

SAS Stored Processes on the Web - Building Blocks

Mark Roberts, Pinnacle Solutions, Indianapolis, IN

ABSTRACT

This paper explains, in a step-by-step format, the concepts, issues, techniques, and code needed to develop a web-based application from stored processes. Running an application on the web allows users to execute powerful SAS procedures without knowing SAS, without having SAS on their desktop, and without a SAS license. It also allows companies to distribute the application to many locations. If the application needs a change, it is fixed in one place, loaded to the SAS portal, and all locations see the corrected code at the same time. A SAS developer needs only an understanding of stored processes, prompts, macros, a little HTML, and a little knowledge of Proc Reports to achieve this.

The paper explains the concepts and techniques clearly via an example that runs consistently throughout. The code used for each concept is also included. The author put everything he learned developing his own application together in one place, something he couldn't find when he first started. One of his objectives for this paper is for the audience to be able to apply the code and successfully develop their own web-based applications with a minimum of difficulty and without spending hours searching the web and SAS documentation.

Introduction

A BI application, running on a web server, allows scientists and laboratory personal to run statistical analyses on data from laboratory test stands. The data is organized by lot. The results are put into a report in the form of tables (PROC TABULATE) and graphs (PROC GPLOT). Information about each report is put into a SQL Server® table. A second BI app reads this table and allows the scientists and laboratory personal to see the results of the analyses, to add comments to the reports, and to change the status of a report. Internal policy requires that each report be reviewed and released or rejected. This second application, called View/Manage Reports, is the subject of this paper.

SAS program modules were developed in SAS Enterprise Guide®, converted to stored processes, and executed in a SAS Portal. Users must enter a lot number in order to see reports associated with that lot. A user may, depending on the authorization level, add a comment to the report, view a report, or change the status of a report to 'Released' or 'Rejected'. The data is displayed to the user, almost exclusively, using PROC REPORT.

It is assumed that readers of this paper have used SAS Enterprise Guide® to create Stored Processes, including familiarity with prompts, that they have written macros and used macro variables, that they have some understanding of HTML, SAS ODS, and PROC REPORT. Tagsets are mentioned in this paper but the reader does not need to have any understanding of them.

Section 1 – Overview of Application from User’s Point of View

After a statistical analysis is run and a report is produced the user wants to see the reports and interact with them by:

- Viewing the contents of a report
- Adding a comment to a report
- Changing the status of a report from Pending to Released
- Changing the status of a report from Pending to Rejected

View/Manage Reports

Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report		Released
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report	Released / Rejected	Pending
ROB_972375564_marlr_23OCT12_14.52.02	Add Comment	View Report	Released / Rejected	Pending
ROB_9841020_nateb_23OCT12_16.09.38	Add Comment	View Report		Released
ROB_9841020_linow_24OCT12_16.22.12	Add Comment	View Report		Rejected
ROB_9841020_marlr_28OCT12_08.07.36	Add Comment	View Report		Released

Figure 1. Example of Displayed List of Reports – View/Manage Main Menu

If the user selects [Add Comments](#), they will see:

Type a comment (200 chars max)

Figure 2. Add Comments Screen

The user can type in a comment and select ‘Run’. The comments are saved and the user returns to the View/Manage Reports main menu.

If the user selects [View Report](#) they will see something like:

Mark V Study Lot 9841006, Level 82

						% of observations within	
Week	Condition	N	Mean	SD	Std to control 8	10 Units	15 Units
2	8	12	18.94	.06			
2	16	15	18.27	.44	-0.57	100	100
2	32	10	19.07	.27	0.03	100	100
2	64	12	19.75	.86	0.61	100	100
					Total:	100	100
3	8	12	17.69	.79			
3	16	12	17.44	.81	-0.15	100	100
3	32	12	18.56	.73	0.78	100	100
3	64	12	19.44	.89	1.45	100	100
					Total:	100	100

Figure 3. Example of Report

If the user decides to Release a report they would see:

Confirm Release

Confirm	User_Message	Cancel
Release	Confirm Release of LAB_4_Lot_779_07MAR2012_14_01	Cancel

Figure 4. Confirm Release Screen

If the user selects [Release](#), the report will be set as Released, which will be reflected in the View/Manage Reports main menu to which they will return.

If the user selects [Cancel](#), the app will return to the View/Manage Reports main menu and the report will not be set to Released.

This is a very simple representation of the application. It is much more complicated but this will be sufficient to illustrate the building blocks of 'SAS Stored Processes on the Web'

Section 2 - Building Blocks

1. Redirection via Direct HTML

The Challenge-

As a web application, allow a stored process (webpage) to redirect to another stored process (webpage).

One Solution-

The initial module that is run when a user executes the app has a prompt that requires the user to enter one or more lot numbers. The data (list of reports) is filtered by lot number(s). This 'controller' module does a number of other things, some of which will be discussed later. Control is then passed to the main menu. The following code, which is at the end of the 'controller' module, makes use of the special device '_webout' which is a SAS reserved filename reference used to output directly to the current HTML document. CARDS4 (or DATALINE4) is used because the HTML code contains semicolons.

```
DATA _null_;
  format infile $char256.;
  input;  infile = resolve(_infile_);  file _webout;
  put infile;
  cards4;
<html>
<head>
<title>This is a webpage title</title>
<meta http-equiv="refresh" content="4;
URL=http://robvm822.us.RobLabs.com:8080/SASStoredProcess/do?_program=/Applica
tions/Stored Processes/main_menu">
<meta name="keywords" content="automatic redirection">
</head>
<body>
This message will display for, in this case, 4 seconds then the app redirects
to main_menu
</body>
</html>
;;;
run;
```

Figure 5. Redirection With Hardcoding

The Challenge-

To gain flexibility by not hardcoding the server address and the folder path in the HTML.

