You CAN Get There from Here (and Back Again):
Adding Hot-link Drill-down Capabilities to ODS HTML Output

Ray Pass
Ray Pass Consulting

ABSTRACT
The SAS Output Delivery System (ODS) can be used to create HTML pages for publishing on the Internet or an Intranet. Right out of the box (and the box is FREE too! – with Base SAS), ODS can be used to create either one page per procedure output, or a collection of HTML pages per procedure. ODS does provide within-procedure HTML navigability, but there is no wholesale provision provided for inter-procedure output linking. By using simple SAS MACRO processing combined with the manipulation of TITLE statements, basic data set variable values and values of variables used as CLASS variables in TABULATE procedures, you can create whole systems of hot-linked pages with almost full drill-down capabilities. The technique is to simply substitute HTML-navigational-tag enriched data values for those displayed values that you want to act as hot links, and let ODS do the rest! Techniques and examples are shown in this paper, along with data-driven techniques for renaming ODS-HTML generated sequential body file names into more meaningful content-oriented names.

INTRODUCTION
The SAS Output Delivery System (ODS to its friends) has given us a method of preparing HTML formatted versions of output from all SAS procedures that produce output. When used with the new TEMPLATE procedure and the new STYLES feature, all aspects of procedure output, including row, column and cell elements as well as titles for the accompanying output, can be fully formatted.

ODS takes the content results from SAS procedures, or the contents of SAS data sets, and combines these data with a display definition to create an output object. This output object is then routed to an ODS destination. The destinations in production as of V8.1 include HTML, LISTING (standard SAS output), RTF, PRINTER and OUTPUT (standard SAS DATA set). Output objects that are sent to the HTML destination are rendered as HTML formatted documents which can then be read via an HTML browser. The documents consist of titles and HTML tables, and can be afforded the full range of HTML formatting enhancements, including Cascading Style Sheets.

Although the transformation from standard SAS output to HTML formatted output is a major step forward in terms of creating high information content reports which can be made available via the internet or any size intranet to large groups of information consumers, it is still rudimentary in nature because each output is static. One of the seminal definitional features of web information presentation is the ability to connect related informational displays via hot links (hypertext), or drill-down techniques. The hot link capability is accomplished by enriching an element so it can serve as a clickable launching point to a related collection of information. The main enrichment is in the form of the location, or address, of the target display. This paper presents the basis of a methodology for creating these enriched data elements, as well as for the automatic data-driven generation of all the static outputs necessary for a complete system of related informational displays (static report outputs). It also discusses techniques for amplifying (renaming) the sequentially labeled body files created by default by ODS HTML. This technique is totally data-driven and yields meaningful content-specific names for the files.

This paper assumes that the reader is already familiar with the basic concepts of ODS. It is not an introduction to these techniques, but is rather an explication of an addition to them. ODS documentation can be found in the SAS publication entitled The Complete Guide to the SAS Output Delivery System, Version 8, although this document is only current up to V8.0. There are also numerous SUGI and Regional SUG papers available on ODS.

SAMPLE DATA
The data to be used throughout his paper consists of periodic sales reports of the fictitious RPC Entertainment Enterprises Corporation. There are six geographical regions in the company (NorthEast, NorthCentral, NorthWest, SouthEast, SouthCentral, SouthWest), and each region is further broken down into the states making up the region. There are two divisions in the company (Games, Toys) and each division is broken down into various items produced by the division.

NON-HTML REPORTS
A series of PROC TABULATEs (and accompanying TITLE statements) can be used without HTML enhancement to produce the needed daily sales reports. Although all of the Region by State by Division by Item data could be presented in their most granular form in one TABULATE output, the end goal here is a hypertext system of related reports. A representative sample of
non-HTML modular reports is presented here. Only one of the six (one for each Region) possible \textit{State by Division} reports, one of the two (one for each Division) possible \textit{Region by Item} reports, and one of the 12 (one for each Region-Division combination) possible \textit{State by Item} reports are shown. These are found in Figs 1-4.

Fig 1. Non-HTML Report  
\textit{Region by Division}

Fig 2. Non-HTML Report  
\textit{State by Division (Region: NorthEast)}

Fig 3. Non-HTML Report  
\textit{Region by Item (Division: Toys)}

Fig 4. Non-HTML Report  
\textit{State by Item (Region: NorthEast, Division: Toys)}
SIMPLE ODS HTML OUTPUT

To use ODS to produce HTML output instead of the standard output listings seen above, we would issue an ODS HTML statement before running the various TABULATE procedures. As this paper assumes the basic mechanics and syntax of ODS are known to the reader, it will not go into great detail about the ODS code used herein.

