



Automation of Process Flow Instances in LSAF



Caro Sluiter

Solution to the manual repetitive task



- Easy way to circumvent the repetitiveness in LSAF when creating a process flow instance
- But before starting to explain the problem and solution
- First introduce:
 - LSAF,
 - process flows (instances)
 - how process flows work in LSAF





Life Science Analytics Framework

- LSAF = SAS Life Science Analytics Framework
- Single solution, integrated system:
 - Centralised, standardised repository
 - SAS programming and execution environment
- Key Feature: workflow capability
 - Automate processes using process flows

Process Flow

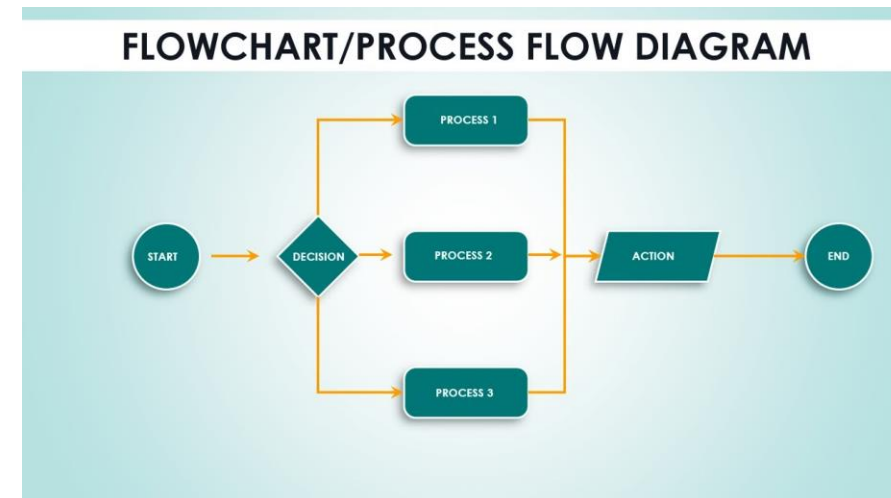


- Process flows provide a visual overview or workflow diagram of all the tasks and relationships involved in a process.
 - Helpful to allocate tasks
 - Keep track of the progress of tasks and activities
 - Reduces manual involvement
 - Automates, streamlines and tracks processes
- Benefits the management of the clinical research process.

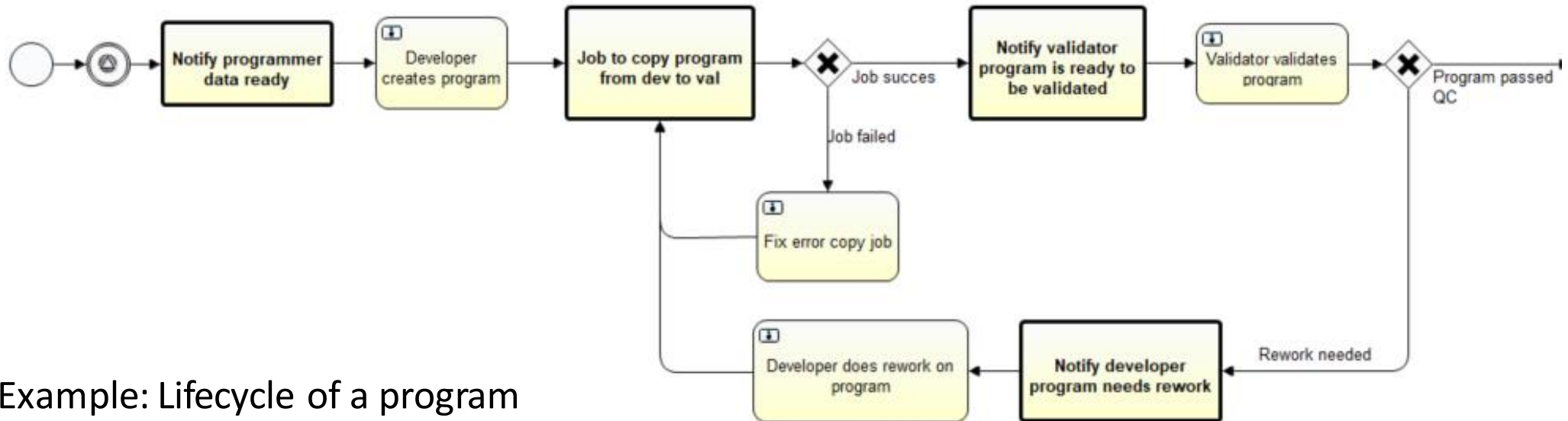


Process Flow

- Particularly useful:
Process consists of several standardized tasks to be executed repetitively across studies.
 - Exchange and validation of study data
 - Lifecycle of a program
- Contains elements that can
 - Notify someone either for a message or a task
 - Notify a group either for a message or a task
 - Activate/run a program/job
 - Pick up a signal



