Paper 10740-2016

# Developing an On-Demand Web Report Platform Using Stored Processes and SAS® Web Application Server

Romain Miralles, Genomic Health

.

## ABSTRACT

As SAS® programmers, we often develop listings, graphs, and reports that need to be delivered frequently to our customers. We might decide to manually run the program every time we get a request, or we might easily schedule an automatic task to send a report at a specific date and time. Both scenarios have some disadvantages. If the report is manual, we have to find and run the program every time someone request an updated version of the output. It takes some time and it is not the most interesting part of the job. If we schedule an automatic task in Windows, we still sometimes get an email from the customers because they need the report immediately. That means that we have to find and run the program for them.

This paper explains how we developed an on-demand report platform using SAS® Enterprise Guide®, SAS® Web Application Server, and stored processes. We had developed many reports for different customer groups, and we were getting more and more emails from them asking for updated versions of their reports. We felt we were not using our time wisely and decided to create an infrastructure where users could easily run their programs through a web interface. The tool that we created enables SAS programmers to easily release on-demand web reports with minimum programming. It has web interfaces developed using stored processes for the administrative tasks, and it also automatically customizes the front end based on the user who connects to the website. One of the challenges of the project was that certain reports had to be available to a specific group of users only.

## INTRODUCTION

When the company switched from SAS desktop version to SAS server with enterprise guide, it opened doors to new opportunities for SAS programmers. It was not easy at first to get used to this new environment, but it quickly became something that could bring a lot of values.

One of the best thing about the new architecture was the discovery of SAS web server. When we realized that we had all the tools to create a platform where users could run SAS programs on demand, we started doing more research on how we could easily take advantage of these new features and implement a system that is relatively easy to use and maintain. We were interested in a platform where SAS programmers could develop and release on-demand reports while different users would be able to run them from a simple HTML front end.

The diagrams below explain the architecture at high level:

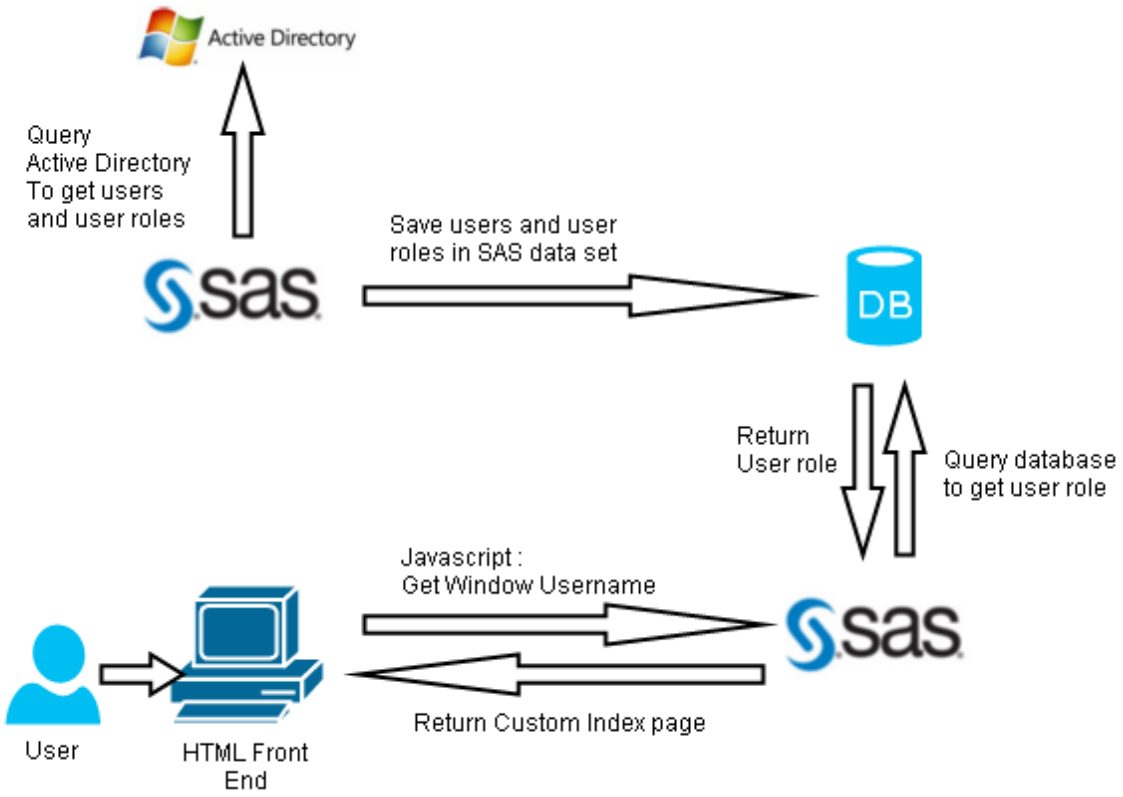


**Figure 1. High Level diagram for creating custom index page**

.