One Solution-

All configurable parameters are defined in a file, which will be addressed later. For 'Figure 6. Redirection Without Hardcoding', the parameters are defined in %let statements so you can better observe the process.

```
%global url_string server_string server_type sp_folder_path prog_link;
%let server_string=http://robvm822.us.RobLabs.com;
%let server_type=8080/SASStoredProcess/do?_program=;
%let sp_folder_path=/Applications/Stored Processes/;
%let prog_link=main_menu;
data _null_;
  length url_string $400;
  url_string=cats("&server_string", '&server_type', "&sp_folder_path", "&
prog_link ");
  call symput('url_string', strip(url_string));
run;

data _null_;
  format infile $char256.;
  input;
  infile = resolve(_infile_); file _webout; put infile;
  cards4;
<html>
<head>
<title>This is a webpage title</title>
<meta http-equiv="refresh" content="2; URL=&url_string">
<meta name="keywords" content="automatic redirection">
</head>
<body>
This message will be display for, in the case, 2 seconds then the app
redirects to main_menu
</body>
</html>
;;; run;
```

Figure 6. Redirection Without Hardcoding

The Challenge-

When the stored process for 'Figure 6. Redirection Without Hardcoding' is run on a SASStoredProcess server, this error appears:

```
41      +
42      +* Redirect back to the main menu *;
43      +data _null_;
44      +  format infile $char256.;
45      +  input;
46      +  infile = resolve(_infile_);
47      +  file _webout;
48      +  put infile;
49      +  cards4;
```

ERROR: File is in use, _WEBOUT .

NOTE: The SAS System stopped processing this step because of errors.

Figure 7. Error – _WEBOUT In Use

One Solution-

One of the options when creating the stored process is 'Include code for'. You must deselect 'Stored process macros' as in the example below. If you allow the stored process to be built with stored process macros, SAS puts code before and after your code. This SAS-added code uses '_webout' so you can't use this device. When you deselect 'Stored process macros' SAS does not put this code around your code and you are free to use '_webout'.

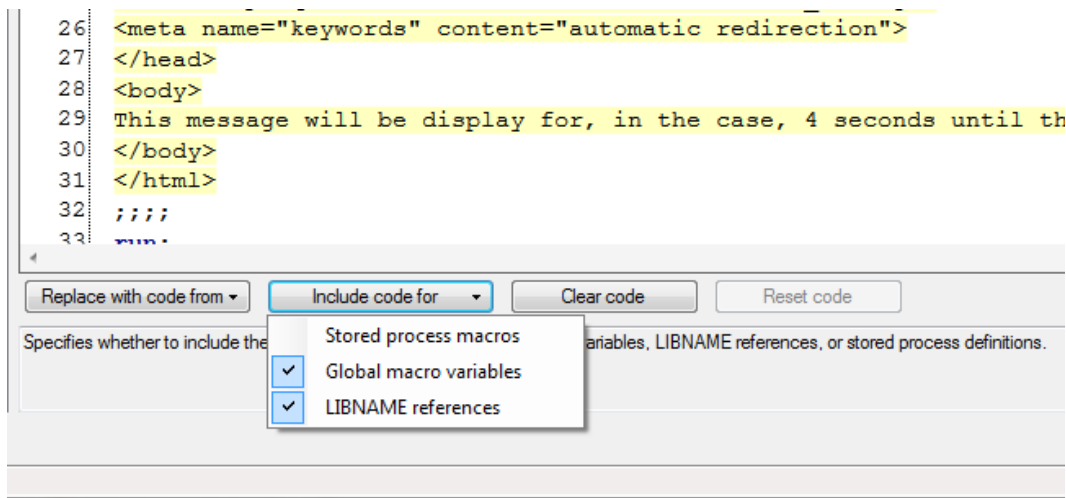


Figure 8. Eliminate Automatic Stored Process Macros

Note: Running this Stored Process in SAS Enterprise Guide® will result in an error, but the same Stored Process will work on the Portal. This is because SAS Enterprise Guide® does not understand _webout. This means, of course, that you must test your code on the SASStoredProcess server and/or the SAS Web Portal.

2. Redirection via User Selection

The Challenge-

Allow a user to select among several options which website to redirect to.

One Solution-

This is a completely different approach than redirection via direct HTML. You can put links on the screen that the user selects to drive the direction of the web app. For example, let's present the user with this screen:

View/Manage Reports

Report Name	Add Comment	View Report
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report
ROB_972375564_marikr_23OCT12_14.52.02	Add Comment	View Report
ROB_9841020_nateb_23OCT12_16.09.38	Add Comment	View Report
ROB_9841020_linow_24OCT12_16.22.12	Add Comment	View Report

Figure 9. Example of Displayed List of Reports

The user can select [View Report](#) to see a RTF version of the report. The user can select [Add Comment](#) to, well, add a comment that will get appended to the report when it is viewed.

This is accomplished with code that uses a tagset file, overwrites some ODS options, and then manually inserts the %stpbegin and %stpend macros. You must disable 'including the stored process macros' when registering this code as a stored process, as demonstrated in the previous example. If you do not disable this option, EG will automatically insert the macros which will cause duplicates and the stored process will fail.

