ABSTRACT
Users often express the need to access values that are printed by procedures but that are not available through an output statement. There is also a need to be able to customize procedure output in terms of data arrangement and labeling. A current development project explores new alternatives for creating and processing procedure listings that will meet these needs and more. The Output Delivery System (ODS) gives the user greater access to procedure output than has been possible in the past. Procedures using ODS create persistent output objects from which you can make SAS data sets, produce printed listings, reorganize and save output in SAS catalogs, and create permanent custom report formats. Two experimental procedures in base SAS software enable you to manipulate and modify ODS output objects: the OUTPUT procedure and the TEMPLATE procedure.

BACKGROUND
Although many procedures employ an OUTPUT statement to directly convert computed results into SAS data sets, there are still values of interest that are only available in printed form. To access these values, you must now use the PRINTTO procedure to capture listings as character files and then use DATA step code to extract their character representation. This process can be tedious and can result in a loss of numerical precision.

Current Options for SAS Procedure Output
A typical example of this process is given in the following code example where a p-value from the GLM procedure is retrieved from a listing character file to determine the significance of the regression.

```
/*- Route GLM Results to a Text File */
filename glmout 'glmout.lst';
options ps 100 is 80;
proc printto print=glmout neln run;
proc glm data one;
class block whole subplot;
model y = block whole block*whole subplot whole*subplot / solution;
run;
proc printto run;
/*- Get p-values from Character File */
data glm(keep parm p_t);
retain flag 0;
infile glmout missover;
/*- Read ALL lines with same format */
/*- and suppress errors. */
input il lin char20 ilin55 p_t 11;
/*- Look for first obs of interest */
if trim(parm) = 'SURFACE' then do;
  output;
  flag = 1;
end;
/*- Output the rest of the parms */
else if flag then do;
  parm = ' ,';
  if p_t < 0.001 then
    comment = 'Highly Significant';
  else if .001 < p_t <= .01 then
    comment = 'Very Significant';
  else if .01 < p_t <= .05 then
    comment = 'Significant';
  else if p_t = . then
    comment = 'Not Significant';
else
  comment = 'Not Significant';
run;
/*- EXAMPLE DATA SET */
data one;
  input eu block whole subplot ztotdl y @@;
cards;
  111116 1112264
  111416 6 6 4121283
790
```
The following output shows the results of post-processing p-values extracted from PROC GLM listings.

```
Significance of P-Values from Solution Vector

<table>
<thead>
<tr>
<th>OBS</th>
<th>PARM</th>
<th>P_T</th>
<th>CURRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTERCEPT</td>
<td>0.0001</td>
<td>Highly Significant</td>
</tr>
<tr>
<td>2</td>
<td>BLOCK</td>
<td>0.0392</td>
<td>Significant</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.4517</td>
<td>Not Significant</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.0067</td>
<td>Very Significant</td>
</tr>
<tr>
<td>5</td>
<td>WHOLE</td>
<td>0.2149</td>
<td>Not Significant</td>
</tr>
<tr>
<td>6</td>
<td>BLOCK*WHOLE</td>
<td>0.5250</td>
<td>Not Significant</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0.0075</td>
<td>Very Significant</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0.0036</td>
<td>Very Significant</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>0.0089</td>
<td>Highly Significant</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0.1147</td>
<td>Not Significant</td>
</tr>
<tr>
<td>11</td>
<td>SUBPLOT</td>
<td>0.0150</td>
<td>Significant</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>0.0416</td>
<td>Significant</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>0.0009</td>
<td>Highly Significant</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>0.0036</td>
<td>Very Significant</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>0.0089</td>
<td>Highly Significant</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>0.1147</td>
<td>Not Significant</td>
</tr>
<tr>
<td>17</td>
<td>WHOLE*SUBPLOT</td>
<td>0.0150</td>
<td>Significant</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>0.0416</td>
<td>Significant</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>0.0009</td>
<td>Highly Significant</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>0.0036</td>
<td>Very Significant</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>0.0089</td>
<td>Highly Significant</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>0.1147</td>
<td>Not Significant</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>0.0150</td>
<td>Significant</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>0.0416</td>
<td>Significant</td>
</tr>
</tbody>
</table>
```

This simplistic approach, while effective, is not robust because it
- depends on the PARM variable field being a certain width
- depends on a fixed starting column for the P_T variable which can be affected by the line size setting
- necessitates the use of input error suppression to read through the listing file.

