

Clinical trials management on the internet — I. Using L^AT_EX and SAS to produce customized forms

Paul A. Thompson, Ph.D.

Email `patjah@sbcglobal.net`

Abstract In clinical trials, forms are used to gather data which is then entered into a database. Paper-based forms are still the standard for data collection, due to portability, stability, and storage considerations. In producing forms, SAS/IntrNet (a SAS product which works with the Internet) is used to facilitate the entry of information about participants in a clinical trial over the internet. Using L^AT_EX, the forms are then processed to produce a .pdf file. The .pdf is returned to the requesting party using a return page on the web browser. The entire process takes about 20 seconds. The system allows highly customized forms to be produced, in which values are inserted into appropriate locations on the forms. L^AT_EX is important due to its superior scripting capabilities, while SAS provides a very flexible database to pull information to be inserted into the forms, as well as providing a method for scripting up the entire transaction. The code required for the process and general approach is outlined.

Introduction

In the process of managing clinical trials, data are collected from subjects or participants and then entered into a data management system. While some clinical trials use full on-line data entry using computerized, on-line forms, these are uncommon as yet, since there is no guarantee that the information is correctly entered nor that complete forms are obtained. For these reasons, most clinical trials today continue to use paper forms for data collection.

Paper forms remain an excellent method for data collection. They are easily moved from location to location, support a wide variety of types of data for entry, and can be filled out by several persons at the same time under a variety of conditions. They can be printed in multi-color or in monochrome, and can include illustrations if necessary. They can be stored in many ways, including in binders, folders, and boxes. They can be placed

in long-term storage off-site. They can be copied into computerized photographic memory using a scan process. Paper forms can be taken into locations that computers are still not able to be used in (inside MR scan rooms, in damp environments, in very hot locations, in situations in which a computer is at risk). In short, they offer great advantages for data collection.

One important problem with packages of forms is the production, storage, and management of forms for large clinical experiments. When large clinical experiments are run, the management of forms can be difficult. Different visits can have different sets of forms for different types of data. In some cases, a particular form may have several versions, in which some versions are appropriate for some visits, and other versions are appropriate for other visits. The forms additionally often need specialized printing on them, which is individually devised for each form separately and for each different individual. For instance, samples are often tagged with accession numbers (a specific tracking number for the particular sample), which is used to specifically track the processing, storage, availability, and location of the sample. The accession number must be maintained on the form without error, because if the number is lost or misprinted, the sample is lost and will not be able to be located in the future.

If forms are printed at the start of the trial, a very large number of forms will need to be printed. In one trial which is managed by the Division of Biostatistics, the forms needed for a single individual comprise more than 1000 pages. Printing all these forms in a gang-printing fashion can be done, but would result in the requirement of setting aside valuable office space for the storage of blank forms.

Internet-based data management

In the Division of Biostatistics of Washington University, we manage a number of clinical trials over the web. One of the main database tools used in our Division for data management is the SAS system ([SAS Institute, Inc, 2008](#)). This system includes data management tools, a full-featured database system (non-relational), statistical tools, matrix manipulation tools, and a variety of other components. One aspect of the SAS system of particular note is the macro system, which is a tool for self-modification and generation of SAS code from within the SAS system. The system also includes internet-management tools, most notably the SAS/IntrNet suite of products. SAS/IntrNet includes a component for receipt of form-based data (in which data can be stripped from HTML forms and converted into SAS macro variables) and a component for serving pages to client browsers (so that new form-based information can be transmitted from the SAS system running

SAS/IntrNet to the requesting location for the transaction, which is done using address information originally supplied on the original request).