Process Flow



- Example: Lifecycle of a program
- Reduces manual participation involved
- Less prone to error
- Streamlines and tracks processes



Process flows in LSAF

➤ Process flow

1. Upload a process flow (BPMN file = Business Process Model and Notation file)
2. Deploy process flow

➤ Process flow instance

1. Create process flow instance
(for every study/location a new process flow instance)
 2. Assign correct pathways/users/jobs etc to the process flow instance elements
 3. Activate process flow instance
-
- Process flow instance = repetitive



Process flows in LSAF

- As shown, for every process flow instance a manual set-up of the elements needs to take place.
 - Study-specific paths
 - Study-specific notification messages
- Goal: Eliminate repetitive tasks as much as possible
- Therefore: We do not want to manually set up each process flow instance.
- Solution: Automatically set up your process flow instances.

Solution

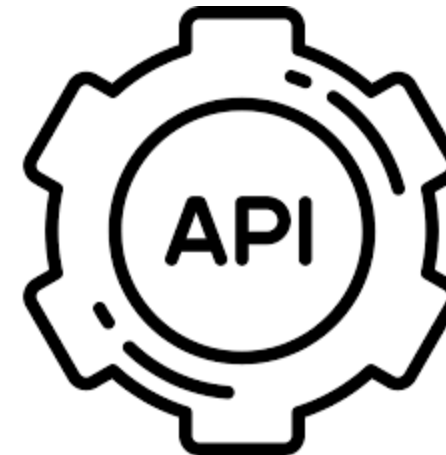


- Every manual task in LSAF can be executed using LSAF specific Application Programming Interface (API) macros
- Job that creates a process flow instance automatically
- Required:
 - Program containing the API macros
 - Job to be able to call the correct input parameters
 - Input parameters



Using API's to set-up process flow instance

- Step 0: Define global variables
- Step 1: Create the process flow instance
- Step 2: Populate the process flow instance
- Step 3: Activate the process flow instance



APIs in LSAF



- In LSAF you can make use of numerous APIs

APIs regarding the ProcessFlow

ProcessFlow	lsaf_activateprocessflow
ProcessFlow	lsaf_createprocessflow
ProcessFlow	lsaf_createprocessflowmanifest
ProcessFlow	lsaf_deleteprocessflow
ProcessFlow	lsaf_getmyprocessflows
ProcessFlow	lsaf_getprocessflowdata
ProcessFlow	lsaf_getprocessflowproperties
ProcessFlow	lsaf_getprocessflows
ProcessFlow	lsaf_processflowexists
ProcessFlow	lsaf_suspendprocessflow
ProcessFlow	lsaf_updateprocessflowdata
ProcessFlow	lsaf_updateprocessflowproperties

APIs regarding the ProcessFlowSetup

ProcessFlow Setup	lsaf_getpfsetupelements
ProcessFlow Setup	lsaf_getpfsetupjobinfo
ProcessFlow Setup	lsaf_getpfsetupjobparameters
ProcessFlow Setup	lsaf_getpfsetupnotifinfo
ProcessFlow Setup	lsaf_getpfsetupnotifrecips
ProcessFlow Setup	lsaf_getpfsetupsignallocs
ProcessFlow Setup	lsaf_getpfsetuptimers
ProcessFlow Setup	lsaf_getpfsetupusercandidates
ProcessFlow Setup	lsaf_getpfsetupuserinfo
ProcessFlow Setup	lsaf_updatepfsetupjobinfo
ProcessFlow Setup	lsaf_updatepfsetupjobparameters
ProcessFlow Setup	lsaf_updatepfsetupnotifinfo
ProcessFlow Setup	lsaf_updatepfsetupnotifrecips
ProcessFlow Setup	lsaf_updatepfsetupsignallocs
ProcessFlow Setup	lsaf_updatepfsetuptimers
ProcessFlow Setup	lsaf_updatepfsetupusercandidates
ProcessFlow Setup	lsaf_updatepfsetupuserinfo