The code used to display the View/Manage Reports screen is shown on the next page. Although I would normally structure the code better, I organized it so it would fit on one page. When the user selects [Add Comment](#), for example, the app executes the stored process 'add_comments' located at &sp_folder_path.

```

ODS PATH work.templat(update) sasuser.templat(read) sashelp.tmplmst(read);
%include 'C:\SASRoot\applications\utility\tableeditor.tpl';
ods escapechar="^";
%let _ODSDEST = tagsets.tableeditor;
%let _ODSOPTIONS= options (sort="yes" background_color="white"
    frozen_headers="yes"
    frozen_rowheaders="yes" load_msg="yes"
    load_img="c:\share\tableeditor\images\ajax-loader.gif"
    window_status='Reports' pagebreak="no" pagebreak_toggle="no");
%let _ODSSTYLE=seaside;
%stpbegin;

* Define the libname to the SQL Server Database *;
LIBNAME demo OLEDB ... ;

* Build the links *;
data work.reports_to_display;
    length View_Report $ 1000 Add_Comments $ 1000;
    set demo.reports;
    Add_Comments='^S={url="&_url?_program=&sp_folder_path.add_comments'
        || '&Report=' || strip(report_name) || ' "}Add Comment';
    View_Report='^S={url="&_url?_program=&sp_folder_path.view_report'
        || '&Report=' || strip(report_name) || ' "}View Report';
run;

* Write the data to the screen so the user can select*;
Title 'View/Manage Reports';
proc report data=work.reports_to_display split='*';
    col report_name add_comments view_report;
    define report_name / display "Report Name";
    define Add_Comments / display "Add*Comment";
    define view_report / display "View*Report";
run;

* Clear the library reference *;
libname demo clear;

%stpend;

```

Figure 10. Redirection via User Selection

3. Passing Values

The Challenge-

Since there is no 'global' concept in this type of app (this is, from stored process to stored process) values need to be passed from one stored process to another stored process. For example, in 'Figure 10. Redirection via User Selection', the 'add_comments' stored process needs to know which report to add comments to.

One Solution-

The HTML functionality of passing a parameter in a URL string is used to pass the value when the stored process is called. In this code snippet from 'Figure 10. Redirection via User Selection', you can see that the report name, contained in the reports table as SAS variable 'report_name', is passed to the add_comments stored process via "&Report=" || strip(report_name)":

```
data work.reports_to_display;
  length View_Report $ 1000 Add_Comments $ 1000;
  set demo.reports;
  Add_Comments=
    '^S={url="&_url?_program=&sp_folder_path.add_comments'
      || '&Report=' || strip(report_name) || '"}Add Comment';
```

Figure 11. Passing One Parameter

In the 'add_comments' stored process 'report' can be used as if it were a macro variable by referring to it as "&report". It is global to the 'add_comments' stored process but is not known outside 'add_comments' unless it is passed in the same manner.

You can pass more than one variable at a time with this technique:

```
Add_Comments =
  '^S={url="&_url?_program=&sp_folder_path.add_comments'
    || '&Report=' || strip(report_name)
    || '&Lot=' || strip(lot)
    || '&Current_Status=' || strip(current_status) || '"}Add Comment'
;
```

Figure 12. Passing More Than One Parameter

Using this code, 'report', 'lot', and 'current_status' in the 'add_comments' stored process can be used as if they were macro variables (&report, &lot, and ¤t_status).

4. Creating an Application Configuration File

The Challenge-

To make this app portable, values that might change should not be hardcoded. The server name, for example, will change from development to test to production environments.

One Solution-

Create a configuration file for this app. A SQL Server ® database table, to which all the stored processes in the app have access, is used. Note: It would be better to read these values from a text file instead of '%let' statements as in this example:

```
%global server_string report_path sp_folder_path;
%let server_string=http://robvm822.us.RobLabs.com;
%let report_path=\\dom.roblab.com\Projects\Development\Reports\;
%let sp_folder_path=/Applications/Stored Processes/;

libname demo oledb ...;
* Delete demo.config_table if it exists *; ...;
data work.prepare_data;
  length
    col_num      $ 1    server_string $ 29
    report_path $ 220   sp_folder_path $ 220
  ;
  col_num        = "1";
  server_string  = strip("&server_string");
  report_path    = strip("&report_path");
  sp_folder_path = strip("&sp_folder_path");
run;
proc sql noprint;
  create table demo.config_table
    (col_num      char (1) ,server_string  char(29)
    ,report_path  char(220) ,sp_folder_path char(220)
    )
  ;
  insert into u.config_table
  select col_num ,server_string ,report_path ,sp_folder_path
  from work.prepare_data
  ; quit;

libname demo clear;
```

Figure 13. Create Application Configuration File

5. Using Files to Pass Values

The Challenge-

Sometimes there are just too many variables to pass them in the manner described in '3. Passing Values'.

One Solution-

Create a file which is used to pass variables from stored process to stored process. Be aware, however, that there are times when you must pass the variable(s) in the manner described in '3. Passing Values'. For example, the report name in 'Figure 10. Redirection via User Selection' is not known until the user selects the corresponding [Add Comment](#) link. When variables are known ahead of time, however, I prefer to pass them via a file. I think that it is easier to read and write from a file than to pass a large number of variables in the manner shown in '3. Passing Values'. I also feel that it is cleaner code and, thus, is easier to maintain and modify. It is easier to see, at a glance, which variables are being used in the called stored process.

A SQL Server ® database table is used in this app. A table, called persist (as in 'persistent data') is created for each user and is used to pass as many variables as possible from stored process to stored process.

The name of this persist file starts with the user ID, obtained via "&sysuserid". This is done so that two users, both running the app at the same time, will not 'collide'.

The persist file is read by each module to obtain the data needed for that module to run successfully. Some modules, in turn, update the persist file.

6. Forcing a Stored Process to Reveal Its Prompt

The Challenge-

When 'Stored Process A' calls 'Stored Process B' and 'Stored Process B' has a prompt, 'Stored Process B' will not show its prompt if called in the manner so far demonstrated. Any reference to the prompt inside 'Stored Process B' will result in an error of an unresolved macro variable. If you have used prompts you realize that this is not true when you manually call the stored process from SAS Enterprise Guide® or on a server. This is only needed when a stored process is calling the stored process that has a prompt.