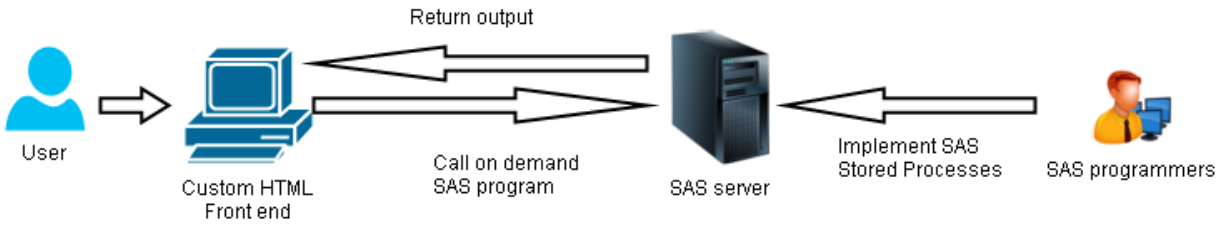


**Figure 2. High Level diagram for creating and running on-demand programs**

- Active directory is used to get information about users. It contains Windows username and role at the company for all the employees. We pull the information using a SAS program.
- The server is the central piece that executes all SAS programs. All the on-demand programs are saved as stored processes on the server. Any SAS programmers can implement on-demand reports
- The web browser is used by customers to connect to the front end and run SAS programs. It also allows the system administrator to manage the platform.
- JavaScript is an essential element to get the username of the user who is connecting to the platform

## CREATING CUSTOM FRONT-END PAGES BASED ON USERS

The reports that SAS programmers develop are most of the time specific to a small or big group of users in the company. They may contain sensitive information such as patient names for the customer service reports or complex data that may be misinterpreted by users who do not have enough knowledge. For that reason, when we developed this tool, one of the primary objective was to be able to easily customize the list of reports that a user could see and run. We could have registered all the users to the SAS metadata but we did not want to do all the maintenance. Plus we already have Active Directory for the management of users at the company level.

What we are doing is that we use the information already existing in active directory to determine the privileges of the different users. If you can get a username/password from IT, it is possible to query the active directory database directly from SAS and pull all the information about the users and groups existing in the system. We used the sample code for active directory import that you can find online and customize it to get the information we needed. The code is in the macro ***getuserad*** that we call passing the parameters below:

```
%getuserad(_filter=(memberOf=CN=Data Management,OU=Groups -
Distribution,DC=mycompany,DC=com),out=dm,indexpageid=8);

%getuserad(_filter=(memberOf=CN=Clinical Lab Quality,OU=Groups -
Security,DC=mycompany,DC=com),out=labQA,indexpageid=9);
```

- _filter: Filter to query active directory. It is used to get the information from an AD group or distribution list
- out: Name of SAS output data set
- indexpage: Indexpage is like a unique identifier for the group. It will be assigned to all the people who belong to that specific group

The macro creates one output data set per group of users. In the example above we created 2 data sets that contain the list of users who belong to the data management group and Clinical Lab Quality group. The data is saved in a SAS data set similar to the one below:

| keyid | name | title | displayname | indexuser |
|---|---|---|---|---|
| CN=John Doe,OU=Users - Employees,DC=mycompany,DC=com | jdoe | Sr Biomedical Data Analyst II | John Doe | 8 |
| CN=peter smith,OU=Users - Employees,DC=mycompany,DC=com | psmith | Prin Biomedical Data Manager | peter smith | 8 |

**Figure 3. Output SAS data set for users in the data management group in Active Directory**

Now that we have information about users and groups they belong to, we just need to find a way to display a customized index page, so only the reports of interest are shown. To do so, we use the Windows user name. Indeed, when you start your Windows session, you have to enter your username and password. This username is the same as the username that is saved in the active directory database (Column 'name' in the data set above). If we can get the Windows login, we can then find the user role in the SAS data set created with the macro ***getuserad.***

It is possible to use HTML and JavaScript in an Internet Explorer session to retrieve the user login without asking him to enter any information. We created the 'index.html' page below for that purpose. It is available at the following URL: http://myserver.com:8080/ondemand/index.html. This page is the entry point: Every user connects to the page to access the on-demand SAS reports. The system uses

JavaScript to get the Windows login and then pass this information to a SAS stored process.

**INDEX.HTML:**

```
<HTML>
  <head>
     <script type='text/javascript'
src='http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/jquery.min.js'></scri
pt>
     <script type='text/javascript'>
        function initInput()
            {

            var WinNetwork = new ActiveXObject('WScript.Network');
            var winuserid = WinNetwork.UserName;


      window.location.replace("http://myserver.com:8080/SASStoredProcess/do?_
action=execute,nobanner&_program=%2FBDC_Reports%2Fprod_reporting%2Fgenerate_i
ndex&_username=MySASuser&_password=%7BSAS002%7D4JUIUIGFGF&winuserid="+winuser
id)


            }

     </script>
  </head>
  <body onLoad='initInput()'>
  </body>
</HTML>
```

This HTML index page has two functions only:

- First, it gets the Window user name using activeX object and save it in the variable ‘winuserid’
- Then it calls a SAS stored process ‘generate_index’ that will create the custom index page. The ‘winuserid’ variable defined in the first step become a SAS macro variable

One thing to notice about this SAS program is that we have a specific SAS user ‘MySASuser’ to call SAS stored procedure. It is a generic user that was created specifically for the system. It is registered in SAS metadata and we use it every time we have to call a SAS stored process. The password is encoded in the URL. The advantage is that we do not need to register all the customers in the metadata.