For data management, a tool entitled the Web Data Management System (WDES; [Trinkauss et al. 2000](#), [Thompson et al. 2002](#), [Thompson 2003a](#), [Thompson 2003b](#)) is also used. The WDES is a macro-based tool which works with the SAS system ([SAS Institute, Inc, 2008](#)) to perform data management over the internet. The user in the system needs to have a browser, generally Mozilla Firefox. The user navigates to a Division-supported web portal page. Using this portal page, password-protected access to the system is made. Data entry from users is done using the FORMS section of a normal HTML form, using free-text entry into TEXT blanks, the selection of RADIO buttons, the checking of CHECKBOX items, and the selection of values from SELECT lists. When the form returns to the Division of Biostatistics data system, the values entered into the form data items are converted into SAS macro variable values by the SAS/IntrNet system. WDES functions as “middle-ware” within the system, performing the process of moving the data from the browser-CGI component into the SAS system in a standardized, consistent manner. Data are returned to the user by the construction of appropriate web pages by the SAS system.

“Just-in-time” form printing

Form printing for a trial often requires a “just-in-time” (JIT) approach, where forms are generated and then used within a short time in the life cycle of the trial. Using this approach, a participant who is due for an upcoming visit is selected, and forms are generated and printed. The forms are then used within a short time, generally under a week. The completed forms are then entered into the database, stored in a binder or other location, and corrected as needed.

The problem with such forms is that the infrastructure to support the printing of them is not easy to find or build. The tools for forms printing must be accessible for end users, who are usually nurses, physicians, or others whose jobs involve working with patients, not computers. Even for those who understand computer processing, the process of generating form is tedious and usually takes time. Additionally, forms must be prepared with specific information printed on the form, in the correct location, and with the correct values. The forms need ID numbers or names. They also need other specific ID numbers, dates, locations, and a variety of other types of information. Thus, what is required is a system for preparing form packets for printing, with important information pre-populated into the forms. The system must be easy to use, and must enable forms to be printed using standard printers.

One important idea is the notion of “scriptable” production methods. When documents are produced, most people immediately conclude that Microsoft Word will be used (that is, most people who do not read this journal). In using Word, the tool itself (MS Word) must be used as the editor for the document. Each document must be produced by a time-intensive process. Multiple documents can be produced from a single master using a “mail-merge” methodology. For a more programming-type approach, programming languages must be used. These include Microsoft Visual Basic (a scriptable interface to Microsoft products) or the products of the T_EX family can be chosen. The T_EX family tools are particularly useful and scriptable methods for document production.

The term “scriptable” indicates that the document production method can be run by a script, which is a computer program that does not need human intervention, once it has been initiated, to perform a task. Scriptable production methods involves writing the script (possibly with various parameters for changing components to the document) and then running the script at an appropriate time. In this way, a scriptable document production system can produce documents during other processes, if the script can be run by the overall meta-process system manager.

Using the SAS system, the notion of scriptable forms production is quite easy to implement. A SAS program is composed of a string of database access commands, statistical evaluations of data from the databases, and other commands. Included in the other command set are commands which enable the user to “shell-out” or run commands at the level of the underlying operating system upon which the SAS system is run. These operating-system-level commands can be run in a synchronous or asynchronous manner. Synchronous commands pass commands to the outside program which runs until it is finished, at which time control returns to the calling program. Asynchronous programs are run by the generation of an additional copy of SAS which runs separately from the calling program. The use of asynchronous methods is sometimes called “spawning a program”, since the program that runs separately from the calling program is a separate and different program, with all the advantages and disadvantages inherent in that approach.

A solution to the JIT problem

In order to produce forms in a simple, JIT, easy-to-use manner, several tools are needed:

- A database which stores the important information. The SAS system is used.
- Relevant forms set up in a compatible format. Forms are set up using the PostScript-format.

- A scriptable document production system. The L^AT_EX system is used.
- An overall document scripting production engine for running the programs to provide forms. This entails an interface to the database, which can access relevant forms as required, and can actively run the scripts to produce final documents. The SAS system is used.
- a tool which can manage requests and serve information over the internet. The SAS system is used.

