Optional and Multi-Select Parameters with SAS/IntrNet®
Craig Austin, Frank Russell Company, Tacoma, WA

ABSTRACT
When building SAS Applications to be deployed to the web with SAS/IntrNet, it's often desirable or even necessary to allow optional parameters. When selected, those parameters are passed to a SAS program via macro variables through the SAS/IntrNet Broker Common Gateway Interface (CGI). If the SAS program tries to reference a macro variable that was not passed through the CGI, the SAS program may end with an error. This paper will discuss the use of a SAS Macro developed primarily for dealing with optional parameters. The macro also includes functionality for efficiently dealing with multi-select parameters. Both macro code and proper use of the macro will be demonstrated. This paper is intended for SAS/IntrNet developers of all skill levels who have not already found a simple solution for these common problems.

INTRODUCTION
Throughout this discussion the HTML SELECT code shown in exhibit 1 will be used to demonstrate the problem and the macro solution. The HTML code contains a select box with three options for country (Canada, Germany, and USA). In reality, I would not encourage anyone to hard code select option values in an HTML page. Instead, you should use htmSQL, JSP, or an application server to query the actual values from your SAS dataset so that your web pages change dynamically with the data. However, each of those subjects could be a SUGI presentation of its own.

EXHIBIT 1: HTML CODE SNIPPET
<html>
<head>
<title>Multi-Select Demo</title>
</head>
<body>
<form action="http://localhost/cgi-bin/broker.exe?" method="post">
<input type="Hidden" name="_program" value="web.sugidemo.sas">
<table align="center"><tr align="center"><td>
<br><br><font color="Blue" size="+2">
Select Box Demo</font>
</td></tr>
<tr align="center"><td><br>Country</td></tr>
<tr align="center"><td><select multiple name="country" size="3">
<option value="CANADA">Canada</option>
<option value="GERMANY">Germany</option>
<option value="U.S.A.">USA</option>
</select>
</td></tr>
</table>
<input type="Checkbox" name="_debug" value=131>Debug Mode
</form>
</body>
</html>

The HTML code produces the select box shown in exhibit 2. Notice that the country selection is optional. If the user selects a country, the country value selected will be passed to our SAS program as a macro variable named country. If the user does NOT select a country, then no macro variable is passed to our SAS application.

EXHIBIT 2: WEB PAGE

In the HTML code of exhibit 1 is a hidden form variable named _program. The _program variable tells our broker CGI which program to run. In this case, the program we'll use is sugidemo1.sas which is shown is exhibit 3.

EXHIBIT 3: SUGIDEMO1.SAS
ods html file="webout";
proc report data=sashelp.prdsale;
title1 'Sales Report';
where country = "&country";
col country year actual;
define country /group;
define year /group;
define actual /analysis;
rw;
ods html close;

If we select a single country (like Canada) from the web page, the broker CGI intercepts the country variable, converts it to a macro variable, and sends it to our SAS program. The SAS program runs without error and displays the results in exhibit 4.
EXHIBIT 4: RESULTS OF SINGLE SELECT WITH SUGIDEMO1.SAS

Sales Report

<table>
<thead>
<tr>
<th>Country</th>
<th>Year</th>
<th>Actual Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANADA</td>
<td>1993</td>
<td>$121,020.00</td>
</tr>
<tr>
<td></td>
<td>1994</td>
<td>$125,970.00</td>
</tr>
</tbody>
</table>

However, if we don’t select a country (after all, the web page tells us we don’t have to select one), then our program returns nothing but a blank screen. If we turn on debug and view the SAS log, we see in exhibit 5, that the “Apparent symbolic reference COUNTRY is not resolved.” Since the macro variable is not resolved, the program passes the value ”&country” to the where clause and we get no results because there are no records in SASHELP.PRDSALE where country = ”&country”.

EXHIBIT 5: SAS LOG

NOTE: running request program web.sugidem01.sas
NOTE: %INCLUDE (level 1) file /home/sasapps/web/sugidem01.sas
2 +ods html file=_webout;
NOTE: Writing HTML Body file: _WEBOUT
3 +proc report data=sashelp.prdsale;
4 + title1 'Sales Report';
5 + where country = ”&country”;
WARNING: Apparent symbolic reference COUNTRY not resolved.
6 + col country year actual;
7 + define country /group;
8 + define year /group;
9 + define actual /analysis;
run;
NOTE: No observations in input dataset.
NOTE: There were 0 observations read from the data set SASHELP.PRDSALE. WHERE country= ”&country”;

To fix the problem, we think simple: just write a macro to test whether the COUNTRY macro variable exists and add the where clause only when it does. So how do we test for the macro variable? There is no %isdefined function in SAS, so a couple other ideas are to test the length of the resolved macro variable, or test to see if macro variable is equal to itself unresolved. That, in turn enables our %length trick previously discussed to work as desired. Now I can add a simple macro to my sugidemo1.sas program (shown in exhibit 8, sugidemo2.sas), and if no country is selected, the program works just fine (results in exhibit 9).