One Solution-

The Stored Process reserved macro variable '_action' is used to force 'Stored Process B' to show its prompt. The code added in 'Stored Process A' is '&_action=form,properties,execute,nobanner'. In this example, 'add_comments' has a prompt defined so it must be forced to show that prompt. When the user selects [Add Comments](#) this code is executed:

```
Add_Comments =
'^S={url="&_url?_program=&sp_folder_path.add_comments&_action=form,properties,execute,nobanner'
  || '&Report='           || strip(report_name)
  || '&Lot='              '|| strip(lot)
  || '&Current_Status='  || strip(current_status || ' ")}Add Comment'
;
```

Figure 14. Forcing a Stored Process to Reveal Its Prompt

7. Putting More Than One Link in a Cell

The Challenge-

Each report has a default status of 'Pending' when it is first created. The user wants to be able to change this status to [Released](#) or [Rejected](#). The screen should look like this:

View/Manage Reports

Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report		Released
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report	Released / Rejected	Pending
ROB_972375564_marlr_23OCT12_14.52.02	Add Comment	View Report	Released / Rejected	Pending
ROB_9841020_nateb_23OCT12_16.09.38	Add Comment	View Report		Released
ROB_9841020_linow_24OCT12_16.22.12	Add Comment	View Report		Rejected
ROB_9841020_marlr_28OCT12_08.07.36	Add Comment	View Report		Released

Figure 15. Example of Displayed List of Reports

In the 'Next Status' column there are two links, one if the user wants to reject the report and one if the user wants to release the report. Note that if the Current Status is 'Released' or 'Rejected' the user has no Next Status showing. That is the expected behavior. We will later see why.

One Solution-

The code to create this screen is shown on the next page. Remember to include the code that reads the tagset along with %stpbeg and %stpend as discussed in 'Redirection via User Selection'. Also, the libname statements are not shown.

```

data work.reports_to_display;
  length View_Report $ 1000 Add_Comments $ 1000 Next_Status $
1000 Current_Status $ 16;
  set demo.reports;
  Add_Comments=
    '^S={url="&_url?_program=&sp_folder_path.add_comments'
    || '&Report=' || strip(report_name) || ' "}Add Comment';
  View_Report='^S={url="&_url?_program=&sp_folder_path.view_re
port'
    || '&Report=' || strip(report_name) || ' "}View Report';
  Next_Status='';
  if upcase(status)='PENDING' then
    Next_Status=

    '^S={url="&_url?_program=&sp_folder_path.released_confirm'
      || '&Report='          || strip(report_name)
      || '&current_status=' || 'PENDING'
      || '&desired_status=' || 'RELEASED' || ' "}Released'
      || "^S={} / " ||
    '^S={url="&_url?_program=&sp_folder_path.rejected_confirm'
      || '&Report='          || strip(report_name)
      || '&current_status=' || 'PENDING'
      || '&desired_status=' || 'REJECTED' || ' "}Rejected'
    ;
  Current_Status = strip(status);
run;

Title 'View/Manage Reports';
proc report data=work.reports_to_display split='*';
  col report_name add_comments view_report next_status
current_status;
  define report_name      / display "Report Name";
  define Add_Comments    / display "Add*Comment";
  define view_report     / display "View*Report";
  define next_status     / display "Next*Status";
  define current_status  / display "Current*Status";
run;

```

Figure 16. Putting More Than One Link in a Cell

8. A Newline in a Cell with More Than 1 Link

The Challenge-

To save horizontal real estate the user wants each 'Next Status' to be on a separate line like this:

View/Manage Reports

Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report		Released
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report	Released Rejected	Pending
ROB_972375564_martr_23OCT12_14.52.02	Add Comment	View Report	Released Rejected	Pending
ROB_9841020_nateb_23OCT12_16.09.38	Add Comment	View Report		Released
ROB_9841020_linow_24OCT12_16.22.12	Add Comment	View Report		Rejected
ROB_9841020_martr_23OCT12_09.07.38	Add Comment	View Report		Released

Figure 17. Display List of Reports With a Newline in Cells

One Solution-

The code that links the two status values is "`^{\newline}`", as shown in Figure 21.

```
data work.reports_to_display;
  length View_Report $ 1000 Add_Comments $ 1000 Next_Status $ 1000
  Current_Status $ 16;
  set demo.reports;
  Add_Comments='{url="&_url?_program=&sp_folder_path.add_comments'
    || '&Report=' || strip(report_name) || ' "}'Add Comment';
  View_Report='{url="&_url?_program=&sp_folder_path.view_report'
    || '&Report=' || strip(report_name) || ' "}'View Report';
  Next_Status='';
  if upcase(status)='PENDING' then
    Next_Status=
      '^S={url="&_url?_program=&sp_folder_path.released_confirm'
        || '&Report=' || strip(report_name)
        || '&current_status=' || 'PENDING'
        || '&desired_status=' || 'RELEASED' || ' "}'Released'
        || "^\newline" ||
      '^S={url="&_url?_program=&sp_folder_path.rejected_confirm'
        || '&Report=' || strip(report_name)
        || '&current_status=' || 'PENDING'
        || '&desired_status=' || 'REJECTED' || ' "}'Rejected'
    ;
  Current_Status = strip(status);
run;
```

Figure 18. Code to Put a Newline in a Cell with More Than 1 Link

9. Roles, Permissions, and Authentication

The Challenge-

There can be several classes of users for an app, each having different privileges. For example, in this app there is a role called 'USER' and another role called 'SUPER USER'. 'USER' can change a report status from 'Pending' to 'Released' or 'Rejected' but, as you have seen, they cannot change any other status. The 'SUPER USER' will be able to change any status to any other status.