The SAS system (SAS Institute, Inc, 2008) offers an excellent choice for the database, the overall document scripting production engine, and the internet management tool. The SAS system offers a full database (although it is not a relational database), with a fully featured toolset. It also offers a variety of scripting capabilities, in that the programs written in the SAS language can be used to access SAS internal tools (for statistical processing, reporting, analysis, and other related components), and can access operating system-level tools such as external programs. In this way, the SAS program can be used to write a script to run the scriptable manuscript production system in the L^AT_EX system. In addition, the SAS system offers a very high-level macro processing system, which is essentially a method for self-modifying programs within the SAS system. Finally, the SAS/IntrNet system offers functionality for InterNet information management.

Several toolsets are available within the L^AT_EX system for forms preparation formats. In particular, the document formats devised by the Adobe company (encapsulated PostScript, .ps; portable document format, .pdf) are very useful. Encapsulated PostScript files are simply regular PostScript files in which a bounding box command (a command which defines the boundaries of the document in point values) has been added. These document formats are widely supported, and have a variety of third-party support products. These document production formats are also supported by browsers, so that a document represented using the .pdf system can be displayed in a very highly formatted manner within the browser. The fully-featured toolset (pstricks) in L^AT_EX for the handling of PostScript documents is quite useful. While pdftricks is also available, it is not currently as developed as pstricks, nor was it available when the methods described in this paper were set up. Conversion utilities (ps2pdf) ensure that documents set up in one system can be converted into the other, in a scriptable manner.

Producing forms on the fly

As part of the data management system for the HALT-PKD clinical evaluation of ACE/AARB medications for PKD, form packets are produced on the fly for participants for 20 different visits. For the baseline visits, a packet involving more than 20 forms is produced, which differ based on the condition. For later visits, form packets with varying numbers of forms are produced.

Steps to forms packet production

The basic steps for forms packet production are as follows:

1. User accesses forms request page on WDES portal. Participant ID, visit, and visit date are entered. Visit choice pre-populates forms selection, using JavaScript methods stored in a JavaScript file linked to the forms packet page. If a given form is not needed, it may be de-selected prior to packet production.
2. Forms request page is returned to the SAS system.
3. Forms generation process determines if accession numbers are required for the visit.

There are 2 basic types of accession numbers:

- (a) Sample identification numbers: there are a number of different types of samples, which are all identified within the sample accession number dataset by a series of index values; and
- (b) MR scan identification numbers: There are three different types of MR scans, identified by the accession number and scan type (MR, RBF, cardiac).

This is done by accessing the accession number database. Accession numbers are obtained and converted into SAS macro variables.

4. The form generation process is performed by writing L^AT_EX code to an external file. The code includes:
 - (a) Header code for the entire process;
 - (b) Form code, if the form is to be added, which initiates a page, and inserts the form;
 - (c) Code for the identification header;
 - (d) If the form is generated in a text-modified manner, code for inserted text;
 - (e) Completion text for the form; and
 - (f) Closing text to complete the packet.

5. A second script file is constructed which contains the commands to run the \LaTeX file is created.
6. The final `.pdf` file is moved to a location on the system which is associated with a `url` address.
7. SAS then completes processing by writing a page which includes a link tag which identifies the forms packet encoded as a `.pdf` file.
8. Page is returned to user for forms access. Forms may be examined, saved locally, printed, or ignored.

Required \LaTeX packages

There are several basic packages used for the forms production process. These packages are:

1. `geometry`: enables the dimensions of a page to be set in a simple interface.
2. `graphicx`: enables the import of graphical items.
3. `times`: use of the Times font family
4. `eso-pic`: adds pictures to the page. The picture can function as a wallpaper, which is the basic mechanism for forms production. The picture is read in, using the `\AddToShipoutPicture` command. All current picture values are flushed using `\ClearShipoutPicture`.
5. `pstricks`, `pstricks-add`: The `pstricks` family offers a flexible method of inserting graphical information into a file. The `pspicture` environment allows graphical items (including text) to be inserted at any location on a page. Inside the `pspicture` environment, the `\rput` command inserts text at a given location, with optional choices about orientation.
6. `\fbox`, `\parbox`: These are standard box-making tools available in base \LaTeX . `\fbox` produces a framed box. `\parbox` produces a paragraph box, which allows full use of multiple lines in the box.