The SAS stored process ‘generate_index’ generates the custom index page on the fly. This SAS program uses the ‘winuserid’ collected by the JavaScript method above to determine the user privileges. It queries the SAS databases created from Active Directory to find out the on-demand reports available for this type of user. Below is the SAS code for this program. In the first part, it extracts information about user and programs from the SAS data sets. The second part generates the custom HTML page based on the information found in the first step:

**GENERATE_INDEX.SAS:**

```
/*macro variable to save user ID*/
%global winuserid;
%let winuserid=&winuserid;

%macro genindex;
%let indexname=;
%let nbcat=0;
%let winuserid=%lowcase(&winuserid);

/*get index page for user*/
%let nuser=0;
data _null_;
      set dt.listuser;
      if name eq "&winuserid." then do;
            call symput('indexuser',strip(left(indexuser)));
            call symput('nuser',compress(_N_));
      end;
run;

%if &nuser ne 0 %then %do;
      proc sort data=dt.grouppage(where=(idindex=&indexuser.))
out=grouppage;by idgroup;run;
      proc sort data=dt.groups out=groups;by idgroup;run;

      data grouppage;
                  merge grouppage (in=a) groups;
                  by idgroup;
                  if a ;
      run;
      proc sort;by DESCENDING topcat idgroup;run;

      data grouppage;
                  set grouppage;
                  by DESCENDING topcat idgroup;
                  if first.topcat and topcat ne '' then istop=1;
```

```
      run;

      /*get list of group*/
      data _null_;
                  SET grouppage  end=eof;
                  call symput ('idgroup'||compress(_N_),compress(idgroup));
                  call symput
('groupname'||compress(_N_),strip(left(groupname)));
        call symput ('topcat'||compress(_N_),strip(left(topcat)));
        call symput ('istop'||compress(_N_),compress(istop));
                  if eof then call symput ('nbcat',compress(_N_));
      run;

      proc sql noprint;
            select distinct idgroup into:lsgroup separated by ","
            from grouppage;
      quit;



      /* prepare list prog by category*/
      proc sort data=dt.groupproglink(where=(idgroup in (&lsgroup.)))
out=groupproglink;by idprog;run;


      proc sort data=dt.listprogmeta out=progmeta(keep=idprog name location
description type);by idprog;run;


data groupproglink;
            merge groupproglink (in=a) progmeta;
            by idprog;
            if a;
            if description='' then description=name;


run;


      /*Get the list of program for each category*/
      %do i=1 %to &nbcat;
            %let nbprog&i=0;
            data _null_;
```

```
            SET groupproglink (where=( idgroup=&&idgroup&i.))  end=eof;
            call symput
('idprog'||compress(&i.)||'_'||compress(_N_),strip(left(idprog)));
            call symput
('namep'||compress(&i.)||'_'||compress(_N_),strip(left(name)));
            call symput
('locationp'||compress(&i.)||'_'||compress(_N_),strip(left(location)));
            call symput
('descriptionp'||compress(&i.)||'_'||compress(_N_),strip(left(description)));
                  call symput
('typep'||compress(&i.)||'_'||compress(_N_),strip(left(type)));
            if eof then call symput ('nbprog'||compress(&i.),compress(_N_));
            run;
      %end;


      data _null_;
            file _webout;
            put"<HTML>";
            put"<head>";
            put"<meta http-equiv='refresh' content='885'>";/*Refresh page
every 14 minutes to avoid disconnection*/
            put" <link rel='stylesheet' type='text/css'
href=""&urlserver./&spfolder./CSS/pagestyle.css"">";


            put"<script type='text/javascript'>";
            /*Function to verify the user name*/
            put"function initInput()";
            put" {";
            put"  var WinNetwork = new ActiveXObject('WScript.Network');";
            put"  var getlogged = WinNetwork.UserName.toLowerCase();";


            /* If AD user name is different from user name in the URL then we
reopen the entry index page*/
            put " if(!(getlogged == '&winuserid')){";
            put
"window.location.replace(""&urlserver./&spfolder./index.html"")";
            put "}}";
            put '</script>';
```

```
            put"<title>Index</title>";
            put"<style type='text/css'>";
            put "html, body { height: 100%; width:100%}";
            put" .wrapper {";
            put"  position: relative;";
            put"   height: 100%;";
            put"   width: 100%;";

            put"}";
            put".container-left {";
            put"   position: relative;"; /* Necessary to put this frame on
top of the other */
            put"   z-index: 1;";
            put"   width: 200px;";
            put"   height: 100%;";
            put "float:left;";

            put"}";
            put".container-right {";
            put"   position: absolute;";
            put"   top: 0;";
            put"  width: 86%;";
            put"   height: 100%;";
            /*put"  padding-left: 230px;";  */
            /* put"  box-sizing: border-box;";  */
            put"}";

            put"</style>";
            put"</head>";
            put"<body onLoad='initInput()'>";
            /* Top frame for image*/
            put"<iframe name='topframe'   frameborder='0' scrolling='auto'
style='width:100%;height:100px' marginwidth='0' marginheight='0'
src=""&urlserver./&spfolder./toppage2.html"" >";
            put"</iframe>";
            /* Divide main page with menu on the left and output on the
right*/
            put"<div class='wrapper'>";
            put"<div class='container-left'>";
```

```
                  put"<div id='menu' >";
                   %do i=1 %to &nbcat.;
                        %if &&istop&i. eq 1 %then %do; /*If there is a top
category and it is the first sub category*/
                              put"<h1>&&topcat&i.</h1>";
                              put"<h2>&&groupname&i.</h2>";
                        %end;
                        %else %if &&topcat&i ne  %then %do; /*If there is a
top category and it is not the first sub category*/
                              put"<h2>&&groupname&i.</h2>";
                        %end;
                        %else  %do;
                              put"<h1>&&groupname&i.</h1>";  /*If there is no
top category*/
                        %end;


                              %if &&nbprog&i. ne 0 %then %do; /*Get all the
programs for the category*/
                              %do j=1 %to &&nbprog&i.;
                               loc=urlencode("&&locationp&i._&j.");
                               pname=urlencode("&&namep&i._&j.");
                                    %if &&typep&i._&j. ne 2 %then %do;/* For
type 1 report*/
                                          put"<li><a
href='&urlpart1.&_program="loc +(-1)"%2F"pname +(-
1)'&_username=MySASuser&_password=%HKJH786767KJKSN'"' target='mainframe'
>&&descriptionp&i._&j.</a></li><br>";
                                    %end;
                              %else %if &&typep&i._&j. eq 2 %then %do;/* For
type 2 report*/
                                          put"<li><a
href=""&urlserver./&spfolder./&&namep&i._&j...html""
target='mainframe'   >&&descriptionp&i._&j.</a></li><br>";
                                    %end;
                              %end;
                        %end;
                        put"<BR> <BR>";
                        %end;


                   put"</div>";
              put"</div>";
```

```
            put"<div class='container-right'>";
            put" <iframe name='mainframe' style='height:
100%;width:100%'   frameborder='0' scrolling='auto'  marginwidth='5'
marginheight='5' scrolling='auto' >";
          put"</iframe>";
          put"</div>";
      put"</div>";
      put"</BODY>";
      put"</HTML>";

      run;
%end;
%mend;
%genindex;
quit;
```