The roles are created and controlled by a BI Administrator. Individual users are usually put into groups. Executing an app can be restricted by groups. However, that is beyond the scope of this paper. The focus here will be on how permissions are used within an app. Note: _METAPERSON will only exist on Stored Process and Portal servers.

One Solution-

The role of the current user can be ascertained by using the following code:

```
data _null_ ;
  length GroupUri $256 type $60 id $17 GroupName $60 GroupList
$5000;
  call missing(GroupUri, type, id, GroupName);
  if symexist("_metaperson") then
clientname=trim(symget('_metaperson'));
  else clientname=trim(symget('sysuserid'));
  group_obj=cat("omsobj:IdentityGroup?IdentityGroup[MemberIdentities
/Person[@Name='",trim(clientname),'']]");
  groups=metadata_resolve(group_obj,type,id);
  GroupList='';
  if (groups >0) then do n=1 to groups;
    nobj=metadata_getnobj(group_obj,n,GroupUri);
    rc=metadata_getattr(GroupUri,"Name",GroupName);
    if substr(groupname,1,16)=labapp' then
GroupList=trim(left(GroupList)) || ' ' || trim(left(groupname)) ||
''';
  end;
  call symput('GroupList',trim(left(grouplist)));
  call symput('UserIDVal',trim(left(clientname)));
run;
```

Figure 19. Get the Role(s) for the Current User

The Challenge-

Once the user's role is known, permissions need to set within the code.

One Solution-

App-specific group names are written to the application configuration file. This, of course, is done once, when the app config file is created, well before the app is executed. 'lab_user' and 'lab_super' are the group names the BI Administrator created and populated with individual user log on IDs.

```
%let auth_group_user=lab_user;  
%let auth_group_super=lab_super;
```

Figure 20. Assign BI Group Name to Program Macro Variables

When the app runs and the role of the user is ascertained (previous page), a variable, created and named by programmer and called, in this case, 'user_auth', is set to an internal code for processing, like this:

```
%global user_auth;  
data _null_;  
  if "&auth_group_super" in (&GroupList) then  
    user_auth = 'SUPE';  
  else if "&auth_group_user" in (&GroupList) then  
    user_auth = 'USER';  
  else  
    user_auth = 'NONE';  
  call symput('user_auth', user_auth);  
run;
```

Figure 21. Create Internal Permissions

The value of 'user_auth' is added to the persist file so all stored process modules have access to it. 'user_auth' can now be used to control who can see what on the screen. The code to accomplish this is on the next page. Only pertinent code is shown.

```

%if &user_auth=SUPE %then %do;
  if upcase(status)='PENDING' then
    Next_Status=
      '^S={url="&_url?_program=&sp_folder_path.released_confirm'
        || '&Report='           || strip(report_name)
        || '&current_status=' || 'PENDING'
        || '&desired_status=' || 'RELEASED' || ' "}Released'
        || ""^S={} / " ||
      '^S={url="&_url?_program=&sp_folder_path.rejected_confirm'
        || '&Report='           || strip(report_name)
        || '&current_status=' || 'PENDING'
        || '&desired_status=' || 'REJECTED' || ' "}Rejected'
    ;
  else if upcase(status)='RELEASED' then
    Next_Status=
      '^S={url="&_url?_program=&sp_folder_path.pending_confirm'
        || '&Report='           || strip(report_name)
        || '&current_status=' || 'RELEASED'
        || '&desired_status=' || 'PENDING' || ' "}Pending'
        || ""^S={} / " ||
      '^S={url="&_url?_program=&sp_folder_path.rejected_confirm'
        || '&Report='           || strip(report_name)
        || '&current_status=' || 'RELEASED'
        || '&desired_status=' || 'REJECTED' || ' "}Rejected'
    ;
  else if upcase(status)='REJECTED' then
    Next_Status=
      (code left out for space reasons)
    ;
%end;
%else %if &user_auth=USER %then %do;
  if upcase(status)='PENDING' then
    Next_Status=
      (see Code Example 16 for this code)
%end;

```

Figure 22. Create the Functionality of the Webpage Based on User Role

Portions of the two screens (USER and SUPE) are shown on the next page.

USER Screen –

View/Manage Reports

Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report		Released
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report	Released / Rejected	Pending
ROB_972375564_markr_23OCT12_14.52.02	Add Comment	View Report	Released / Rejected	Pending
ROB_9841020_nateb_23OCT12_16.09.38	Add Comment	View Report		Released
ROB_9841020_linow_24OCT12_16.22.12	Add Comment	View Report		Rejected
ROB_9841020_markr_28OCT12_08.07.36	Add Comment	View Report		Released

Figure 23. Webpage for a User Role

SUPER USER Screen -

View/Manage Reports

Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report	Pending / Rejected	Released
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report	Released / Rejected	Pending
ROB_972375564_markr_23OCT12_14.52.02	Add Comment	View Report	Released / Rejected	Pending
ROB_9841020_nateb_23OCT12_16.09.38	Add Comment	View Report	Pending / Rejected	Released
ROB_9841020_linow_24OCT12_16.22.12	Add Comment	View Report	Pending / Released	Rejected
ROB_9841020_markr_28OCT12_08.07.36	Add Comment	View Report	Pending / Rejected	Released

Figure 24. Webpage for a Super User Role

10. Debugging

The Challenge-

Since running on a Stored Process or Web Portal server is another step removed from the programmer, a method is needed to debug the code if something goes wrong, 'and something always goes wrong'.

One Solution-

There are two techniques and two tools for debugging stored process code.

