A Dynamic SAS/IntrNet Example with Help of ASP and JavaScript: Online Clinical Trial Randomization List Generator

Sijian Zhang¹, Mark Carpenter², Michael Windle¹
1. University of Alabama at Birmingham, AL
2. University of Alabama, AL

Abstract
In programming for interactive SAS® Internet services, there are two important issues that programmers have to deal with. One is dynamically passing the parameters from web page to SAS program; another is the validation of the input values. SAS macro programming can handle the first one very well. However, for the validation issue, although it can be accomplished by SAS/IntrNet® solely, the programming in other languages can be easier and more efficient. By taking a practical example of an interactive SAS/IntrNet application, the Online Clinical Trial Randomization List Generator, we demonstrate the methods and skills of how to use SAS macro programs to handle the parameter passing dynamically, and how to integrate ASP and JavaScript with SAS/IntrNet tasks to make the programming simple and enhance the performance efficiency of the SAS/IntrNet applications.

Keywords: SAS/IntrNet, ASP, JavaScript, clinical trial, randomization, dynamic, efficiency

Introduction
SAS/IntrNet is the component for SAS to vender its output to the Internet. Since Internet is now very popular, and the accessibility is not a problem for ordinary people, the advantages of IntrNet services become obvious. On the end user side, they need not install the SAS software on their computers, need not to understand the SAS programming, and even need not to aware what they get on their browsers are generated from SAS programs; and on the programmer side, it is easy to maintain the applications and powerful to offer a service to multiple users at the same time. When the web service is dynamic, it will involve an interactive process in which end users will enter certain parameters to get the customized results. However, if the values of the parameters are invalid or out of range, the SAS program may stop due to error or generate incorrect results. To avoid this kind of mistakes, close ended style data entry and validation checking are commonly used. Since IntrNet works only on the server side, it is not efficient to handle some of these, especially if there involve many interactive steps that the validation for each step depends on the inputs of previous entries. It means many communications between the browser and the server. If some of the validations can be done at the browser, it will alleviate the burden on the server and make services more efficient. In terms of programming for this purpose, other commonly used user and server side programs are easier and more efficient than IntrNet. This article uses an example to demonstrate how to build an efficient SAS/IntrNet application with help of ASP and JavaScript.

Online Clinical Trial Randomization List Generator
First, let’s take a look at this dynamic SAS/IntrNet application. One thing that must be ready before starting a clinical trial is the randomization list. The Online Clinical Trial Randomization List Generator can produce the randomization lists online given the values of some parameters, such as total sample size, names of treatments, number of treatments, ratio of treatments, whether it is blocked, block size if blocked, whether it is by study site, number of sites if by study site, sample size for each site, seed, etc.

For a simple instance: total sample size = 20, two treatments (drug : placebo = 1:1), blocked (block size = 4), randomized by site, 2 study sites, sample size for Site 1 is 8, sample size for Site 2 is 12, and seed = 15273.

Figure 1 shows the page that collects the basic information for the randomization. Based on the number of treatments and number of study sites, two tables dynamically appear on the following page (see Figure 2). And on the same page, the block size options will be changed dynamically according to the treatment ratio. After pressing Submit button on Figure 2, the randomization list will come to the user’s browser immediately (see Figure 3).
SAS Code Focus

1. Receiving Macro Variables Dynamically From Web Form

IntrNet uses macro variables to pass parameter values. It is easy to handle single macro variable as we usually do, like in Code 1. A clinical trial often has multiply study sites, and each of them may have different sample size. The %do loop should be used to set the right number of macro variables dynamically according to the individual requests (see Code 2).

One thing that should be noted in Code 2 is the macro key word %global. Since %do loop can be only used within a macro, the scope of the macro variables created in that macro cannot go beyond that macro. So in order to use those macro variables in other part of SAS code, you need to declare them global using %global.

**Code 1:**

* number of study sites;  
%let siteNum=&siteNum_;  

**Code 2:**

* assign sample size for each study site;  
%macro get_site_size;  
  %do i=1 %to &siteNum;  
    %global site&i.Size;  
    %let site&i.Size=&&site&i.Size_;  
  %end;  
%mend get_site_size;

2. Macro Variable Value Validation

If a value entered is not validated on the client side, it should be checked on the server side. For instance, the default value for subject ID start number is 1000. But users may enter whatever they want. In order to make sense to final output, certain limit should be set to it, like in Code 3.

**Code 3:**

* assign the start subject ID;  
%let startNum=&startNum_;  
%macro check_start_num;  
  %if &startNum_<0 | &startNum_>1000000 %then %let startNum=0;  
%mend check_start_num;
3. Block Randomization
The core part of the background SAS code is the procedure, Proc Plan, which plays the roll of a randomization sequence generator, see Code 4.

Code 4:
* randomization within block;
%macro ran_block(blockNum);
...
%if &seed= %then %do;
   proc plan;
       factors N=&blockSize / noprint;
       output data=One out=Two;
   run;
%end;
%else %do;
   proc plan seed=%eval(&seed+&blockNum);
       factors N=&blockSize / noprint;
       output data=One out=Two;
   run;
%end;
...
%mend ran_block;

According to whether the macro variable seed has value or not, Proc Plan will use system time or predefined value as the randomization seed. The difference is that using system time will give the user different output every time the user runs the program, while a specific seed will give the same output no matter how many times the user runs the program, which is recommended. In dataset One, a block has been set up according to the block size. After randomized by Proc Plan, the subjects in the block are in a randomized sequence and result is output as dataset Two.