Below is the page displayed by generate_index when a user from the Data Management group connect to the platform:

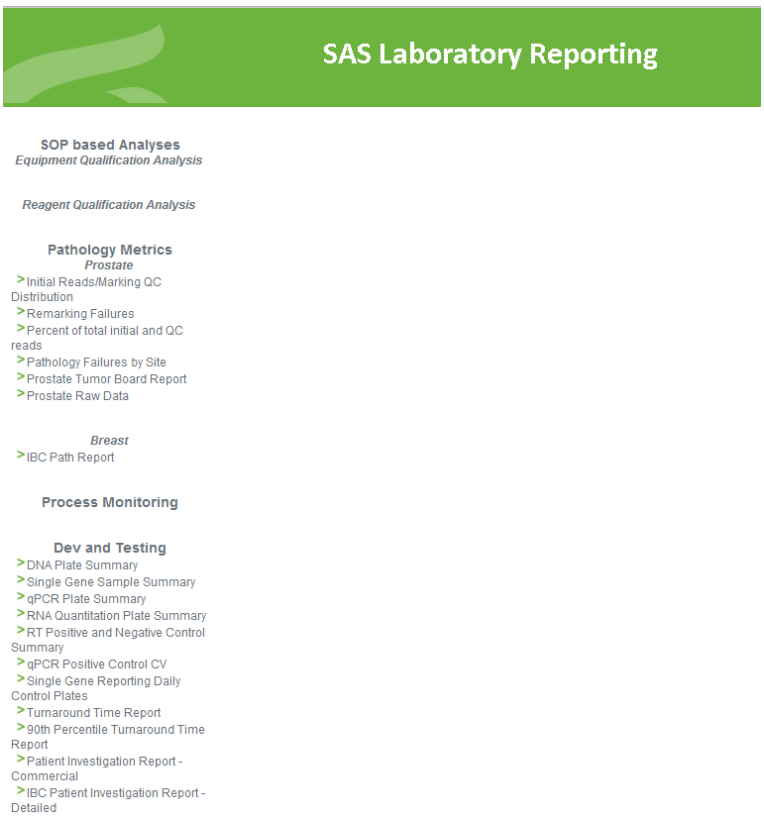


**Figure 4. Index page generated on the fly when a user from the data management group connect to the system**

Now if someone from the Clinical Lab Quality group connect to the same page, it will see something completely different since he only has access to a certain set of reports:

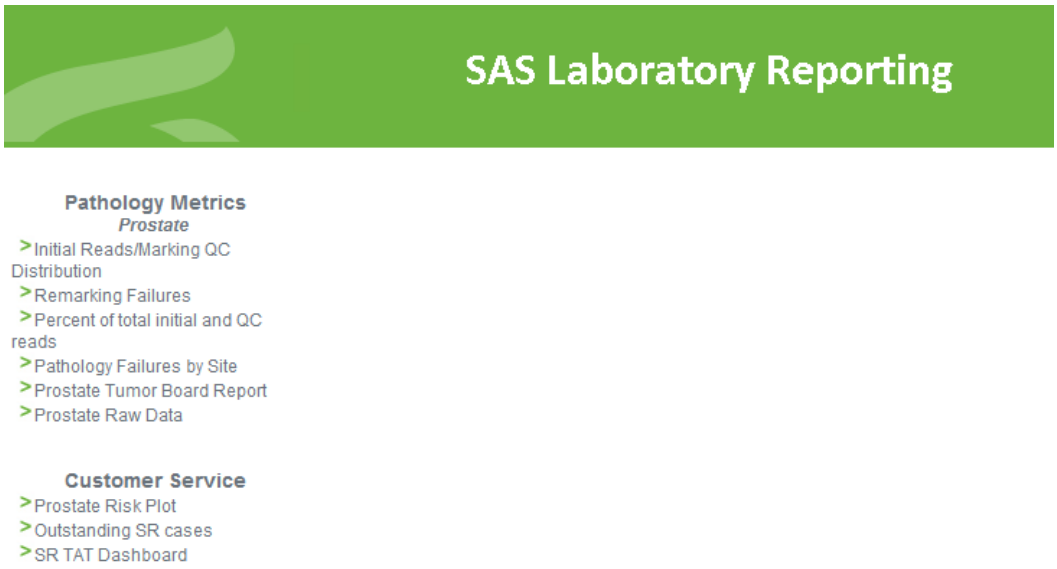


**Figure 5.** ***Index page generated on the fly when a user from the Clinical lab quality group connect to the system***

The index page always has the same layout. There is a menu on the left side, title at the top and the right side is used to display input forms and outputs. The only difference is the list of on-demand programs available to the user in the menu.

This index page does not physically exist on the drive. It is automatically created by the program genrate_index (see above) every time a user connect to the system. Additionally, there is no programming involved to display different type of index page. Everything is handled by the SAS programs. The only role of the administrator is to assign users to their specific groups.

## DEVELOPING PROGRAMS FOR THE ON-DEMAND REPORT PLATFORM

The on-demand programs are developed and deployed on the platform by SAS programmers. These programs need to follow a predefined architecture. It was a requirement to have a platform that can be used by many different programmers. However, it still provides a lot of flexibility, and they can decide if they want to develop something quickly or something a little more advanced.

All programs have to be registered as SAS stored processes on the server and they need to use a standard template that contains some predefined macros required by the system. There are different templates available depending on the program input/output. Below is an example of one of them. The programmer enters his code inside the macro 'myprog'. They start with this template when they want to make a program that creates an external file in PDF, Word or Excel format. The last macro %val_openfile_bottom will automatically open the file:

```
%let spfolder=dev_reporting;
```

```
%include "\\SASLabReporting\&spfolder.\admin\initServer.sas";

%LET DMStd2 = &pathtop.\&spfolder.\programs\;
%INCLUDE "&DMStd2.initSP.sas";

%LET ReportPath=&pathtop.\&spfolder.\outputs\;

/*************************************************************************
Enter a name for the report (it is used for the log file)

It should be the name of the external file that you create with this stored
process
*************************************************************************/
%LET ReportName=my_output_name.pdf ;

/*Specify the output type. Possible values are excel,pdf or rtf*/
%let _outtype=pdf;

proc printto log="&pathtop.\&spfolder.\logs\programs\&ReportName..log"  new;
run;

%valuser_openfile_top;

/*************************************************************************
 Enter the program code below
*************************************************************************/
/*Enter the code inside the macro and call the macro at the end*/
%macro myprog;

%mend;

%myprog;
/*************************************************************************
End program code
*************************************************************************/

/*Macro to open the final output*/
```

```
/* Important: the macro variable ReportName defined at the top of the
template*/

/* should include the file name and the extension*/

%val_openfile_bottom(path_f=%STR(&ReportPath),file_n=&ReportName,outtype=&_ou
ttype);
```

As explained above, the system is quite flexible and allows programmers to decide what type of input and output they want to implement in their programs.

**INPUT FORM**

Regarding the user front end, programmers can go two different ways:

- They can use prompts menu in enterprise guide when developing stored processes. It is the easiest way to do it; SAS will automatically create the form, and there is no HTML programming. The only drawback is that you are limited to the options available in SAS and you cannot format the page as you exactly want.