Taking this approach, a custom DATA step would need to be formed each time the parameters of PROC GLM changed. Of course the robustness of the p-value extraction code could be improved but it would be nice if the p-values could be accessed directly regardless of the form of the MODEL statement. Later in this paper, we will show some new alternative solutions to this problem using ODS with the MIXED procedure.

In addition to the issue of accessing data values contained only in procedure listings, procedure listings themselves are not always arranged in the form that the SAS user needs. To modify the listing appearance now requires a post procedure editing process which can be tedious. Later in this paper we show how PROC TEMPLATE can be used to permanently alter the appearance of a procedure listing to meet specific needs.

THE OUTPUT DELIVERY SYSTEM - ODS

It has been a goal to develop a way for procedures to record their results in a form that gives the end user the most flexibility in using those results. The user should be able to
- put any numeric listing value into a SAS data set without losing machine precision
- print any listing or groups of listings any number of times after
- the procedure has ended
- group listings in a way that makes sense for the task at hand
- save listings or groups of listings to SAS catalogs to be read in at a later time without losing any numeric precision
- alter the printed format of a listing after the procedure has ended
- alter the default format of a particular procedure listing
- link procedure output more directly into other SAS products such as SAS/EIS software.

To achieve these goals, the Output Delivery System (ODS) has been developed. ODS is an object-oriented facility which SAS developers can incorporate into procedures. Procedures using ODS produce their results in the form of output objects, data structures which persist in memory across procedure boundaries.

CURRENT ODS COMPONENTS

Currently, ODS includes PROC OUTPUT, PROC TEMPLATE, ODS-compatible procedures, and two macro variables for system control.

PROC OUTPUT

The experimental OUTPUT procedure enables you to perform operations on output objects such as converting them directly to SAS data sets or printing them as standard listings. It also provides general ODS hierarchy management enabling you to copy, delete, list, and move memory resident output objects and to save output objects in SAS catalogs for loading during subsequent SAS sessions. It has both an interactive line mode and a window mode.

PROC TEMPLATE

The experimental TEMPLATE procedure enables you to redefine the contents and appearance of a procedure's output by creating and storing alternate templates for a procedure to use during its output phase. These alternate templates control the arrangement and formats of data values, as well as listing annotations such as spanning titles and column headings.

ODS-Compatible Procedures

Currently, only the MIXED procedure is ODS-compatible (with Release 607). PROC MIXED fits mixed linear models (fixed and random-effects models). A mixed model is a generalization of the standard linear model used in the GLM procedure, the generalization being that you can analyze data with several sources of variation instead of just one. The inferential statistics are similar, but their scope is broader.

Macro Variables for Control

Two global macro variables control the actions of ODS: _DISK_ and _PRINT_. You must define these macro variables before the execution of the ODS-compatible procedure to control the way that procedure output is handled. Since these macro variables are global, they will be effective until the end of the SAS session or until they are reset.

Setting _DISK_ to ON will cause procedures to produce persistent output objects. If _DISK_ is not defined, no persistent output objects are produced. Setting _PRINT_ to OFF suppresses the
printing of listings during procedure execution. If _PRINT_ is not defined, printing occurs as normal.

THE ODS HIERARCHY

When procedures create ODS output objects, the objects are placed in the ODS hierarchy. All subsequent references to these objects are made in terms of their location in this hierarchy. The ODS hierarchy is comprised of two basic classes of output objects: listing objects and group objects shown in Figure 2.

![Future Direction for SAS Procedure Output: Output Delivery System - ODS](image)

Listing objects contain the data produced by procedures and consist of two parts, templates and data values. Templates define the organization of the data as well as presentation attributes such as column headers and formats. Data values are the unformatted numbers generated by the procedure. Because the data values have not been formatted, numerical information in a listing object retains its full precision.

Group objects, on the other hand, exist primarily to organize listing objects and other group objects into manageable subsets. These groupings and subgroupings of output objects form the ODS hierarchy which always builds on the ROOT group.

Normally, each invocation of an ODS-compatible procedure groups its listings under a group object with the name of the procedure. These groups are then placed under the ROOT group in the order that they are generated. Interactive procedures may insert run groups under the procedure grouping. If BY-processing is requested, BY-group objects may also be inserted in the hierarchy between the procedure run-groups and the individual listing objects for that BY-group. The SAS user can also create group objects using PROC OUTPUT to help organize their work (see "PROC OUTPUT Line Mode Example" later for more information).