Outputs of the rudimentary use of ODS code as viewed via an HTML browser are shown in Figs 5-8. Full HTML table formatting, including font and color choices, etc. is available via the TEMPLATE procedure. The output presented here uses default formatting values.

The results of using the ODS HTML statement are a series of independent HTML documents containing the output from the TABULATE procedure translated into HTML tagged code. The names of these output file are contained as parameters in the ODS code. As an example, the actual names of the HTML files as displayed via browser in Figs 5-8 could be respectively:

- simple 2000-06-01-regxdiv.htm
- simple 2000-06-01-stxdiv-NE.htm
- simple 2000-06-01-regxitm-Toys.htm

ENHANCED ODS HTML OUTPUT

Now that we have the individual reports rendered as HTML documents, the next step is to create the navigational tools to be able to go from one report to another by clicking on a hot link. This part is the crux of the whole method presented here, and is actually really very simple. All that needs to be done is to create a set of alternate variables from which the tables are constructed. The change that is necessary is to enhance each data item (that is to be displayed as a hot link) with additional location information contained in HTML tags, specifically HREF tags. As an example, each region is
actually coded as a two-byte character variable called REG with values of: NC, NE, NW, SE, SC, or SW. The non-HTML output uses a user-defined format, $REGFMT, to display expanded names via a FORMAT statement in the TABULATE code. If a separate State by Division HTML table was created for each region, they could have file names of:

- enhanced 2000-06-01-stxdiv-NC.htm
- enhanced 2000-06-01-stxdiv-NE.htm
- enhanced 2000-06-01-stxdiv-NW.htm
- enhanced 2000-06-01-stxdiv-SC.htm
- enhanced 2000-06-01-stxdiv-SE.htm
- enhanced 2000-06-01-stxdiv-SW.htm

If we create an alternate variable REG2 by surrounding the old values of REG with the above path names along with the appropriate HTML code, we could then produce HTML clickable links in our TABULATE output by using REG2 in the TABULATE code instead of REG. We would also create a variable REGEXT as a FORMAT expanded version of REG. The code to create these variables would look something like:

```plaintext
regext = put(reg, $regfmt.);
reg2 = "<A HREF='enhanced 2000-06-01-stxdiv-" || trim(reg) || ".htm'>" || regext || "</A> ;
```

This would produce a converted value for NE of:

```plaintext
<A HREF='enhanced 2000-06-01-stxdiv-NE.htm'>NorthEast</A> .
```
Actually, there would have to be two alternate versions of \textit{REG}, a \textit{REG2} and a \textit{REG3}, because clicking on a region name in the \textit{Region by Division} table would have to link to a \textit{State by Division} table for the clicked region, whereas clicking on the same region name in a \textit{Region by Item} table would link to a \textit{State by Item} table for the clicked region. Since the names of the two target files would be different, the values of \textit{REG2} and \textit{REG3}, both based on \textit{REG}, would be different. Two alternate versions of \textit{DIV} would be needed as well.

If the new versions of \textit{REG} and \textit{DIV} were used in the same \texttt{TABULATE} code that was used to produce the tables shown in Figs 5-8, ODS would cause the region and division names to be displayed as clickable HTML hot links. The resulting HTML output files as seen via an HTML browser are shown in Figs. 9-12.

\begin{tabular}{|c|c|c|}
\hline
\textbf{Region} & \textbf{Games} & \textbf{Toys} \\
\hline
\textit{NorthCentral} & 10,490 & 9,590 & 20,080 \\
\hline
\textit{NorthEast} & 11,940 & 12,828 & 24,768 \\
\hline
\textit{NorthWest} & 5,202 & 5,626 & 10,828 \\
\hline
\textit{SouthCentral} & 6,070 & 6,314 & 12,384 \\
\hline
\textit{SouthEast} & 9,990 & 9,990 & 19,980 \\
\hline
\textit{SouthWest} & 7,389 & 6,277 & 13,666 \\
\hline
\textbf{TOTAL} & 51,081 & 50,625 & 101,706 \\
\hline
\end{tabular}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig9.png}
\caption{Enhanced HTML Report \hfill Region by Division}
\end{figure}

That’s basically all there is to it. Just substitute enhanced data values for those values that you want to act as hot links and let ODS do the rest. There is more to the overall method to deal with, but that is the basic technique.