EXHIBIT 6: SASLOG SHOWING ERROR WITH %NRSTR()

2 +if &country ne %NRSTR(&country) +then
3 $do;
4 where country = ”&country”;
5 +end;
6 +mend;
7 +data junk;
8 +set sashelp.prdsale;
9 +run;
WARNING: Apparent symbolic reference COUNTRY not resolved.
ERROR: A character operand was found in the %IF condition where a numeric operand is required. The condition was: &country ne &country
ERROR: The macro TEST will stop executing.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.JUNK may be incomplete. When this step was stopped there were 0 observations and 10 variables

So, to resolve the problem, I’ll use the view SASHELP.VMACRO. SASHELP.VMACRO is one of many metadata (data about data) views provided automatically in SAS. We’ll use the view to determine whether the macro variable exists. And, since this is a section of code that I’ll use over and over again, I’ll place it in a macro named MVEXIST. The code for MVEXIST is shown in exhibit 7.

EXHIBIT 7: MVEXIST MACRO (VERSION 1)

%macro mvexist(mvarname);
%global &mvarname;
proc sql noprint;
select count(*) into :mvcntval
from sashelp.vmacro
where name = ”%upcase(&mvarname)”;
quit;
%if &mvcntval = 0 %then %let &mvarname=;
%mend;

So far, the only thing the macro does, is test to see if the macro variable exists. If it does exist, the macro does nothing. If it doesn’t exist, the macro creates a variable of the same name with a null value. That, in turn enables our %length trick previously discussed to work as desired. Now I can add a simple macro to my sugidemo1.sas program (shown in exhibit 8, sugidemo2.sas), and if no country is selected, the program works just fine (results in exhibit 9).

EXHIBIT8: SUGIDEMO2.SAS

%mvexist(country);
%macro wherecl;
  %if %length(&country) > 0 %then $do;
  where country = ”&country”;
  %end;
%mend;
ods html file=_webout;
proc report data=sashelp.prdsale;
title1 'Sales Report';
%wherecl;
  col country year actual;
  define country /group;
  define year /group;
  define actual /analysis;
run;
ods html close;
So now that that works, what do we do about multi-selects? What happens if the user selects more than one country? Well if we turn on debug, we can see in the partial SAS log of exhibit 10 that each country is passed as a separate macro variable. If we select two countries (Canada and Germany), the original macro variable &country is equal to the first selection “Canada” but now we have three other macro variables: &country0, &country1, and &country2. &country0 contains the number of country variables passed, while &country1 and &country2 contain the value of the first and second countries. The number of variables will always be equal to the number of selections plus two (one with the original name of the variable with a value equal to the first selection and another with zero appended to the name and a value equal to the number of selections made.)

**EXHIBIT 11: MVEXIST MACRO (FINAL VERSION)**

```sas
%mvexist(mvarname);
%global &mvarname;
proc sql noprint;
select count(*) into :mvcntval from sashelp.vmacro
where name = "%upcase(&mvarname)";
quit;
%if &mvcntval = 0 %then %let &mvarname=;
%else %do;
proc sql noprint;
select count(*) into :mvcntval from sashelp.vmacro
where name = "%upcase(&mvarname.0)";
quit;
%if &mvcntval = 1 %then
%do i = 1 to &mvarname.0;
%if &i > 1 %then %let &mvarname = &mvarname&i;
%end;
%end;
%mend;
```

**EXHIBIT 12: ADDQUOTES MACRO**

```sas
%macro addquotes(mvarname);
%let i=1;
%let acctlist=;
%let acctv=%scan(%quote(&&mvarname),&i,%str(,%', ));
%do %while(%length(&acctv) > 0);
%if &i > 1 %then %let acctlist=
%trim(%quote(&acctlist)),;
%let acctlist=%trim(%quote(&acctlist)) %str(%')%trim(&acctv)%str(%');
%let i=%eval(&i+1);
%let acctv=%scan(%quote(&&mvarname),&i,%str(,%', ));
%end;
%let &mvarname = %trim(%quote(&acctlist));
%mend addquotes;
```

**EXHIBIT 13: SUGIDEMO.SAS (FINAL VERSION)**

```sas
%mvexist(country);
%addquotes(country);
%macro bldwhere;
%if %length(&country) > 0 %then %do;
%let &country = %trim({quote(&country)});
%end;
%mend;
ods html file=_webout;
proc report data=sashelp.prdsale;
title1 'Sales Report';
%bldwhere;
col country year actual;
define country /group;
define year /group;
define actual /analysis;
run;
ods html close;
```

The final SAS program is shown in exhibit 13. Now if we select more than one country, like Canada and Germany, the program works without a hitch.
CONCLUSION
The MVEXIST macro and ADDQUOTES macro shown in exhibits 11 and 12 are two macros that I use for nearly every web application that I build using SAS/IntrNet. The MVEXIST macro is the primary topic of this paper and is an easy solution for dealing with optional and multi-select parameters with SAS/IntrNet. The ADDQUOTES macro provides a means for quoting delimited text in a macro variable. It comes in handy when building multi-select where clauses for character variables. I hope that you will find both of these macros useful.

ACKNOWLEDGMENTS
The author would like to express his appreciation to the following people for their assistance with this paper:

Jim Acker, Frank Russell Company

TRADEMARK CITATION
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 registered trademarks of their respective companies.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. You may contact the author at:
Craig Austin
Frank Russell Company
909 A Street
Tacoma WA 98402-5120
(253) 573-4709
Fax: (253) 502-4299
Email: caustin@russell.com