4. ODS to Browser
The ODS statements that send the output from server to browser are simple, see Code 5.

Code 5:
ODS HTML BODY=_WEBOUT(DYNAMIC TITLE='Randomization List Generator') RS=NONE;
...
SAS Code ...

ODS HTML CLOSE;

_WEBOUT destination and DYNAMIC option are used to generate dynamic web content. TITLE option gives the title of the browser. RS is the alias for the ODS option RECORD_SEPARATOR, which specifies an alternative character or string that separates lines in the output files. RS=NONE option is required when writing content to the _WEBOUT fileref.

ASP and JavaScript Code Point
Because the two web pages that collect randomization parameter information do not involve any data related contents, there is no advantage to use SAS/IntrNet to generate them. If you insist using SAS/IntrNet, the additional thing you need to do is to add some overheads, like DATA _NULL_ and many PUT statements.

Since those two web pages have the dynamic contents and are required to validate some values, ASP and JavaScript can offer good help here. ASP, active server pages, is a server side program that combines HTML and script logic. It is free on IIS server, and is supported by Sun Java System Active Server Pages 4.0 on other servers. ‘asp’ is the extension of the program files. JavaScript is a common client side program. The following three pieces of code will give you a sense how they can help for some applications.

1. ASP
In order to get the right number of table rows for site sample size entry, Code 6 is embedded in the code generating the second web page. Say you have 10 study sites, siteNum=10. When you submit, this code will loops 10 times on the
server and generates the HTML code as Code 7 which is sent to client. This part will be displayed on the browser as in Figure 4.

**Code 6:**
```html
<% for i=1 to Request.QueryString("siteNum") %>
<tr>
    <td width="50%" align="center">
        <%=i%>
    </td>
    <td align="center">
        <input type="text" name="site<%=i%>Size">
    </td>
</tr>
%>next %>
```

**Code 7:**
```html
<tr>
    <td width="50%" align="center">
        1
    </td>
    <td align="center">
        <input type="text" name="site1Size">
    </td>
</tr>
<tr>
    10
</tr>
<tr>
    <td width="50%" align="center">
        <input type="text" name="site10Size">
    </td>
</tr>
```

---

**Figure 4.**

<table>
<thead>
<tr>
<th>Site/Unit No.</th>
<th>Sample Size</th>
<th>Equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

2. **JavaScript**

Code 8 will give a warning message when the user clicks on buttons for the first scenario on Figure 1 if the total sample size is less than 1 or the number of treatments is less than 1 or the total sample size is less than the number of
treatments. Using this kind of functions, we can dramatically reduce the errors caused by wrong entries. This is much more efficient by doing checking on the user’s machine other than sending the parameters to the web server that may forward them to another SAS server to do the checking.

**Code 8:**
```javascript
<SCRIPT LANGUAGE="JavaScript"><!--
function onSubmitForm() {
  if (document. form1.treatNum.value < 2 | document. form1.treatNum.value>document. form1.TSS.value)
    alert("The total sample size must not less than the number of study treatments which is at least 2.");
  else
    return true;
  return false;
}
//--></SCRIPT>
```

3. **ASP and JavaScript**
ASP and JavaScript can work hand in hand to make programmer’s life easier. Code 9 shows how the convenience button, Equal, is set up in Figure 4. The function for the Equal button is that it will enter all the site sample sizes automatically if all the site sample sizes are equal. If there are many study sites, this button will make users feel good by clicking on it.

**Code 9:**
```javascript
<SCRIPT LANGUAGE="JavaScript"><!--
function equalSize() {
  <% for i=1 to Request.QueryString("siteNum") %>
    document.form2.site<%=i%>Size.value=Math.round(<%=Request.QueryString("TSS")%>/<%=Request.QueryString("siteNum")%>);
  <% next %>
}
//--></SCRIPT>
```

For instance, the total sample size is 200 and the number of study sites is 10, pressing the Equal button on Figure 4 will generate the following HTML code, Code 10, and the result of this code is that every blank will be filled with 20 in Figure 4.

**Code 10:**
```javascript
<SCRIPT LANGUAGE="JavaScript"><!--
function equalSize() {
  document.form2.site1Size.value=Math.round(200/10);
  document.form2.site2Size.value=Math.round(200/10);
  ...
  ...
  document.form2.site10Size.value=Math.round(200/10);
}
//--></SCRIPT>
```

**Summary**
SAS/IntrNet is a wonderful bridge connecting SAS platform and Internet. By using it, SAS programmer can offer SAS online services. To take the full advantages of this bridge, we may also need good ramps, traffic lights and signs. Programs, such as ASP, JSP, JavaScript, VBScript, and so on, are candidates for good helpers. For most of the time, IntrNet can do the same thing as other server side programs, but others could be easier for the parts without data processing components. In terms of client side programming, script languages are the tools that should be used. As demonstrated in this example, for some interactive SAS online projects, the integration of all these programming skills not only satisfies the customers with quality services, but also makes the programming job more interesting.
Reference

Contact Information
Sijian Zhang, MS
Database Analyst
Data Management and Statistical Analysis Unit
Center for the Advancement of Youth Health
University of Alabama at Birmingham
912 18th Street South
Birmingham, AL 35294
Phone: (205) 975-0365
Fax: (205) 975-9494
Email: sijian@uab.edu

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

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