When this method is actually used on a periodic basis to create the reports, the \texttt{TODAY()} function is used for the dates, although its output format is modified with the following statement:

\begin{verbatim}
%let today1=%sysfunc(today(),ymmd10.);
\end{verbatim}

Another date transformation is used to get the formatted date in the titles (each \texttt{TITLE} statement has \&\texttt{TODAY2} in it):

\begin{verbatim}
%let today2=%sysfunc(today(),worddate12.);
\end{verbatim}

\begin{document}
\begin{table}
\centering
\begin{tabular}{|c|c|c|}
\hline
\textbf{Division} & \	extbf{TOTAL} \\
& \\
\hline
\textbf{State} & \\
& \\
\hline
\textit{CT} & 631 & 1,075 & 1,706 \\
\hline
\textit{DC} & 1,319 & 1,763 & 3,082 \\
\hline
\textit{DE} & 207 & 631 & 838 \\
\hline
\textit{WA} & 1,075 & 1,319 & 2,394 \\
\hline
\textit{MD} & 1,763 & 207 & 1,970 \\
\hline
\textit{ME} & 631 & 1,075 & 1,706 \\
\hline
\textit{NH} & 1,319 & 1,763 & 3,082 \\
\hline
\textit{NJ} & 207 & 631 & 838 \\
\hline
\textit{NY} & 1,075 & 1,319 & 2,394 \\
\hline
\textit{PA} & 1,763 & 207 & 1,970 \\
\hline
\textit{RI} & 631 & 1,075 & 1,706 \\
\hline
\textit{VT} & 1,319 & 1,763 & 3,082 \\
\hline
\textbf{TOTAL} & 11,940 & 12,828 & 24,768 \\
\hline
\end{tabular}
\caption{RPC Entertainment Enterprises \hfill State by Division Sales Report: June 1, 2000}
\end{table}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig10.png}
\caption{Enhanced HTML Report \hfill State by Division (Region: NE)}
\end{figure}

This creates date values of the form 2000-06-01. The actual code for the \texttt{REG2} assignment statement is then:

\begin{verbatim}
reg2 = "<A HREF="enhanced" \\
"&today1.-stxdiv-" \\
trim(reg) \\
".htm">" \\
\texttt{regext} \\
"</A>";
\end{verbatim}

Another date transformation is used to get the formatted date in the titles (each \texttt{TITLE} statement has \&\texttt{TODAY2} in it):

\begin{verbatim}
%let today2=%sysfunc(today(),worddate12.);
\end{verbatim}

ADDITIONAL ODS HTML OUTPUT

OK, now we know how to create hot links in the row and column headers in the ODS HTML output tables. But, there’s more to it than that. We can also create a stand-alone HTML Table of Contents by passing PROC PRINT, PROC REPORT, or even PROC TABULATE output through ODS HTML processing. This notion of creating separate stand-alone ODS HTML Tables of Contents is covered more fully in another paper by the current author entitled I’ll Have the Tabulates a la ODS Please, With a Table of Contents on the Side (found in the Proceedings of the NESUG 2000 and SESUG 2000 conferences). The following macro code creates the output Table of Contents page as shown in Fig. 13 (as viewed through an HTML browser). The string values for REP and the TITLE statement are broken here to fit into the column requirements of this paper; in actual code they are each one continuous string.

```sas
%macro repstoc;
  data reps;
    length rep $ 77;
    label rep="Reports for &today2";
    rep="<A HREF='&today1.-regxdiv.htm'>Region by Division Sales Report</A>";
    output;
    rep="<U>Report 2 (<I>non-operational</I>)</U>";
    output;
    rep="<U>Report 3 (<I>non-operational</I>)</U>";
    output;
  run;
%mend;
```

Fig 11. Enhanced HTML Report
Region by Item (Division: Toys)