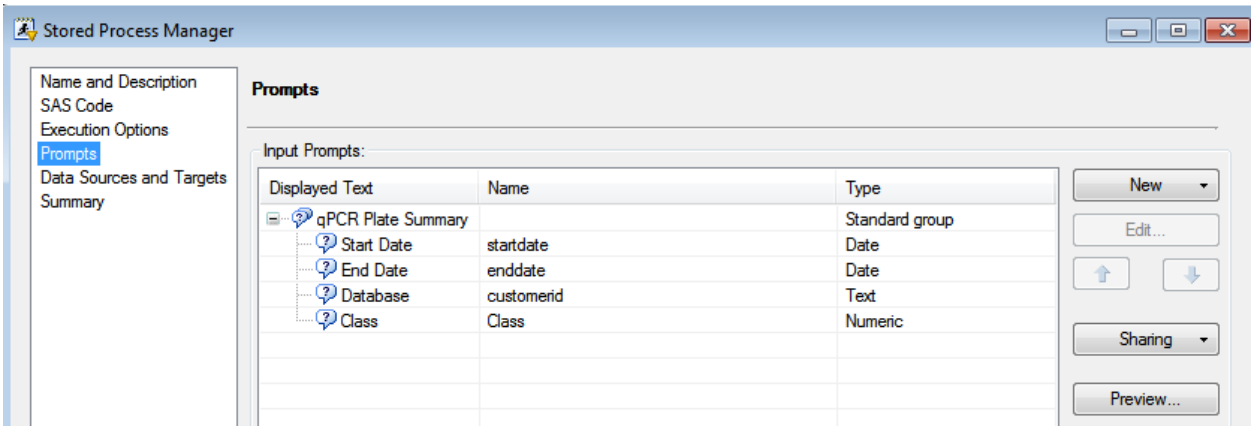


**Figure 6. Prompts menu in enterprise guide**

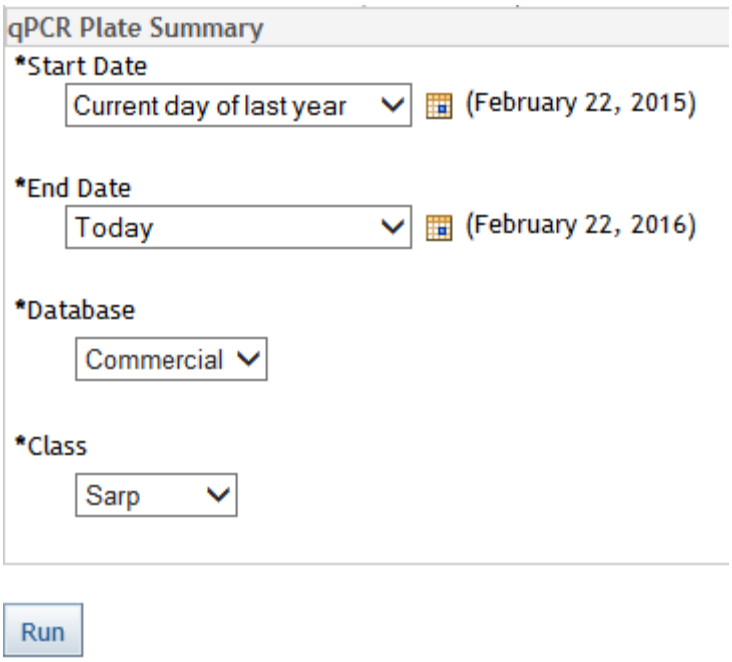


**Figure 7. HTML input form automatically created by SAS**

- The second option is to create an HTML page and use it as input form. It takes a little bit of programming but it gives you more freedom. It becomes possible to create the exact page you need. Additionally, you get access to all the features available to HTML programming such as JavaScript.

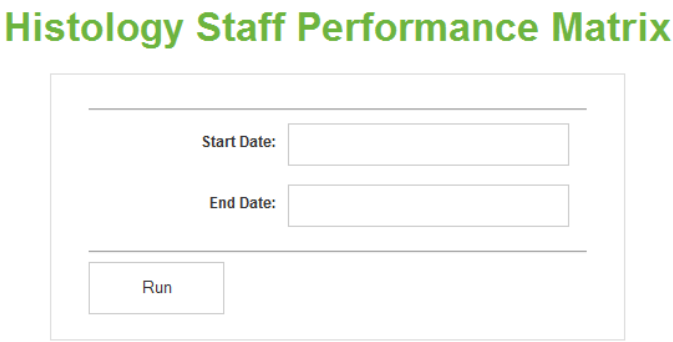


**Figure 8. Custom HTML page**

All the items on the input forms become SAS macro variables and are passed to the stored processes before execution. It can be a free text field, date field or any dropdown. It is also possible to create a form with upload feature. Users upload a file that becomes an input to the SAS programs.

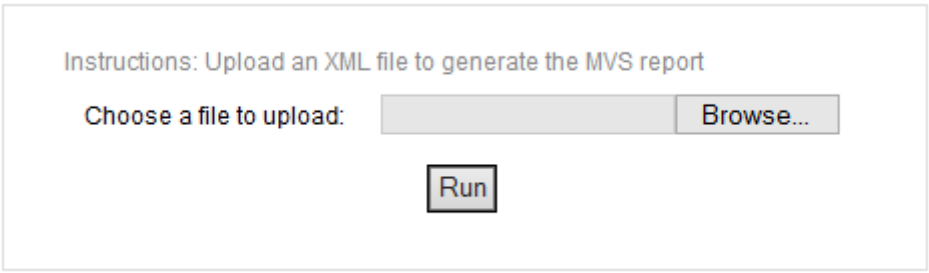


**Figure 9. Example input form with file upload**

### OUTPUTS

Programmers have different options for the output. They can opt to have their programs create an external file in PDF, Excel or Word format. After running the program, users will have the choice to open or save the file directly from the HTML page. The template provided to the programmers includes this feature, and they only need to make sure the name of their file is valid.

