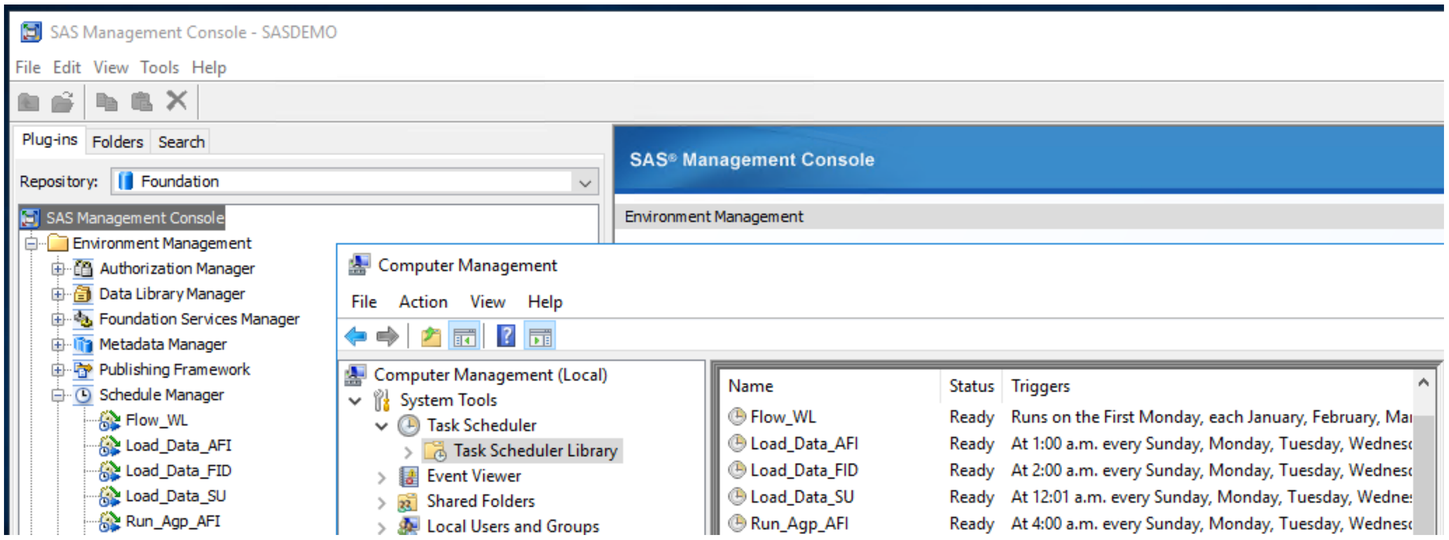
Metadata Name: UserWrittenSourceCode

Server: <default>



Expression Text:

```
&_vFlagETL = 0
```



Name	Status	Triggers
Flow_WL	Ready	Runs on the First Monday, each January, February, Ma
Load_Data_AFI	Ready	At 1:00 a.m. every Sunday, Monday, Tuesday, Wednes
Load_Data_FID	Ready	At 2:00 a.m. every Sunday, Monday, Tuesday, Wednes
Load_Data_SU	Ready	At 12:01 a.m. every Sunday, Monday, Tuesday, Wedne
Run_Agp_AFI	Ready	At 4:00 a.m. every Sunday, Monday, Tuesday, Wednes

CONTACT INFORMATION

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

DatAnalitica
7801 NW 37TH ST
DO80Q75172N
DORAL,
FLORIDA 33195-6503

Email: info@datanalitica.com

Angye Rivero: angye.r@datanalitica.com
Osmel Brito-Bigott: osmel.b@datanalitica.com