\LaTeX code for portrait form production

Using SAS and \LaTeX together, forms can be produced using the algorithm defined above. The specific \LaTeX code required for the processing of a single portrait-oriented page for a form is shown in Figure 1. In this code, we see several things:

1. Line 1: New page is generated. Each form is placed on a single page.

2. Line 2: `eso-pic` package commands which clear any pictures from the picture store.
3. Line 3-5: `eso-pic` package commands which adds new picture found in File `picfile.ps`. This file must be an encapsulated PostScript file.
4. Lines 6-15: `pspicture` command block used to insert text into the picture space for the form. For a normal portrait orientation of a form (taller rather than wider), the specification shown is used. Landscape orientation form is shown in Figure 2.
5. Line 7: `pspicture` code to define the unit. Choices are `1in` (inch), `1pt` (point), or `1cm` (centimeter).
6. Line 8: `pspicture` code to insert text into the picture environment. The values inside the parentheses defines the coordinates of the center of the box.
7. Line 9-14: Header text box printed to identify the form. A `framebox` is placed around a `parabox`. The `parabox` allows multiple lines of text to be inserted. The `parabox` as shown is 1.05 inch in height and 4 inch in length. The `parabox` includes information about the participant ID Z1122334, visit identification, spaces to write in the actual date, a reference date printed on the form, and the form number. The `parabox`, `framebox`, and `\rput` commands are all closed on Line 14 (3 separate boxes, and three separate close braces).

```

1 \newpage
2 \ClearShipoutPicture
3 \AddToShipoutPicture
4 {\includegraphics[natheight=11in,natwidth=8.5in]
5 {picfile.ps}}
6 \begin{pspicture}(0,0)(8.5,11)
7 \psset{unit=1in}
8 \rput(5.4,3.55){
9 \fbox{\parbox[t][1.05in][t]{4in}{
10 \textbf{\LARGE HALT-PKD ID: Z1122334} \\
11 \textbf{Visit: Base} - \textbf{Date: ___/___/___} \\
12 \textbf{Form Date of Action: 6 / 13 / 2006} \\
13 \textbf{Form Number 09} }
14 }}}
15 \end{pspicture}

```

Code Block 1: Code for form page production for one portrait-oriented page

L^AT_EX code for landscape forms with special values

The specific L^AT_EX code required for the processing of a single landscape-oriented page for a form is shown in Figure 2. Only differences will be discussed:

1. Line 3-5: `eso-pic` package commands which adds new picture found in File `picfile.ps`. This file must be an encapsulated PostScript file. The orientation of the form is redefined using the `natheight` and `natwidth` commands. These commands define a landscape page, which is short and wide, rather than tall and thin.
2. Lines 6-17: `pspicture` command block used to insert text into the picture space for the form. The orientation is again defined by values of the `pspicture` command.
3. Lines 15-16: `pspicture` command block used to insert additional special text into the form. This text is used, in this case, to insert special accession numbers into specific locations on the form. Note, unfortunately, that these locations are specified either by measuring the locations on the form with a ruler, or by using trial-and-error methods. I have mostly used trial-and-error methods to identify the correct locations on the form.

```
1 \newpage
2 \ClearShipoutPicture
3 \AddToShipoutPicture
4 {\includegraphics[natheight=8.5in,natwidth=11in]
5 {picfile.ps}}
6 \begin{pspicture}(0,0)(11,8.5)
7 \psset{unit=1in}
8 \rput(5.4,3.55){
9 \fbox{\parbox[t][1.05in][t]{4in}{
10 \textbf{\LARGE HALT-PKD ID: Z1122334}} \
11 \textbf{Visit: Base} - \textbf{Date: ___/___/___} \
12 \textbf{Form Date of Action: 6 / 13 / 2006} \
13 \textbf{Form Number 09} }
14 }}}
15 \rput(2.6,0.38){\Large \textbf{AC \# 45038148}}
16 \rput(2.6,-2.70){\Large \textbf{AC \# 29548797}}
17 \end{pspicture}
```