The first technique is to add debugging code directly to your code. This is accomplished with two tools. You are already familiar with the first tool:

```
options symbolgen mprint mlogic;
```

Figure 25. SAS Debug Options

To force the debugged output to show on the Stored Process server, the second tool, some HTML code ("&_debug=131"), is added to the call.

For 'Redirection via Direct HTML':

```
<meta http-equiv="refresh" content="4; URL=&url_string&_debug=131">
```

Figure 26. Adding '&_debug' to Direct HTML URL

For 'Redirection via User Selection':

```
Add_Comments='^S={url="&_url?_program=&sp_folder_path.add_comments&_debug=131'  
  || '&Report=' || strip(report_name) || ' " }Add Comment';
```

Figure 27. Adding '&_debug' to User Selection URL

Portions of the output are shown on the next page

```

debug = 131
grafloc = /sasweb/graph
htcook = __utma=1.1467885718.1345213347 ; __unam=631772-13!
htua = Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET C
program =
reqmeth = GET
rmtaddr =
rmthost =
srvname =
srvport = 8080
url = /SASStoredProcess/do
userlocale =
username = Mark Roberts
version = Version 9.3 (Build 474)

```

View/Manage Reports

Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report		Released
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report	Released Rejected	Pending
ROB_972375564_mark_23OCT12_14.52.02	Add Comment	View Report	Released Rejected	Pending
ROB_9841020_nateb_23OCT12_16.09.38	Add Comment	View Report		Released

SAS Log

```

1                               The SAS System                               11:28 Thursday, March 21, 2013

NOTE: Copyright (c) 2002-2010 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.3 (TS1M1)
      Licensed to
NOTE: This session is executing on the X64_S08R2 platform.

NOTE: Updated analytical products:
SAS/STAT 9.3_M1, SAS/ETS 9.3_M1, SAS/OR 9.3_M1

NOTE: SAS Initialization used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

NOTE: The autoexec file, e:\Apps\SAS93\config\Levl\SASApp\WorkspaceServer\autoexec.sas, was executed
      at server initialization.

>>> SAS Macro Variables:

__APSLIST=__url,__htua,__debug,__client,__htcook,__result,__grafloc,__program,__reqmeth,__rmtaddr,__rmthost,__sr
vname,__srvport,__version,__metauser,__username,__metafolder,__metaperson,__userlocale,__SECUREUSERNAME
__CLIENT=StoredProcessService 9.3;
__DEBUG=131
__GRAFLOC=/sasweb/graph
__HTCOOK=__utma=1.1467885718.
__unam=631772-13934f73263                JSESSIONID=D4A54F1A30294
__HTUA=Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR
2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; MS-RTC LM 8; .NET
4.0C; .NET4.0E)

```

Figure 28. Example of Debugging a Webpage via '&_debug=' Argument

The second technique is to run the stored process directly from the address bar of the browser. The fully-qualified path and stored process name are typed into the address bar and “&_debug=131” is added. This screen print illustrates the results of running the stored process, in “debug mode”, directly from the address bar. When you run a stored process on a Stored Process server you see the name in the address bar:

The screenshot shows a web browser window with the address bar containing the URL: `http://robvm822.us.RobLabs.com:8080/SASStoredProcess/do?sapfs_sessionid:..&_program=/Applications/Development/Stored Process/main_menu`. The browser tabs include "Web Slice Gallery", "Google", "ProMail - Login", "SAP NetWeaver Portal", and "SAS Training in the U.S.". The main content area is titled "View/Manage Reports" and displays a table of reports. On the left, there is a navigation tree with folders for "Development", "Prompts", "Stored Processes", "MyWebAppTest", and "Documentation".

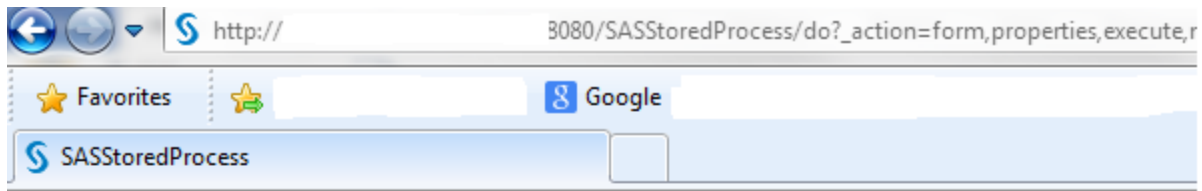
Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841006_ronann_23OCT12_16.12.052	Add Comment	View Report	Pending / Rejected	Released
ROB_9841006_ronann_23OCT12_16.12.052	Add Comment	View Report	Released / Rejected	Pending
ROB_972375664_mate_23OCT12_14.52.02	Add Comment	View Report	Released / Rejected	Pending
ROB_9841020_nate_23OCT12_16.09.38	Add Comment	View Report	Pending / Rejected	Released
ROB_9841020_inow_24OCT12_16.22.12	Add Comment	View Report	Pending / Rejected	Rejected
ROB_9841020_mate_28OCT12_08.07.36	Add Comment	View Report	Pending / Rejected	Released
ROB_9841020_valr_28OCT12_08.10.14	Add Comment	View Report	Pending / Rejected	Released
ROB_9841020_mate_28OCT12_16.52.14	Add Comment	View Report	Pending / Rejected	Released
ROB_9841020_inow_28OCT12_16.59.28	Add Comment	View Report	Pending / Rejected	Released
ROB_9841020_nate_28OCT12_18.58.18	Add Comment	View Report	Released / Rejected	Pending
ROB_972375664_nate_28OCT12_19.59.29	Add Comment	View Report	Released / Rejected	Pending

You simply add “&_debug=131” to the end of the address:

The screenshot shows the browser address bar with the URL: `http://robvm822.us.RobLabs.com:8080/SASStoredProcess/do?sapfs_sessionid:..&_program=/Applications/Development/Stored Process/main_menu&_debug=131`.