<table>
<thead>
<tr>
<th>Item</th>
<th>GI Joe</th>
<th>SI Jim</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NorthCentral</td>
<td>4,740</td>
<td>4,850</td>
<td>9,590</td>
</tr>
<tr>
<td>NorthEast</td>
<td>5,748</td>
<td>7,080</td>
<td>12,828</td>
</tr>
<tr>
<td>NorthWest</td>
<td>2,430</td>
<td>3,146</td>
<td>5,576</td>
</tr>
<tr>
<td>SouthCentral</td>
<td>2,824</td>
<td>3,430</td>
<td>6,254</td>
</tr>
<tr>
<td>SouthEast</td>
<td>4,440</td>
<td>5,550</td>
<td>9,990</td>
</tr>
<tr>
<td>SouthWest</td>
<td>2,750</td>
<td>3,527</td>
<td>6,277</td>
</tr>
<tr>
<td>TOTAL</td>
<td>22,982</td>
<td>27,643</td>
<td>50,625</td>
</tr>
</tbody>
</table>

Fig 12. Enhanced HTML Report
State by Item (Region: NE Division: Toys)

```sas
ods html path = repout body = "&today1._reps_toc.htm";
*-----------------------------------------;
title1 "<H4><A HREF='_cal.htm'>Calendar</A></H4>";
run;
*-----------------------------------------;
proc print data=reps noobs label;
run;
*-----------------------------------------;
ods html close;
%mend;
```

Fig 13. Report Table of Contents
We can also use ODS HTML processing to create a Calendar page (as viewed via an HTML browser in Fig. 14). New reports are created every two weeks in this example with the creation date used as part of the file name for each report (HTML file). The code used to create the calendar is presented below. The method also includes a start date for the first month to be displayed.

```sas
%macro calendar;
data day;
  length date $38;
  date = "<A HREF=&_today1._reps-toc.htm>" || put(_today(),_day2.) || "</A>";
  datex = _today();
*---------------------------------------------------;
proc sort data=alldays; by year month datex; run;
*---------------------------------------------------;
proc sort data=alldays; by datex; run;
*---------------------------------------------------;
proc format;
  value $miss (default=38) '=';
*---------------------------------------------------;
data alldays(drop=start d);
  length date $38;
  start = '01jun2000'd;
  d = 0;
  do until (datex=intnx('month',today(),1)-1);
    datex = start + d;
    date = put(_today(),_day2.)
    year = year(datex);
    month = month(datex);
    output;
    d+1;
  end;
  run;
*---------------------------------------------------;
proc sort data=alldays; by datex; run;
*---------------------------------------------------;
data alldays;
  update alldays hot._days; by datex; run;
*---------------------------------------------------;
data alldays;
  by year month;
  set alldays end=lastrec;
  length mon_yr monyy7. ;
  array daysx{7} sunx monx tuex wedx thux frix satx;
  array days{7} $ sun mon tue wed thu fri sat;
  format sun mon tue wed thu fri sat $miss. ;
  by year month;
  define mon_yr / order noprint;
  columns mon_yr sun mon tue wed thu fri sat;
  define monyy7. / order noprint;
  columns monyy7. ;
  run;
  endcomp;
*---------------------------------------------------;
ods html close
;$%end calendar;
```

## HTML ENHANCED TITLES

The next parts of the process needed to create the fully navigational system are hot-link enhanced titles. These are easy to do because they come along for a free ride when ODS is used to create HTML. The trick is to make sure that the correct titles appear at the top of each page. Parts of the titles can be generally modularized as follows:

```sas
%let mcal = <A HREF=_cal.htm>Calendar</A>;
%let mrep = <A HREF=_today1._reps-toc.htm>Reports</A>;
```

These would be used to produce the hot links to go to the Calendar page and a day-specific Table of Contents page, as in:

```sas
title1 "<H3>RPC Entertainment Enterprises</H3> Region by Division Sales Report: &_today2</H3>"

title2 "<H4>&mcal &mrep</H4>"
```

Other titles would be constructed using the same general format.