Code Block 2: Code for form page production for one landscape-oriented page

\LaTeX code for full document

The \LaTeX code required for the production of a packet of forms is shown in Figure 3. The specific code for a specific form is shown in brief:

1. Line 1: Initiation of \LaTeX
2. Lines 2-3: Inclusion of packages
3. Lines 4-5: Dimension definitions using `geometry`
4. Lines 6-11: Body of the document
5. Lines 7-8: Optional inclusion of other file; a listing of the files and signing sheet for overall packet are included in our project.
6. Lines 9-10: The code defined above in Figures 1 and 2 is included here.

```
1 \documentclass[dvips]{article}
2 \usepackage{geometry,graphicx,eso-pic,times}
3 \usepackage{xcolor,pifont,pstricks,pstricks-add}
4 \geometry{letterpaper,tmargin=0in,lmargin=0in}
5 \geometry{textheight=11in,textwidth=8.5in}
6 \begin{document}
7 \input file1
8 \input file2
9 CODE FOR FORM A - See Figure 1
10 CODE FOR FORM B - See Figure 2
11 \end{document}
```

Code Block 3: Code for form package production

Script file to process \LaTeX file

The \LaTeX file, as defined in Figures 1, 2, and 3, is processed using the script file shown in Figure 4. Although each command can be run directly as a “shell-out” command, it is better to write a script file, and then “shell-out” to process the script file.

```
1 #!/bin/sh
2 /usr/bin/latex texfile
3 /usr/bin/latex texfile
4 /usr/bin/dvips -o texfile.ps texfile
5 /usr/bin/ps2pdf texfile.ps texfile.pdf
```

Code Block 4: Code for form package production

Producing a forms packet

The separate code blocks to produce forms packets are shown above in Code Blocks 1, 2, 3, and 4. The steps to the production of a forms packet are outlined above in the section “Steps to forms packet production”. The final information needed is the SAS code needed for forms packet production. The SAS code is shown in Code Block 5.

Here is a brief description of some of the code:

1. Lines 4: The LIBNAME statement defines a reference to the database location
2. Lines 8-13: Access database containing accession numbers. Using the function PROC SYMPUT, create macro variables `_an1` and `_an2` to store these numbers. When a macro variable is created, it can be “resolved” (variable reference replaced by value) using the ampersand form.
3. Lines 17-43: The text used to create a forms packet containing one form is written a text file for later processing. The FILE statement designates the file to be used. The PUT statement writes information to that file.
4. Lines 34: This line writes variable information strings to the file, dependent on the values stored in the macro variables. The macro variable `&_haltid` contains the ID number of the person being processed. The value is stored in this macro variable by the SAS/IntrNet system as the page is returned from the web browser. When SAS encounters the ampersand form `&_haltid`, the name is replaced by the value; this process is termed “macro resolution.” The SAS macro system is quite similar to the L^AT_EX system in the substitution process.
5. Lines 35, 36, 39, 40: These lines write variable information strings to the file (`&_visit`, `&_actd`, `&_actm`, `&_acty`, `&_an1`, `&_an2`). These macro variables contain information either inserted by the SAS/IntrNet system, or by actions which occur earlier in the file production process. The process is similar to

that on Line 34. The macro variables in Lines 39 and 40 were created in Lines 11 and 12 by CALL SYMPUT.