Figure 29. Example of Debugging via ‘&_debug=’ in the Address Bar

Select ‘Enter’ and you see:



>>> 0.001 Stored Process Input Parameters:

```
_debug = 131
_grafloc = /sasweb/graph
_htcook = __utma=1.1467885718.1345213347          __unam=631772-13934
_htua = Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2;
_program = /Applications/                      Development/Stored Processes/
_reqmeth = GET
_rmtaddr =
_rmthost =
_srvname =
_srvport = 8080
_url = /SASStoredProcess/do
_userlocale =
_username = Mark Roberts
_version = Version 9.3 (Build 474)
```

General

_debug

Run

Figure 30. Output from Adding '&_debug=' to the Address Bar

Select 'Run' and you see your regular output plus the SAS log.

11. Links in Titles

The Challenge-

The users requested filter and sort functionality. They want to filter the list in case too many reports are displayed for the Lot numbers they initially entered. They want to sort by several variables, one of which is Current Status. The Reset Filter functionality returns to the original list of reports.

One Solution-

The user will select one of the links in Title2 to execute this functionality:

View/Manage Reports
[Filter Reports](#) / [Reset Filter](#) / [Sort Reports](#)

Report Name	Add Comment	View Report	Next Status	Current Status
ROB_9841005_ronenh_22OCT12_15.12.052	Add Comment	View Report		Released
ROB_9841006_ronenh_22OCT12_15.12.052	Add Comment	View Report	Released / Rejected	Pending
ROB_972375564_marikr_23OCT12_14.52.02	Add Comment	View Report	Released / Rejected	Pending
ROB_9841008_ronenh_22OCT12_15.12.052	Add Comment	View Report		Released

Figure 31. Adding Links to Titles

The functionality was added in TITLE2 as follows:

```
title2
'^S={url="&_url?_program=&sp_folder_path.Filter&_action=form,properties,execute,nobanner"}Filter Reports'
'^S={} / "
'^S={url="&_url?_program=&sp_folder_path.Reset&_action=form,properties,execute,nobanner"}Reset Filter'
'^S={} / "
'^S={url="&_url?_program=&sp_folder_path.Sort&_action=form,properties,execute,nobanner"}Sort Reports'
;
```

Figure 32. Code for Links in Titles

12. Requiring the User to Log Back In

The Challenge-

One of the objectives of this project is to replace all the paper copies of the reports with electronic copies. Signatures are required on the paper reports and the users want to continue this by using electronic signatures. To do this, it is required that before a user can Release a report they must log back in. If they successfully log back in and the user ID matches the ID of the person who originally logged in then the report can be released. Also, their electronic signature, via their user ID, can be added to the signature page of the report.

One Solution-

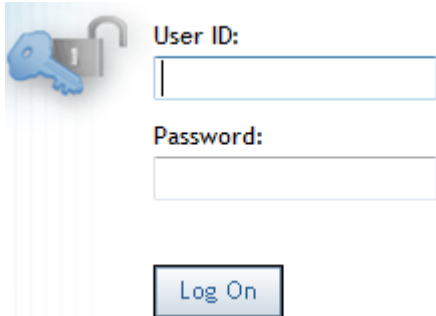
One of the services is SASLogin which is utilized for this requirement. Please note that the SAS code cannot obtain the password. This is as it should be. To do otherwise would create a security nightmare. However, if the user successfully logs in and their user ID matches the ID of the person who originally logged in then it is felt that this is sufficient authorization.

If the user cannot log back in then the system allows them to continue to try until the number of attempts established by the IT department is reached. At that point their ID is locked. If a person logs in but the user ID does not match the original ID then the person is not allowed to continue in the app.

For example, if the user selects [Release Report](#) we want to present these two screens:

Confirm Release

Confirm	User_Message	Cancel
Confirm Release	Confirm Release of Report	Cancel



The login form features a blue key icon on the left. It includes a 'User ID:' label above a text input field, a 'Password:' label above another text input field, and a 'Log On' button at the bottom.

Figure 33. Example of Screen to Require User to Log In

The code to accomplish this is on the next page.

```

data work.confirm_release;
  length Confirm $ 250 User_Message $ 200 Cancel $ 250;
  Confirm=

'^S={url="http://robvm822.us.RobLabs.com:8080/SASLogon/index.jsp?_program=/Ap
plications/Stored Processes/
Enter_password&_sasapp=Stored+Process+Web+App+9.3}Confirm Release';
  User_Message = "Confirm Release of Report";
  Cancel =
'^S={url="&_url?_program=&sp_folder_path.run_password_test"}Cancel';
run;
Title 'Confirm Release';
proc report;
  col confirm user_message cancel;
  define confirm      / display;
  define user_message / display;
  define cancel       / display;
run;

```

Figure 34. Confirm Release and Require Password

Section 3 – Nice to Know

1. Issue: HTML in a Macro

You can't put a Cards/Dataline statement a macro.

The following code:

```
%macro html_in_macro;  
  data _null_; format infile $char256.; input; infile = resolve(_infile_);  
  file _webout; put infile; cards4;  
  <html><head><title>This is a webpage title</title>  
  <meta http-equiv="refresh" content="4; URL=&url_string">  
  <meta name="keywords" content="automatic redirection">  
  </head><body>  
  Message  
  </body></html>  
  ;;;; run;  
%mend; %html_in_macro
```

Figure 35. Error – HTML in a Macro

results in this error message:

ERROR: The macro HTML_IN_MACRO generated CARDS (data lines) for the DATA step, which could cause incorrect results. The DATA step and the macro will stop executing.