## DATA-DRIVEN MACRO AUTOMATION

To make the whole system automatic and data driven, it is all contained in a system of macros, which works as follows. `%CALENDAR` is used to recreate the HTML Calendar page each report day, and `%REPSTOC` is run to create a new date-specific HTML Table of Contents page. Each day that the program is run the source data set is recreated as a SAS data set called `HLDATA`. This data set contains variables `REG`, `DIV` and `SALES`. An alternate data set, `HLDATA2` is created with the converted variables `REGESET`, `REGE2`, `REGE3`, `DIV2` and `DIV3`, along with `SALES`. Next, each type of individual TABULATE report is created as an HTML table and sent to a .htm file. The code for each different type of TABULATE (Region by Division, State by Division, Region by Item, State by Item) is contained in a table macro (`%REGESETDIV`, `%STXDIV`, `%REGEKITM`, `%STXKITM`) along with its ODS code. The code for one of these table macros (`%STXDIV`) is as follows (the rest are similar):
%macro stxdiv;
  *------------------------------------------------------------------------;
  ods html path =repout
    body ="&today1.-stxdiv-1.htm"
    newfile=bygroup;
  *------------------------------------------------------------------------;
  proc sort data=hldata2;
    by regext;
  run;
  *------------------------------------------------------------------------;
  title1 "&H3>RPC Entertainment
    Enterprises
    State by Division Sales
    Report: &today2
    &BR>Region=#byval(regext)</H3>";
  title2 "&mcal &t_sp &mrep &t_sp
    <A HREF=&today1.-regxdiv.htm>
    All Regions</A></H4>";
  run;
  *------------------------------------------------------------------------;
  proc tabulate data=hldata2 missing;
    by regext;
    class state div3;
    var sales;
    table state='State' all='TOTAL',
      (div3='Division' all='TOTAL')
      *sales=' '*sum=' '*f=comma32.;
  run;
  *------------------------------------------------------------------------;
  ods html close;
%mend stxdiv;

A few notes about this macro (and its sister macros for
other table types) are in order. The macro variable &t_sp
is created early in the program as follows:

%let t_sp = %nrstr(   );

and is used to insert visible spaces in the HTML output.
Also, note that the dataset is pre-sorted by the categorical
variable for which drill-down tables are being created
(REGEXT in this case.) This sorting, in conjunction with
the ODS NEWFILE=BYGROUP option (new in V 8.1) is
what creates a separate HTML file for each byvar level.
Note also that the current byvar value is inserted in the
title by means of the #BYVAL notation (the NOBYLINE
option is also in effect.)

Another major feature employed in this system is the
renaming of the ODS produced files from a simple
sequentially suffixed set (name1, name2, ...) to a
meaningfully suffixed set. This is accomplished by
combining data-driven pre-processing of the data with
post-processing file naming via operating system
commands. The code for the first part of the process is
contained in macro %MVARS. This macro is totally data-
driven and contains information about the values of REG
and DIV present in the source data.

%macro mvars;
  %global mrdcount mrcount mdcount regdivs
    mregs mdivs mrdnums mrnums mdnums;
  proc sql;
    create table regdivs as
      select distinct reg,
        div
      from hot._hldata
      order by reg,
        div;
    create table regs as
      select distinct reg
      from regdivs
      order by reg;
    create table divs as
      select distinct div
      from regdivs
      order by div;
    quit;
  *------------------------------------------------------------------------;
  data regdivs(drop=reg div);
    set regdivs;
    regdiv = trim(reg)||'-'||trim(div);
    num = _n_; 
  run;
  *------------------------------------------------------------------------;
  data regs;
    set regs;
    num = _n_; 
  run;
  *------------------------------------------------------------------------;
  data divs;
    set divs;
    num = _n_; 
  run;
  *------------------------------------------------------------------------;
  proc sql noprint;
    select count(*),
      regdiv,
      num
    into :mrdcount,
    :mregdivs separated by '#',
    :mrdnums separated by '#'
    from regdivs;
    select count(*),
      reg,
      num
    into :mrcount,
    :mregs separated by '#',
    :mrnums separated by '#'
    from regs;
    select count(*),
      div,
      num
    into :mdcount,
    :mdivs separated by '#',
    :mdnums separated by '#'
    from divs;
    quit;
  %mend mvars;

The results of the above macro execution are nine macro
variables with the following values:
The actual values of the macro variables are dependent on the actual data in the data set each day – this is the fully populated version.

The second part of the process uses operating system deletion and renaming commands to change the names of the ODS produced files (this code is PC-based and uses the DOS `DEL` and `REN` commands; in UNIX, you would use the `RM` and `MV` commands, etc.) In the above `%STXDIV` macro, the

\[
\text{body} = \ast @\text{today1.-stxdiv-1.htm} \\
\text{newfile} = \text{bygroup}
\]

statements instruct ODS to create a new file for each new byvar level, and to name them (assuming the current date is June 1, 2000 and there are 6 regions)