WORKING WITH ODS LISTING OBJECTS

As stated before, ODS-compatible procedures can produce both group and listing objects. Currently, listing objects can be converted into data sets, printed, or saved in a SAS catalog using PROC OUTPUT. In the future it is planned that you will be able to directly incorporate listing objects into products like SAS/EIS software.

/* Getting Values From Listing Objects */
/* See ODS Control Variables */
let _disk_ = on;
let _print_ = off;
/* Generate Linear Model */
proc mixed data = one;
  class block whole subplot;
  model y = block whole block*whole
       subplot whole*subplot / solution;
run;
/* Convert Listings to Data Set */
proc output;
  make 'MIXED.SolutionF'
    out = mixed(keep = parm p_t);
run;
/* Generate New Data Set */
data mixed;
  set mixed;
  length comment $20;
  if 0 < p_t <= .001 then
    comment = 'Highly Significant';
  else if .001 < p_t <= .01 then
    comment = 'Very Significant';
  else if .01 < p_t <= .05 then
    comment = 'Significant';
  else if p_t > .05 then
    comment = 'Not Significant';
else
  comment = 'Not Significant';
title 'Significance of P-Values from Solution Vector';
proc print; run;

Notice that the PROC OUTPUT step is not dependent on the particular MODEL statement being used by PROC MIXED. The variable tags source and p_t are established by the procedure author and are documented with the procedure.
If a printed copy of the Tests listing had been needed, the following PROC OUTPUT code could have been substituted in the previous example.

```sas
proc output;
    print 'MIXED.SolutionF';
    make 'MIXED.SolutionF' out = mixed(keep = parm p_t);
run;
```

And if we had wanted to save the SolutionF listing in a SAS catalog as well, the following PROC OUTPUT code could have been used:

```sas
proc output;
    print 'MIXED.SolutionF';
    make 'MIXED.SolutionF' out = mixed(keep = parm p_t);
    save 'MIXED.SolutionF' cat = sasuser.mixed.solf;
run;
```

PROC OUTPUT is discussed in further detail later.

**The MAKE Statement**

If we want to create data sets directly from a procedure, ODS provides the global MAKE statement which can be used within a procedure itself. Using the MAKE statement, our previous example reduces to the following:

```sas
/*- No Printing, No Output Objects -*/
%let _disk_ = on;
%let _print_ = off;
proc mixed data = t1;
    class block whole subplot;
    model y = block whole block whole subplot whole*subplot / solution;
run;
```

```sas
/*- MAKE the 'SolutionF' listing into a data set */
make 'SolutionF' out = mixed(keep = parm p_t);
run;
```

```sas
data mixed;
    set mixed;
    length comment $20;
    if 0 < p_t < .001 then comment = 'Highly significant';
    else if .001 < p_t < .01 then comment = 'Very significant';
    else if .01 < p_t < .05 then comment = 'Significant';
    else p_t = then comment = 'Not significant';
    else comment = 'Not Significant';
run;
```

**MANAGING THE ODS HIERARCHY WITH PROC OUTPUT**

The OUTPUT procedure is used to manage the objects in the ODS hierarchy. It enables you to:

- convert listing objects into data sets
- print listing objects
- manage group and listing objects (add, copy, delete, list, load, move, and save).

The OUTPUT procedure provides two types of user interface:

- Interactive line mode
- Window mode

During interactive line mode operation, statements are entered in run groups. These statements use path names to identify the objects to operate on. Window mode presents an output browser that shows up to three levels of the ODS hierarchy at one time in scrollable lists. Commands are issued via selection-sensitive pop-up menus.

**PROC OUTPUT Line Mode Example**

The following example demonstrates a typical session using PROC OUTPUT in line mode. PROC MIXED is executed twice on the same data set with two different variance-components models, placing its results in the ODS hierarchy. The OUTPUT procedure is then invoked to:

- list the output objects
- print two listing objects for comparison
- make a data set from a listing object
- record information about the analysis
- regroup the output objects
- save the ODS hierarchy permanently in a SAS catalog.

```sas
proc mixed data = t1 mmeq mmeqSol;
    class ration sire;
    model y = ration / solution;
    random sire / g qi;
run;
```

```sas
proc mixed data = t1 mmeq mmeqSol;
    class ration sire;
    model y = ration / solution;
    random sire * ration / g qi;
run;
```

The resulting procedure listing objects are placed in the ODS.
hierarchy. At this point you can list the objects in the hierarchy by using the LIST statement of PROC OUTPUT as follows:

```
proc output; list all; run;
```