Work Around-

Build the url_string with the desired next action in a macro then execute the 'data _null_' code outside the macro.

```
%macro work_around; %global url_string;  
  data _null_;  
    length url_string $400;  
    url_string=cats("&server_string", "&server_type",  
"&sp_folder_path", "&prog_link");  
    call symput('url_string', strip(url_string));  
  run;  
%mend;  
%work_around;  
data _null_; format infile $char256.; input;  
  infile = resolve(_infile_); file _webout; put infile;  
  cards4;  
  <html><head><title>This is a webpage title</title>  
  <meta http-equiv="refresh" content="2; URL=&url_string">  
  <meta name="keywords" content="automatic redirection">  
  </head><<body> Message </body>  
  </html>  
  ;;;; run;
```

Figure 36. Error – HTML in a Macro

2. Issue: SAS Warnings

SAS generates a warning message for `&_action` because SAS thinks it is a SAS macro variable when, in fact, it is meant to part of the HTML code.

WARNING: Apparent symbolic reference `_ACTION` not resolved.

It is just an annoyance and does not cause the program to fail.

Work Around-

The following will eliminate the warning:

```
%let _action=%nrstr(&_action);
```

This works for any HTML value and also for SAS values passed to another stored process.

3. Issue: Allow the User to Confirm an Action

The user would like to see the following if, for example, they select [Release](#):

Confirm Release

Confirm	User_Message	Cancel
Release	Confirm Release of LAB_4_Lot_779_07MAR2012_14_01	Cancel

Figure 37. Confirm Release Screen

The following code will allow a user to either confirm that they want to release the report or to cancel and return to the main menu:

```
data work.confirm_release;
  length Confirm $ 250 User_Message $ 200 Cancel $ 250
Report_Name $ 50;
  report_name = "&report_name";
  Confirm=
    '^S={url="&_url?_program=&sp_folder_path.Release' ||
    '&report_id=' || strip(report_name) || ' " }Release'
  ;
  User_Message = "Confirm Release of &report_name";
  Cancel =
'^S={url="&_url?_program=&sp_folder_path.main_menu"}Cancel';
run;
Title 'Confirm Release';
proc report data=work.confirm_release;
  col confirm user_message cancel;
  define confirm / display;
  define user_message / display;
  define cancel / display;
run;
```

Figure 38. Code to Build and Display Confirm Release Screen

CONCLUSION

Putting the power of SAS on the Web takes some planning but these building blocks are easy for an experienced SAS programmer to grasp. This helps to make the first steps into this area less intimidating. Using these building blocks will allow you to more quickly and accurately create excellent web apps. Of course, as you work with these tools you will find other, and better, ways to apply them as well as new tools. Please share what you find.

REFERENCES

STORED PROCESSES

- The 50 Keys to Learning SAS Stored Processes
Tricia Aanderud and Angela Hall
First Edition, 2012, Siamese Publishing
- A SAS® Programmer's Guide to Stored Processes
Joe Flynn
<http://support.sas.com/resources/papers/proceedings11/050-2011.pdf>
- Unleashing the power behind Stored Processes
Ian Amaranayake
http://www.amadeus.co.uk/_assets/files/unleashing-the-power-behind-stored-processes.pdf
- ODS Options and SAS® Stored Processes
Cynthia L. Zender
<http://www2.sas.com/proceedings/forum2007/021-2007.pdf>
- SAS Stored Processes Reserved Macro Variables
http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/reserved.html
- Macro Variables That Are Generated from Prompts
<http://support.sas.com/documentation/cdl/en/stpug/62758/HTML/default/viewer.htm#n1x5bwm15z6zcmn1jjzrschl4z90.htm>
- SAS® 9.3 Stored Processes Developer's Guide
<http://support.sas.com/documentation/cdl/en/stpug/62758/PDF/default/stpug.pdf>

ODS

- Using ODS Options with SAS Stored Processes
Cynthia L. Zender
http://dc-sug.org/zqj_ODS_Presentations_DCSUG/dcsug_ods_options_stored_processes.pdf
- SAS ODS Style Examples
http://stat.lsu.edu/SAS_ODS_styles/SAS_ODS_styles.htm
- ODS PDF “Tips and Tricks
Barry Hong
http://www.sas.com/offices/NA/canada/downloads/presentations/ghsug_Spring2007/ODS_PDF_Barry.pdf
- PDF Can be Pretty Darn Fancy (ODS and PDF)
Pete Lund
<http://www2.sas.com/proceedings/forum2008/033-2008.pdf>
- ODS and RTF
http://support.sas.com/rnd/base/ods/templateFAQ/Template_rtf.html
- SAS9 ODS Tip Sheet
<http://support.sas.com/rnd/base/ods/scratch/ods-tips.pdf>
- Using RTF codes in ODS RTF outputs
Rachabattula, Sriharsha
<http://www.nesug.org/Proceedings/nesug10/po/po40.pdf>

ODS ESCAPECHAR

- The Great Escape(char) Redux
Louise S. Hadden
<http://www.nesug.org/proceedings/nesug08/np/np10.pdf>
- Enhancing RTF Output with RTF Control Words and In-Line Formatting
Lori S. Parsons
<http://www2.sas.com/proceedings/forum2007/151-2007.pdf>
- ODS ESCAPECHAR Statement
<http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/viewer.htm#a002233270.htm>

ACKNOWLEDGMENTS

My grateful thanks to the following for their valuable help with this paper.

DJ Penix

Dave Foster

Philip Fowler

Nader Afshar

CONTACT INFORMATION

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

Mark Roberts:
Pinnacle Solutions
426 E New York St
Indianapolis, IN 46202
317-440-6534
mark.roberts@psiconsultants.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.