6. Lines 47-55: Create a script file to process the L^AT_EX code. The latex command is issued twice to clear unresolved issues from the aux file. The dvips command converts the result into a .ps file. The ps2pdf command converts the .ps file into a .pdf file. The final command moves the file to a URL-addressable location.
7. Lines 60-61: Set permissions and process script file. The command X "command" executes the command in the quotes at the operating system level.
8. Lines 65-90: Create and return the page containing the reference to the .pdf file. Note the mix of single and double quotes; there are always difficulties with quoting.
9. Line 66: Send the page back to the pre-set destination _WEBOUT. This is a special SAS file reference, which uses the current URL to return the page to the originating

```

1  /* ***** */
2  /* SAS symbolic directory name */
3  /* ***** */
4  LIBNAME HPKD "/users/data/halt";
5  /* ***** */
6  /* Read resource file and convert accession */
7  /*          numbers to macros */
8  /* ***** */
9  DATA _NULL_;
10     SET HPKD.ACCNFORM;
11     WHERE (HALTID = "&_haltid");
12     IF (SAMP = 2) THEN CALL SYMPUT("_an1",PUT(ACCN,8.));
13     IF (SAMP = 3) THEN CALL SYMPUT("_an2",PUT(ACCN,8.));
14  RUN;
15  /* ***** */
16  /* Create file containing LaTeX code to produce */
17  /*          forms packet */
18  /* ***** */
19  DATA _NULL_;
20     FILE "/users/halt/scripts/texfile.tex";
21     PUT "\documentclass[dvips]{article}";
22     PUT "\usepackage{geometry,graphicx,eso-pic,times}";
23     PUT "\usepackage{xcolor,pifont,pstricks,pstricks-add}";

```

```

24     PUT "\geometry{letterpaper,tmargin=0in,lmargin=0in}";
25     PUT "\geometry{textheight=11in,textwidth=8.5in}";
26     PUT "\begin{document}";
27     PUT "\newpage";
28     PUT "\ClearShipoutPicture";
29     PUT "\AddToShipoutPicture";
30     PUT "{\includegraphics[natheight=8.5in,natwidth=11in]";
31     PUT "{picfile.ps}}";
32     PUT "\begin{pspicture}(0,0)(11,8.5)";
33     PUT "\psset{unit=1in}";
34     PUT "\rput(5.4,3.55) {";
35     PUT "\fbox{\parbox[t][1.05in][t]{4in}}{";
36     PUT "\textbf{\LARGE HALT-PKD ID: &_haltid}}\\";
37     PUT "\textbf{Visit: &_visit - Date:___/___/___}}\\";
38     PUT "\textbf{Form Date of Action:&_actm/&_actd/";
39         &_acty}}\\";
40     PUT "\textbf{Form Number 09 } ";
41     PUT "}}}}";
42     PUT "\rput(2.6,0.38){\Large \textbf{AC \# &_an1}}";
43     PUT "\rput(2.6,-2.70){\Large \textbf{AC \# &_an2}}";
44     PUT "\end{pspicture}";
45     PUT "\end{document}";
46 RUN;
47 /* ***** */
48 /* Produce script file for subsequent processing */
49 /* ***** */
50 DATA _NULL_;          * Make file for forms packet script;
51     FILE "/users/halt/scripts/curscript";
52     PUT "#!/bin/sh";
53     PUT "/usr/bin/latex texfile";
54     PUT "/usr/bin/latex texfile";
55     PUT "/usr/bin/dvips -o texfile.ps texfile";
56     PUT "/usr/bin/ps2pdf texfile.ps texfile.pdf";
57     PUT "cp texfile.pdf /users/public/forms";
58 RUN;
59