Listing For Path: ROOT

1. MIXED...
   1.1 ClassLevels
   1.2 REML
   1.3 G
   1.4 Q
   1.5 CovParms
   1.6 Fitting
   1.7 MSQ
   1.8 MMEQ
   1.9 SolutionF
   1.10 Tests

2. MIXED...
   2.1 ClassLevels
   2.2 REML
   2.3 G
   2.4 Q
   2.5 CovParms
   2.6 Fitting
   2.7 MSQ
   2.8 MMEQ
   2.9 SolutionF
   2.10 Tests

```

The output objects for the two runs of PROC MIXED have been placed into two groups and labeled with the name of the procedure that generated them (in this case, PROC MIXED). Within each group are the listing objects, which correspond to various MIXED tables. These tables are referred to by abbreviated table names, which are documented for each ODS-compatible procedure.

You will notice that each object has been assigned an index number which is related to the current hierarchy path setting. You can use this number to refer to an object when issuing commands. In the following example, the second PROC MIXED group is listed by itself.

```
list '2'; run;
```

Listing For Path: 2 MIXED

1. CLASSLEVELS
2. REML
3. G
4. Q
5. COVPARMS
6. FITTING
7. MSQ
8. MMEQ
9. SOLUTIONF
10. TESTS

The SAS System

The MIXED Procedure

Model Fitting Information for Y

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observations</td>
<td>18.0000</td>
</tr>
<tr>
<td>Variance Estimate</td>
<td>3.9806</td>
</tr>
<tr>
<td>Standard Deviation Estimate</td>
<td>1.9972</td>
</tr>
<tr>
<td>REML Log Likelihood</td>
<td>-36.6372</td>
</tr>
<tr>
<td>Akaike's Information Criterion</td>
<td>-38.6372</td>
</tr>
<tr>
<td>Schwartz's Bayesian Criterion</td>
<td>-39.4098</td>
</tr>
<tr>
<td>-2 REML Log Likelihood</td>
<td>71.2744</td>
</tr>
<tr>
<td>Null Model LRT Chi-Square</td>
<td>0.5974</td>
</tr>
<tr>
<td>Null Model LRT DF</td>
<td>1.0960</td>
</tr>
<tr>
<td>Null Model LRT F-Value</td>
<td>6.4469</td>
</tr>
</tbody>
</table>

NOTE: 2 output item(s) printed.

You can also use wildcard characters when specifying pathnames. Among other things, this helps to compare similar listings from different executions of the procedure. In the following example, the Fitting tables from the two runs of PROC MIXED are printed for comparison.

```
print '*.fitting'; run;
```

The SAS System

The MIXED Procedure

Model Fitting Information for Y

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observations</td>
<td>18.6060</td>
</tr>
<tr>
<td>Variance Estimate</td>
<td>3.1791</td>
</tr>
<tr>
<td>Standard Deviation Estimate</td>
<td>1.4762</td>
</tr>
<tr>
<td>REML Log Likelihood</td>
<td>-34.5315</td>
</tr>
<tr>
<td>Akaike's Information Criterion</td>
<td>-37.5315</td>
</tr>
<tr>
<td>Schwartz's Bayesian Criterion</td>
<td>-38.6060</td>
</tr>
<tr>
<td>-2 REML Log Likelihood</td>
<td>69.0360</td>
</tr>
<tr>
<td>Null Model LRT Chi-Square</td>
<td>4.7849</td>
</tr>
<tr>
<td>Null Model LRT DF</td>
<td>2.0000</td>
</tr>
<tr>
<td>Null Model LRT F-Value</td>
<td>0.0914</td>
</tr>
</tbody>
</table>

NOTE: 2 output item(s) printed.

The MAKE statement can be used to convert any of the listing objects into a SAS data set as follows:

```
make '1.mmeq' out=mmeq; run;
```

NOTE: 1 output item(s) converted.

<table>
<thead>
<tr>
<th>NAME: WORK.MMEQ</th>
<th>DESC: random sire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Set</td>
<td></td>
</tr>
<tr>
<td>Observations</td>
<td>18.0000</td>
</tr>
<tr>
<td>Variance Estimate</td>
<td>3.9806</td>
</tr>
<tr>
<td>Standard Deviation Estimate</td>
<td>1.9972</td>
</tr>
<tr>
<td>REML Log Likelihood</td>
<td>-36.6372</td>
</tr>
<tr>
<td>Akaike's Information Criterion</td>
<td>-38.6372</td>
</tr>
<tr>
<td>Schwartz's Bayesian Criterion</td>
<td>-39.4098</td>
</tr>
<tr>
<td>-2 REML Log Likelihood</td>
<td>71.2744</td>
</tr>
<tr>
<td>Null Model LRT Chi-Square</td>
<td>0.5974</td>
</tr>
<tr>
<td>Null Model LRT DF</td>
<td>1.0960</td>
</tr>
<tr>
<td>Null Model LRT F-Value</td>
<td>6.4469</td>
</tr>
</tbody>
</table>

NOTE: The data set WORK.MMEQ has 18 observations and 15 variables.

Suppose you want to revise the names and descriptions of the two PROC MIXED groups to record the different modeling approaches. This can be done using the INFO statement as follows:

```
info '1' name='MIXED_R1' desc='random sire'; run;
```

NOTE: OUTPUT object MIXED_R1 has been updated.

```
info '1'; run;
```

<table>
<thead>
<tr>
<th>NAME: MIXED_R1</th>
<th>DESC: random sire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Set</td>
<td></td>
</tr>
</tbody>
</table>

794
Now suppose that you wish to group these two PROC MIXED runs under another group which represents work done on the data set t1. Create a new group object using the ADD statement and then move hierarchy segments under it using the MOVE statement as follows:

```
add name='TI_DS' descr='Ration analysis'; run;
NOTE: TI_DS group has been added under ROOT group.
list; run;
Listing For Path: ROOT
  1 MIXED_R1...
  2 MIXED_R2...
  3 TI_DS...
move 'mixed_r1' under 'tl_ds';
move 'mixed_r2' under 'tl_ds';
run;
NOTE: 2 output item(s) moved under TI_DS.
NOTE: 2 output item(s) moved under TI_DS.
list all; run;
Listing For Path: ROOT
  1 TI_DS...
    1.1 MIXED_R1...
      1.1.1 ClassLevels
      1.1.2 REML
      1.1.3 G
      1.1.4 GI
      1.1.5 CovParms
      1.1.6 Fitting
      1.1.7 MMEQ
      1.1.8 MMEQSOL
      1.1.9 SolutionF
      1.1.10 Tests
    1.2 MIXED_R2...
      1.2.1 ClassLevels
      1.2.2 REML
      1.2.3 G
      1.2.4 GI
      1.2.5 CovParms
      1.2.6 Fitting
      1.2.7 MMEQ
      1.2.8 MMEQSOL
      1.2.9 SolutionF
      1.2.10 Tests
```

To save this entire ODS hierarchy to a catalog for later recall (even in another SAS session), use the SAVE statement as follows:

```
save '1' cat=sasuser.series1.tl; run;
NOTE: An output group or item has been saved to catalog.
```

**PROC OUTPUT Window Mode**

By invoking PROC OUTPUT with the WINDOW option, you are presented with an output browser which shows up to three levels of output hierarchy at once. Action menus for the displayed output objects are accessed through a popup command key.

At initialization, the contents of the ROOT group are displayed in the leftmost list. If you wish to operate on one of the items listed in the ROOT group or any subgroup list, select that item and popup the action menu. In general, the menu that appears in any given list applies to the item currently selected in that list.

![PROC OUTPUT Window Mode](image)

**Figure 4 PROC OUTPUT Window Mode**

If you select a list item that is a group (denoted by trailing ellipses), its contents are displayed in the next list pane to the right. If the selected group is in the rightmost list pane, all lists are shifted left to reveal the contents of the newly selected group. If you wish to look at an output level that is not currently in view, you can use the arrow buttons on the right side of the browser window to shift lists into view.

The END button ends PROC OUTPUT but does not delete any output objects. You can reenter PROC OUTPUT at any time during a SAS session and find the output objects as you left them, along with any that have been added by subsequently run ODS-compatible procedures. When the SAS session ends, however, any ODS objects that have not been saved in a SAS catalog will be lost.

**CUSTOMIZING PROCEDURE OUTPUT WITH PROC TEMPLATE**

The TEMPLATE procedure enables you to modify the attributes of listing object templates. This, in turn, will affect the appearance of the printed listing object. Currently, the only editable listing object template is the TABLE template.
Figure 5 Customizing Procedure Output with PROC TEMPLATE

By editing a table template you can

- modify column headings
- specify the width, format, and justification for a column
- specify the space between columns
- specify the character that splits column titles
- delete, rearrange, or suppress the printing of columns
- flow text over multiple lines in a column
- force a page break before a column
- specify a column variable name for use when converting the table to a SAS data set
- specify titles that span columns.

PROC TEMPLATE is designed to operate in the batch environment only. Future plans include an interactive window mode template editor as an extension to PROC OUTPUT (see Future Directions for Templates). PROC TEMPLATE enables you to save modified template definitions to a local SAS catalog. ODS-compatible procedures can then use these templates during execution as overrides to their default templates.

Example Using PROC TEMPLATE

As an example, consider the following table, which appears in the default format provided by PROC MIXED:

Parameter	Estimate	Std Error	T Pr > |T|  
INTERCEPT	25.59	4.04209620	6.21	0.0001  
A 1	3.25	2.04613.17	0.3118  
A 2	4.75	2.04612.36	0.1533  
B 1	6.50	1.87360.37	0.7936  
B 2	0.00

Perhaps instead you want a listing that looks like the following one. Using PROC TEMPLATE, you can create a template that underlines the column titles, expands and wraps the title of the Std Error column, specifies different formats, adds extra titles, and deletes the DDF, Alpha, Lower, and Upper columns.

The name of this listing object is SolutionF. Generating the SolutionF template source with PROC TEMPLATE produces the following SAS statements:

```sas
table / space = 4 width = 12 split = '/';
column param est se ddf t p_t alpha lower upper;
define parm / 'Parameter' varname = parm;
define est1 / 'Estimate' format = 12.8 varname = est;
define se / 'Std Error' format = 12.8 varname = se;
define ddf / 'DDF' format = 4.0 width = 4 varname = ddf;
define t / 'T' format = 6.2 width = 6 varname = t;
define p_t / 'Pr > |T|' format = 6.4 width = 6 varname = p_t;
define alpha / 'Alpha' format = best. varname = alpha;
define lower / 'Lower' format = 8.4 width = 8 varname = lower;
define upper / 'Upper' format = 8.4 width = 8 varname = upper;
span parm upper / 'The Mixed Procedure', 'Solution for Fixed Effects';
```
To create the new template for the SolutionF listing, you must modify the SAS source code and compile it with PROC TEMPLATE, storing the modified template in SASUSER_PROFILE. To create the desired SolutionF listing format, you would submit the following code:

```
proc template
type = table
out = sasuser.profile.mixoutl.outtemp;
  
  table / space = 2
  width = 12
  split = '/'
  
  headline;
  
  column parm est se p_t;
  define parm / 'Parameter' varname = parm;
  define est / 'Estimate' width = 8 format = 8.2 varname = est;
  define se / 'Standard/Error' format = 12.4 varname = se;
  define t / 'T' format = ~6.2 width = 6 varname = t;
  define p_t / 'Pr> |t|' format = 8.4 width = 8 varname = p_t;
  
  span parm parm I
  'Research Project 19-B' 'Test Group C';
  span parm p_t
  , =, ', 'Solution For Fixed Effects';
run:
```

The OUT= catalog entry name must match the template name expected by PROC MIXED for this listing and the entry must be of entry type OUTTEMP. When PROC MIXED generates output, it looks first in SASUSER_PROFILE for the correct OUTTEMP entry to use as a template. If there is none, PROC MIXED uses the default template.

**Future Directions for Templates**

Currently PROC TEMPLATE can only modify predefined procedure templates and this modification must be done before the procedure which uses them is run. In the future it is planned that SAS users will be able to

* edit the template component of listing objects after a procedure has been run thus eliminating the need to re-run the procedure to see template changes
* save modified listing templates in a local SAS catalog
* specify the location of alternate default templates using a global TEMPLATE statement.

A window-based template editor is being planned which will enable you to edit the template component of listing objects interactively. The results of this edit would be in effect immediately for print and make operations.

**CONCLUSIONS**

The Output Delivery System as it stands today presents an exciting new direction for handling SAS procedure output. You are given much more direct access to SAS procedure output through the use of persistent output objects. ODS provides a broad palette of output management options, which include printing, directly converting output objects to SAS data sets, and storing output objects in SAS catalogs, all with no loss of numeric precision. And output object template editing gives you the freedom to arrange information to meet your needs.

**YOUR TURN**

If you have any comments or suggestions concerning procedure output management, we can be reached through Internet Email as follows:

Chris Olinger - <sascro@unix.sas.com>,
Robert Ray - <sasrsr@unix.sas.com>.

Thank you for your interest.

SAS, SAS/AF, and SAS/EIS are registered trademarks of SAS Institute Inc. in the USA and other countries. indicates US registration.