Step 0: Define global variables

- Used to make programming life easier later in the program

```
/******  
Create global variable(s)  
*****/  
/*Split up macro path into the different sections*/  
%let path_real = %sysfunc(tranwrd(&path,&_sasws_, ""));  
%let compound = %scan(&path_real,2,"/");  
%let indication = %scan(&path_real,3,"/");  
%let study = %scan(&path_real,4,"/");  
  
%let date = %sysfunc(inputn(&sysdate9., date9.), yymmdd10.);  
%let time = %sysfunc(tranwrd(%sysfunc(time(),time6.0),:,_)); /*Time, ':' not allowed in sas name*/  
  
/*Process flow instance name*/  
%let pfname = pf_lifecycle_&compound_&indication_&study_&date_&time  
  
/*Specify group that will be assigned to tasks and/or get notifications*/  
%let groupid1 = dev_&study.; /*Developers of the study*/  
%let groupid2 = val_&study.; /*Validators of the study*/  
  
/*Location of the copy job*/  
%let jobpath1 = /Hands-On_Demos/OCS/OCS LS Process flows/OCS LS Process Flows Lifecycle/dev/programs/copytool_stop2_execute_copy_staging.sas
```



Step 1: Create the process flow instance

```
%lsaf_createprocessflow(lsaf_path      = /&path_real.  
                        ,lsaf_processflow = &pfname.  
                        ,lsaf_processflowdef = DevelopmentProgramLifecycle  
                        );
```

- This API will return an “empty” process flow instance
 - No quotes are needed
 - Value lsaf_path is the context in which the process flow instance should operate
 - Value lsaf_processflow should be unique / not yet exists



Step 2: Populate the process flow instance - UserTask

Assign a group to a UserTask

- Populate the missing information
- Update work.getpfsetupusercandidates

```
/*Create dataset with attributes for UserTask Candidates*/
data work.getpfsetupusercandidates;
  length processflowpath
          processflowname
          grpSrcCtxt
          elementid
          type
          principalid $100;

/*Determine general attributes for UserTasks*/
processflowpath = "/system";
processflowname = "&pfname.";
grpSrcCtxt      = "&path.";

/*UserTask = Create Program*/
elementid      = "createprogram";
type           = "GROUP";
principalid    = "&groupid1.";
output;

/*UserTask = Execute Rework*/
elementid      = "executerework";
type           = "GROUP";
principalid    = "&groupid1.";
output;

/*Update datasets with dataset to assign groups of candidates to
the UserTasks you want to, when done do:*/
%lsaf_updatepfsetupusercandidates(sas_dsname = work.getpfsetupusercandidates);
output;

run;
```



Step 2: Populate the process flow instance - UserTask

Assign a person to a User Task

- Populate the missing information
- Update the dataset lsafgetpfsetupuserinfo

```
data work.lsafgetpfsetupuserinfo;
  length processflowpath
          processflowname
          elementid
          name
          value $100;

  /*Determine general attributes of User information*/
  processflowpath = "/system";
  processflowname = "&pfname.";

  /*Assign person to UserTask*/
  elementID = "fixerrorcopyjob1";
  name      = "assigneeid";
  value     = "carosluijter";
  output;
run;

%lsaf_updatepfsetupuserinfo(sas_dsname=work.lsafgetpfsetupuserinfo);
```



Step 2: Populate the process flow instance - Jobs

Assign Job information

- Update the getpfsetupjobinfo to add the path of the job in the repository

```
data work.getpfsetupjobinfo;
  length processflowpath
         processflowname
         elementid
         name
         value $100 ;

  /*Determin general attributes of jobs*/
  processflowpath = "/system";
  processflowname = "&pfname.";

  /*Set location copydevtoval job*/
  name          = "jobPath" ;
  value         = "&jobpath1.";
  elementID     = "copydevtoval";
  output;

  /*Set location copyvaltoprod*/
  name          = "jobPath" ;
  value         = "&jobpath1.";
  elementID     = "copyvaltoprod";
  output;

run;

/*Update dataset with new information on job location*/
%lsaf_updatepfsetupjobinfo(sas_dsname = work.getpfsetupjobinfo);
```



