

Nothing Can Go Wong

by

J. Michael Morrill, Shamrock Computer Resources

Introduction

Even the smallest application systems are comprised of thousands of events. Many of the events, such as data entry and data storage access may have unacceptable results better know as errors.

Errors have many favors, including data entry errors, file access errors, data base access errors and of course, system coding and design errors.

As system professionals, it is our task to determine the cause of these errors and correct them. However, many times we need to know not only what event triggered the error, but what events lead up to the error. Unfortunately few application systems are designed even log errors.

Error handling is usually done by the applications programmer who writes error handling code in an ad hoc manner. The error messages are cryptic and undocumented and the code that displays the error messages may itself be a source of errors. Furthermore, the error handling code detracts from task code. Task code is the code in a program that is written to fulfil the program's function.

What is needed is a standard method programmers can use to log important events and report errors that occur. ELF (Electronic Logging Facility) provides this needed standard.

Features

ELF has the following features:

- Dynamic merging of static text and real time data to create contextual error messages.
- Consistent display of application messages and consistent handling of application errors.
- Use of SAS macros to prevent the logging and error handling code from detracting from the task code.
- Use of common data areas to allow for generic logging and error handling routines.

ELF System Design

The ELF system architecture is embarrassingly simple. What is required to create the system is SAS/AF® and the will to do so.

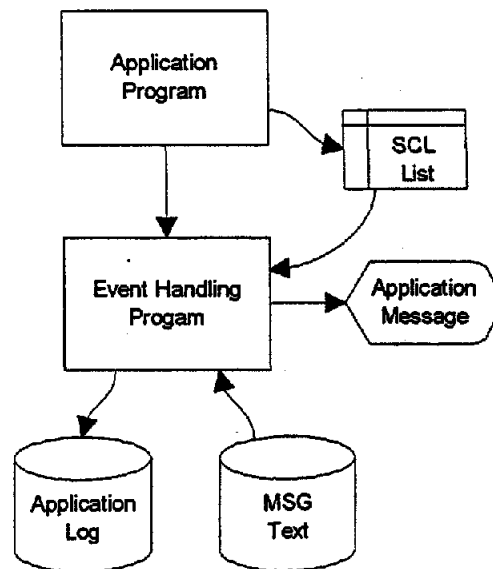


Figure I ELF system flow

The system consist of 4 major parts:

1. The message text file.
2. The log file.
3. The event handling program.
4. The application interface code.

Figure I is the system flow for the ELF system. To log an event the **Application Program** calls the Event Handling Program via a *CALL DISPLAY* AF program statement. Instead of passing data via the *CALL DISPLAY* statement, the application program stores the data in a SCL list. Each data item is a character named item stored in the SCL list. SCL lists are explained in detail the SAS/AF reference manual.

The **Event Handling Program** retrieves the data it needs from the SCL list. First the program retrieves the message number. The program accesses the **MSG (Message) Text** file to retrieve the message text associated with the message number. The program displays the **Application Message** and writes the message to the **Application Log**.

Text Templates and Data Merging

The must advance feature of the ELF system is the text templates.

This is page {Page} of {Total_pages}.
Template

Page = 12 Total_pages = 20
Data

This is page 12 of 20.
Merged Text

Figure 2 Example of Text Templates

The example shows how a text template operates. A template consisting of static text and variables is merged with data to create a merged text string.

A text template is a patten used to create contextual messages. The following scenario

illustrates this point. A data entry field may only contain the values A, Q, W and C. However the user entered Z into the field. The application program detects the error. The program stores the contents of the field in a named item in a SCL list. The name of the named item is F_CONTENTS. The program also stores the error message code 'E073' in the same SCL list. The name of the SCL item storing the error message code is G_MSG.

SCL List	
Named Item Name	Contents
F_CONTENTS	Z
G_MSG	E073

The next step is for the application program to call the Event Handling Program. The handling program retrieves the contents of G_MSG, then accesses the MSG Text file to retrieve the template.

ERROR: You must enter A, Q, W or C. You have entered {F_CONTENTS}.
--

Template for E073

The event program parses the template. It uses the text between the two braces as the name of an named item stored in the SCL list. It retrieves the contents of the named items then uses the contents to replace the braces and the text between the braces. The result is the message

ERROR: You must enter A, Q, W or C. You have entered Z.
--

is displayed to the user. The last step is to write a copy of the message to the log.

The Interface Macro

Every time an application program calls the event handling program, it must store the code for the template in a SCL list, then execute a *CALL DISPLAY* statement. Therefore this code can be

contained in a macro. The macro can be coded as follows:

```
%MACRO EVENT (MSG=S000);  
RC = INSERTC(ENVLIST('L'), "&MSG",  
"G_MSG");  
CALL DISPLAY("EVH.PROGRAM");  
%MEND EVENT;
```

In the last example, the code in the application program to display the error message would have been: %EVENT (MSG=E073);

The MSG Text File

The MSG Text file contains three fields (a.k.a. columns). The fields are the template code, the template text and the process control code.

The template code is a four character field. The first character is usually and 'E', which stands for 'ERROR'. The last three characters are a sequential number assigned as each new template is created. The template code field is a unique key field for the MSG text file.

The process control code contains a value of 'D', 'L' or 'B'. The values stand for DISPLAY, LOG and BOTH respectively. When the event process code is 'D', the merged template text is displayed to the user. When the code is 'L', the text is written to the log but is not displayed. When the code is 'B' the text is both displayed and logged.

The Application Log File

The Application Log file contains three fields. The fields are record timestamp, template code and merged text.

The record timestamp field contains the date and time the log record was written. Many data base management systems have a column type of TIMESTAMP. Use this type for this field. The record timestamp field is a unique key field for the Application Log file.

The template code field contains the template code used to create the log message. The merged text field is the template text that has been merged with the real time data.

Application system support people can use the application log to find trouble spots. For example, an analysis of the data from the application log may show that the users are making the one mistake far more the all the others. That may signal the need to redesign a screen to make it easier to use.

The Event Handling Program

The event handling program is actually two programs. One program handles the access to the template and log files. The other program displays the message. The first program, known as the control program is a SCL program that has no display screen (or a display screen with no fields). Below is pseudo code for the control program.

```
GET TEMPLATE CODE FROM SCL LIST  
GET TEMPLATE TEXT FROM MSG TEXT FILE  
MERGE TEXT TEMPLATE WITH DATA FROM  
SCL LIST  
IF PROCESS CONTROL IS DISPLAY OR BOTH  
CALL MESSAGE DISPLAY PROGRAM  
ENDIF  
IF PROCESS CONTROL IS LOG OR BOTH  
WRITE LOG  
ENDIF
```

The message display program is a simple program that displays merged text. After the text is displayed, the program waits for the user to press the enter key, then the program returns control to the control program.

Future Directions

As this paper is being written, the simple concepts of this system are being enhanced and used in a very exciting way.

Text templates are being enhance to allow text templates to be imbedded within other text templates. Processing flags (like the ones found on

most UNIX commands) are being added to templates.

A descendant of the event handling program has been written to process a simple procedure language. The result is a 'stored procedure' layer is now is not only possible, but is actually being use.

A store procedure layer, allows a developer to write the data base manipulation routines independently of the user interface routines.

This development will the subject of one or more future papers.

About the Author

J. Michael Morrill is a consultant with Shamrock Computer Resources. He can be reached at 0004856878@MCIMAIL.COM.

SAS and SAS/AF are registered trademarks or trademarks of SAS Institute Inc. In the USA and other countries. ® indicates USA registration.