Another possibility is to write directly to the HTML page. The page is automatically refreshed and results are displayed in HTML format. It can easily be done using the SAS standard macros %stpbegin and %stpend.

## THE ADMINISTRATION SYSTEM

The system is administered using SAS stored processes that can be accessed through a web browser. We wanted a platform that did not really require any programming to manage users and programs.

There are 3 stored processes to administer the platform:

### USERS

This view lists all the users registered in the system. It is updated every time we pull data from Active Directory. From the web interface, you can see all the users, their role in the company and the group they belong to (figure 10, index page column) .

**List of users:**

| Name | Title | Index page | Manual Selection | edit |
|---|---|---|---|---|
| lpena | Clinical Lab Scientist | CLS | | edit |
| aford | Clinical Lab Scientist I | CLS | | edit |
| samclean | Clinical Lab Scientist I | CLS | | edit |
| arjapson | Clinical Lab Scientist II | CLS | | edit |
| jlomadilla | Clinical Lab Scientist II | CLS | | edit |

**Figure 10. Stored process to generate list of users view**

If you click on edit button, you can assign the user to a different group. He will then see a different list of on-demand program when he will connect to the system:

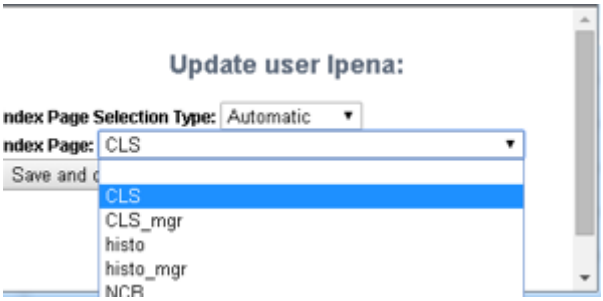


**Figure 11. View to edit a particular user and him assign to a group**

## INDEX PAGE VIEW

The index page view lists all the virtual index pages and the program categories that are displayed on the page when someone from this group connect to the system. Each user belongs to a group (group=virtual index page), and each page has different categories. A category is a submenu in the left side menu, and in each of them, you can put as many reports as you want to. From this page, you can easily modify the list of program categories which are available to the users. You can also create new virtual index pages or delete existing ones.

| Index Name | Categories | Edit | Delete |
|---|---|---|---|
| CLS | | Edit | Delete |
| CLS_mgr | | Edit | Delete |
| histo | Histology | Edit | Delete |
| histo_mgr | Histology<br>Histology - Management<br>Prostate | Edit | Delete |
| NCB | Process Monitoring<br>Breast | Edit | Delete |
| BDO | Equipment Qualification Analysis<br>Process Monitoring<br>Reagent Qualification Analysis<br>Dev and Testing<br>Histology<br>Histology - Management<br>Customer Service<br>Prostate<br>Service Engineering<br>Genomic Sciences<br>Accessioning<br>Breast | Edit | Delete |

**Figure 12. Index page view: Create virtual index pages for different user groups and assign program categories**

## CATEGORY VIEW

In this view, you can assign programs to categories. You can put as many programs as needed per category and you can also see on which virtual pages they are displayed. It is possible to add or delete a program to categories using the edit/delete buttons.

| Category | Top Category | List of Programs | List of Index Page | Edit | Delete |
|---|---|---|---|---|---|
| Equipment Qualification Analysis | SOP based Analyses | | BDO | Edit | Delete |
| Reagent Qualification Analysis | SOP based Analyses | | BDO | Edit | Delete |
| Dev and Testing | | GDNA_SUM<br>SGR_SUM<br>QPCR_SUM<br>RBG_SUM<br>RT_CNTRL<br>QPCR_POSCV<br>SGR-DCCP<br>COMOP_TurnaroundTime<br>COMOP_percentilesTAT<br>REQ_CHECK<br>REQ_CHECK_Breast | BDO | Edit | Delete |
| Histology | | histopathreport<br>breastcolontracking<br>prostatetracking | histo<br>histo_mgr<br>BDO | Edit | Delete |
| Histology - Management | | histomatrix | histo_mgr<br>BDO | Edit | Delete |

**Figure 13. Category page view: Programs are assigned to categories that are then displayed on the virtual index pages selected**