This is default ODS behavior, and is not very useful, although fully understandable. ODS has no way of knowing what to call the files it produces unless you tell it. Using the data-driven/created macros shown above, the following code renames all the ODS-produced .htm files meaningfully, based on the values of the appropriate byvars:

```plaintext
%macro names;
  %if %sysfunc(fileexist( &hldata.@today1.-stxdiv-1.htm))
    %then %do m=1 %to &mdcount;
      %let num = %scan(&mdnums,&m,#);
      %let div = %scan(&mdivs,&m,#);
      %sysexec del &hldata.@today1.-stxdiv-&div..htm;
      %sysexec ren &hldata.@today1.-stxdiv-1.htm &hldata.@today1.-stxdiv-&div..htm;
    %end;
  %if %sysfunc(fileexist( &hldata.@today1.-regxitm-1.htm))
    %then %do m=1 %to &mdcount;
      %let num = %scan(&mdnums,&m,#);
      %let div = %scan(&mdivs,&m,#);
      %sysexec del &hldata.@today1.-regxitm-1.htm;
      %sysexec ren &hldata.@today1.-regxitm-&div..htm &hldata.@today1.-regxitm-1.htm &hldata.@today1.-regxitm-&div..htm;
    %end;
%end names;
```

For example, 2000-06-01-stxdiv-1.htm becomes 2000-06-01-stxdiv-NC.htm, 2000-06-01-stxdiv-2.htm becomes 2000-06-01-stxdiv-NE.htm, etc. These names now match the names built in to the drill-down HREF names in the TABULATE macros.

The final, fully HTML navigational system, comprised of the calendar, report table of contents and all component table reports, is shown in Figs 14-19, as viewed through an HTML browser.
Fig 15. Final Report Table of Contents

Fig 16. Final HTML Report
Region by Division

Fig 17. Final HTML Report
Region by Item (Division: Toys)
**CONCLUSION**

This paper has presented an introduction to an optional method of turning the static HTML documents created by the SAS Output Delivery System into a fully functional, HTML drill-down navigational, data-driven system of information display that can be implemented on a routine basis in an Internet or intranet environment. The basic paradigm is to enhance the data being used as input to ODS, so the values displayed can be rendered by an HTML browser as fully functional HTML hot links.

The examples presented herein deal with outputs from runs of TABULATE, PRINT and REPORT procedures, as well as enhanced TITLE statements. In addition, a method is displayed in which default ODS created sequentially suffixed HTML file names are renamed to content-meaningful names via totally automated and data-driven processing. This is an integral part of the entire system.

As far as the author is aware, with the exception of some minor use of PROC FORMAT related techniques, the methods presented in this paper are not documented in the SI collection of distributed literature.

The author has used these techniques successfully on a large nationwide intranet system with extremely productive results. In fact, the initial implementation has spawned numerous other similar systems on the same intranet. Now it's your turn to go out and create your own systems. The tools are there.

**REFERENCES**

SAS is a registered trademark of the SAS Institute Inc., Cary, NC, USA.

**CONTACT INFORMATION**

The author can be contacted at:

Ray Pass
Ray Pass Consulting
5 Sinclair Place
Hartsdale, NY 10530

(914) 693-5553
raypass@att.net

---

**Fig 19. Enhanced HTML Report**  
**State by Item (Region: NE**  
**Division: Toys)**

<table>
<thead>
<tr>
<th></th>
<th>Item</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GI Joe</td>
<td>SI Jim</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CT</td>
<td>492</td>
<td>593</td>
</tr>
<tr>
<td>DC</td>
<td>925</td>
<td>937</td>
</tr>
<tr>
<td>DE</td>
<td>260</td>
<td>371</td>
</tr>
<tr>
<td>MA</td>
<td>604</td>
<td>715</td>
</tr>
<tr>
<td>ND</td>
<td>49</td>
<td>159</td>
</tr>
<tr>
<td>NE</td>
<td>482</td>
<td>593</td>
</tr>
<tr>
<td>NH</td>
<td>825</td>
<td>937</td>
</tr>
<tr>
<td>NJ</td>
<td>260</td>
<td>371</td>
</tr>
<tr>
<td>NY</td>
<td>604</td>
<td>715</td>
</tr>
<tr>
<td>PA</td>
<td>49</td>
<td>159</td>
</tr>
<tr>
<td>RI</td>
<td>482</td>
<td>593</td>
</tr>
<tr>
<td>VT</td>
<td>825</td>
<td>937</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>5,749</td>
<td>7,000</td>
</tr>
</tbody>
</table>