Step 2: Populate the process flow instance - Jobs

Assign Job-parameter information

- Populate the getpfsetupjobparameters dataset to populate the job parameters

```
data work.getpfsetupjobparameters;
  length processflowpath
         processflowname
         elementid
         jobpath
         name
         type
         value $100;

  /*Determin general attributes of job parameters*/
  processflowpath = "&path_real.";
  processflowname = "&pfname.";
  fileVersion     = "";

  /*Add parameter for job copydevtoval*/
  elementid       = "copydevtoval";
  jobpath         = "&jobpath1.";
  name            = "input_path";
  type            = "Folder";
  value           = "&path_real./dev";
  includesubfolders = 0;
  output;

  /*Add parameters for job copydevtoval*/
  elementid       = "copydevtoval";
  jobpath         = "&jobpath1.";
  name            = "output_path";

  /*Update dataset with new information on job input parameters*/
  %lsaf_updatepfsetupjobparameters(sas_dsname = work.getpfsetupjobparameters);
  includesubfolders = 0;
  output;

run;
```



Step 2: Populate the process flow instance - Signal

- Add location for signal catching event
- /system/* means the path including all subfolders

```
data work.lsafgetpfsetupsignallocs;  
  length processflowpath  
          processflowname  
          elementid  
          location $100;  
  
  processflowpath = "/system.";   
  processflowname = "&pfname.";   
  elementid       = "catchdatauploaded";   
  location        = "/&path_real./input";   
  output;  
run;  
  
%lsaf_updatepfsetupsignallocs(sas_dsname=work.lsafgetpfsetupsignallocs);
```





Step 2: Populate the process flow instance - Notify

- Set-up Notification message
- Set-up Group Recipients of message
- Set-up Person to receive message

```
data work.lsafgetpfsetupnotifinfo;
  length processflowpath
          processflowname
          elementid
          name
          value $100 ;

  /*Determine general attributes of message*/
  processflowpath = "/system";
  processflowname = "&pfname.";

  /*Set message to be send out if new files are being uploaded*/
  elementID      = "notifydataready"
  name           = "message";
  value          = "Study: &study; Data is uploaded and ready for programming.";
  output;

  elementID      = "notifyreadyforvalidation";
  name           = "message";
  value          = "Study: &study; Program ready for validation.";
  output;

  elementID      = "notifyprograminproduction";
  name           = "message";
  value          = "Study: &study; Program moved to production";
  output;

run;

%lsaf_updatepfsetupnotifinfo(sas_dsname = work.lsafgetpfsetupnotifinfo);
```

```
data work.lsafgetpfsetupnotifrecips;
  length processflowpath
          processflowname
          grpSrcCtxt
          elementid
          type
          principalid $100;

  /*Determine general attributes for notification recipients*/
  processflowpath = "/system";
  processflowname = "&pfname.";
  grpSrcCtxt      = "&path_real.";

  /*Set-up a group to receive message*/
  elementid       = "notifydataready";
  type            = "GROUP";
  principalid     = "&groupid1.";
  output;

  elementid       = "notofyreadyforvalidation";
  type            = "GROUP";
  principalid     = "&groupid2.";
  output;

  /*Set-up a person to receive a message*/
  elementid       = "notifyprograminproduction";
  type            = "USER";
  principalid     = "carosluijter";
  output;

run;

%lsaf_updatepfsetupnotifrecips(sas_dsname = work.lsafgetpfsetupnotifrecips);
```



Step 3: Activate the process flow instance

```
%lsaf_activateprocessflow(lsaf_path = /system          /*Context*/  
                        ,lsaf_processflow = &pfname. /*Name*/  
                        );
```





Tips for creating job to automate process flows

- Use of macro variables for the variables that are needed in all datasets
 - ProcessFlowPath
 - ProcessFlowName
 - GroupID
- Use of macro variables for pathways that might change when moving a program and determining the values of these macro variables at the top of the program
 - Job paths
 - Output paths
 - Input paths

Result



- Let me show you in LSAF how quick I can now set-up a process flow instance

Conclusion



- Programmers do not like to execute repetitive tasks
 - Process flow capability reduces repetitive tasks by completing them automatically
 - Using LSAF API's will eliminate even the repetitiveness of the creation of process flow instances
 - Eliminating these tasks makes sure you can switch your focus from the process to the content



Questions: sasquestions@ocs-consulting.com

More information:

- LSAF setup : SD05: Using jobs and Macros API to create a structured, controlled and personalised LSAF
- LSAF data management: DH05: Utilising LSAF for data management at a Belgian biotech

www.ocs-lifesciences.com

OCS Life Sciences



Caro Sluijter

caro.sluijter@ocs-consulting.com
office: +31 (0)73 523 6000

www.ocs-lifesciences.com

Visit
Ruwekampweg 2G
5222 AT
's-Hertogenbosch
The Netherlands
Mail
Postbus 3434
5203 DK
's-Hertogenbosch
The Netherlands