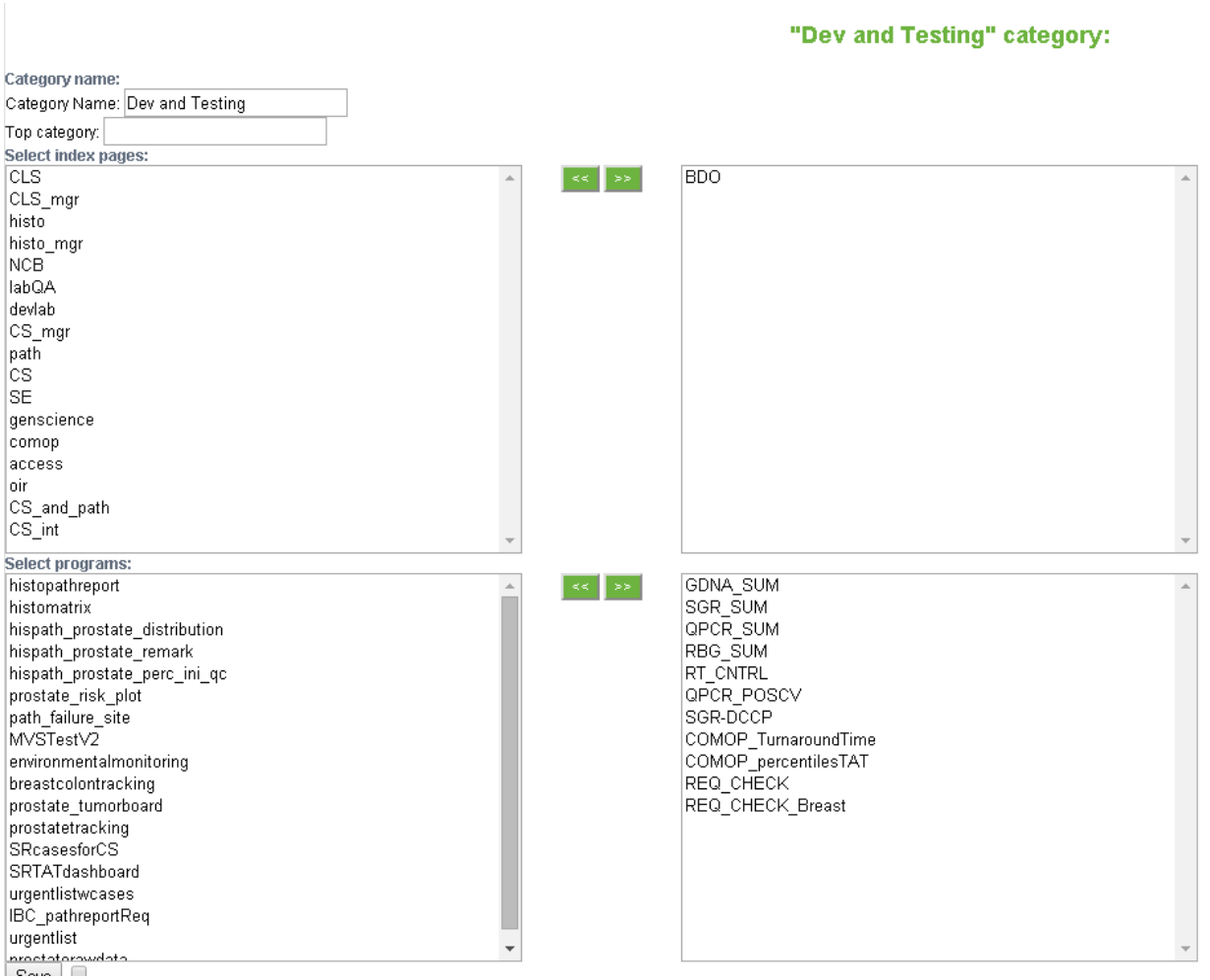


**Figure 14. View to edit a category: From this view you can select the programs you want to assign to a category. You also select which virtual index pages will display the category.**

## LOG SYSTEM

The platform includes a log system to keep track of the usage. It automatically collects relevant information such as the date, user and program name. Data is saved in a text file and is viewable by the system administrator. It helps identify the programs that are heavily used and also make sure that they are run only by users who have the right privileges.

```
03AUG2015:6:48:56,aadams,/BDC_Reports/prod_reporting/type1/urgentlistwcases
03AUG2015:6:50:00,tmallick,/BDC_Reports/prod_reporting/type2/IBC_pathreportReq
03AUG2015:6:56:13,aadams,/BDC_Reports/prod_reporting/type1/urgentlist
03AUG2015:8:27:40,zmathir,/BDC_Reports/prod_reporting/type2/IBC_pathreportReq
03AUG2015:8:30:32,zmathir,/BDC_Reports/prod_reporting/type2/IBC_pathreportReq
03AUG2015:8:35:18,dtom,/BDC_Reports/prod_reporting/type1/urgentlist
03AUG2015:8:35:55,dtom,/BDC_Reports/prod_reporting/type1/urgentlist
03AUG2015:8:49:13,dtom,/BDC_Reports/prod_reporting/type1/urgentlist
03AUG2015:9:18:09,aadams,/BDC_Reports/prod_reporting/type1/urgentlist
03AUG2015:9:37:51,aadams,/BDC_Reports/prod_reporting/type1/urgentlist
```

**Figure 15. Example of Log file**

The system also saves every single output generated on the platform as well as the corresponding SAS log files. It is possible to retrieve reports created months ago and it also help with troubleshooting.

## CONCLUSION

With some imagination and the right tools, it is possible to develop an on-demand web report platform so your customers can run their favorite SAS programs from the web browser. They enjoy being able to get their reports whenever they want.

For SAS programmers, it is also a great tool. They can release new on-demand programs and choose who is going to be able to run them.

Finally, the system administrator can manage the platform through SAS stored processes. He does not need any programming to maintain the system. Only point and click through the different menus.

## CONTACT INFORMATION

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

Romain Miralles
Genomic Health
Mrom34@gmail.com