```

```

60 /* ***** */
61 /* Set script file with proper permissioning, and */
62 /*          execute script */
63 /* file from within SAS */
64 /* ***** */
65 x "chmod 550 curscript"; * Set permission to executable ;
66 x "curscript";
67 /* ***** */
68 /* Prepare web page and transmit to calling program */
69 /* ***** */
70 DATA _NULL_;
71     FILE _WEBOUT;          * _WEBOUT is special
72                          * web-destination name;
73     PUT '!DOCTYPE HTML PUBLIC "-//W3C//DTD
74         HTML 4.0//EN">';
75     PUT '<HTML><HEAD><TITLE>Forms Processing -
76         HALT-PKD</TITLE>';
77     PUT '<SCRIPT LANGUAGE="JavaScript" '
78         ' SRC="/haltpkd/java/mainjava.js"></SCRIPT>';
79     PUT '<SCRIPT LANGUAGE="JavaScript" '
80         ' SRC="/haltpkd/java/formjava.js" ></SCRIPT>';
81     PUT '<LINK REL=STYLESHEET type="text/css" '
82         ' HREF="/haltpkd/css/maincss.css"></LINK>';
83     PUT "</HEAD><BODY>";
84     PUT '<FORM ACTION="/cgi-bin/haltpkd/broker" '
85         ' METHOD="POST" name="formform">';
86     PUT '<INPUT TYPE="HIDDEN" NAME="_brkname" '
87         ' VALUE="/cgi-bin/haltpkd/broker">';
88     PUT "<BR>";
89     PUT '<INPUT TYPE="HIDDEN" NAME="_program"
90         VALUE="xpkdtn.fl.sas">';
91     PUT '<INPUT TYPE="HIDDEN" NAME="_service"
92         VALUE="haltpkd">';
93     PUT '<INPUT TYPE="HIDDEN" NAME="_debug" VALUE="0">';
94     PUT '<input TYPE="HIDDEN" NAME="_prtype"
95         VALUE="2">';

```

```

96     PUT "<H3>HALT-PKD: ID &_haltid, Visit
97         &_visit</H3><BR>";
98     PUT "<H3>Current forms package</H3><BR>";
99     PUT '<A HREF="/haltpkd/forms/rF8801952_F21.pdf">';
100    PUT "Forms packet in PDF format</A><BR><BR>";
101    PUT "</FORM></BODY></HTML>";
102    RUN;

```

Code Block 5: SAS Code for overall packet production

Conclusions

Methods for management of forms packet production over the web require the integration of technologies for database management, text production, and web technologies. The entire process must be integrated together using a scripting tool that can facilitate the three components noted above. The SAS system provides a convenient platform for all of the components except the text production. The text production component can be nicely managed using \LaTeX and documents encoded using the encapsulated PostScript format. Using the scripting capabilities of the SAS system, the entire process of

1. making a request for a forms packet;
2. getting information from a database to support the request;
3. producing the code to make the forms packet;
4. producing the code to run the forms packet code; and
5. returning the final forms packet to the requesting user

can be accomplished within a single program, in real time.

Although it is difficult to demonstrate the use of this system in a text document, the system for the HALT-PKD trial was used to generate a number of forms packets. Time for forms packet generation is shown in Table 1 below.

Packet	Pages	Forms	Specials	Seconds
1	40	21	9	56
2	2	1	0	25
3	25	13	0	38
4	37	19	9	48

Table 1: Time for forms packet generation

References

SAS Institute, Inc (2008). *SAS OnlineDoc, Version 9.1.3*. Cary, NC, USA: SAS Institute Inc.

Thompson, P. A. (2003a). *The Web Data Entry System: Reference Guide, Version 2.5*.

Thompson, P. A. (2003b). *The Web Data Entry System: User's Guide, Version 2.5*.

Thompson, P. A., S. A. Littlewood, A. J. Adelman, and J. P. Miller (2002). *The Web Data Entry System: Methods for web data entry and SAS data management*. In SAS Institute Inc (Ed.), *Proceedings of the 27th Annual SAS Users Group International Meeting*. Cary, NC, USA: SAS INstitute Inc.

Trinkaus, K. M., P. A. Thompson, and J. P. Miller (2000). *Using SAS/IntrNet software to support distributed data entry*. In SAS Institute Inc (Ed.), *Proceedings of the 25th Annual SAS Users Group International Conference*, pp. 1212–1217. Cary, NC, USA: SAS Institute Inc.