

An Introduction to SAS® Hash Programming Techniques

Kirk Paul Lafler, Software Intelligence Corporation, Spring Valley, California

Abstract

SAS® users are always interested in learning techniques that will help them improve the performance of table lookup, search, and sort operations. SAS software supports a DATA step programming technique known as a hash object to associate a key with one or more values. This presentation introduces what a hash object is, how it works, the syntax required, and simple applications of its use. Essential programming techniques will be illustrated to sort data and search memory-resident data using a simple key to find a single value.

Introduction

One of the more exciting and relevant programming techniques available to SAS users today is the Hash object. Available as a DATA step construct, users are able to construct relatively simple code to perform match-merge and/or join operations. The purpose of this paper and presentation is to introduce the basics of what a hash table is and to illustrate practical applications so SAS users everywhere can begin to take advantage of this powerful Base-SAS programming feature.

Example Tables

The data used in all the examples in this paper consists of a Movies table containing six columns: title, length, category, year, studio, and rating. Title, category, studio, and rating are defined as character columns with length and year being defined as numeric columns. The data stored in the Movies table is shown below.

MOVIES Table

| | Title | Length | Category | Year | Studio | Rating |
|----|-----------------------------|--------|----------------------|------|--------------------|--------|
| 1 | Brave Heart | 177 | Action/Adventure | 1995 | Paramount Pictures | R |
| 2 | Casablanca | 103 | Drama | 1942 | MGM / UA | PG |
| 3 | Christmas Vacation | 97 | Comedy | 1989 | Warner Brothers | PG-13 |
| 4 | Coming to America | 116 | Comedy | 1988 | Paramount Pictures | R |
| 5 | Dracula | 130 | Horror | 1993 | Columbia TriStar | R |
| 6 | Dressed to Kill | 105 | Drama/Mysteries | 1980 | Filmways Pictures | R |
| 7 | Forrest Gump | 142 | Drama | 1994 | Paramount Pictures | PG-13 |
| 8 | Ghost | 127 | Drama/Romance | 1990 | Paramount Pictures | PG-13 |
| 9 | Jaws | 125 | Action/Adventure | 1975 | Universal Studios | PG |
| 10 | Jurassic Park | 127 | Action | 1993 | Universal Pictures | PG-13 |
| 11 | Lethal Weapon | 110 | Action/Cops & Robber | 1987 | Warner Brothers | R |
| 12 | Michael | 106 | Drama | 1997 | Warner Brothers | PG-13 |
| 13 | National Lampoon's Vacation | 98 | Comedy | 1983 | Warner Brothers | PG-13 |
| 14 | Poltergeist | 115 | Horror | 1982 | MGM / UA | PG |
| 15 | Rocky | 120 | Action/Adventure | 1976 | MGM / UA | PG |
| 16 | Scarface | 170 | Action/Cops & Robber | 1983 | Universal Studios | R |
| 17 | Silence of the Lambs | 118 | Drama/Suspense | 1991 | Orion | R |
| 18 | Star Wars | 124 | Action/Sci-Fi | 1977 | Lucas Film Ltd | PG |
| 19 | The Hunt for Red October | 135 | Action/Adventure | 1989 | Paramount Pictures | PG |
| 20 | The Terminator | 108 | Action/Sci-Fi | 1984 | Live Entertainment | R |
| 21 | The Wizard of Oz | 101 | Adventure | 1939 | MGM / UA | G |
| 22 | Titanic | 194 | Drama/Romance | 1997 | Paramount Pictures | PG-13 |

The second table used in the examples is the ACTORS table. It contains three columns: title, actor_leading, and actor_supporting, all of which are defined as character columns, and is illustrated below.

ACTORS Table

| | Title | Actor_Leading | Actor_Supporting |
|----|-----------------------------|-----------------------|------------------|
| 1 | Brave Heart | Mel Gibson | Sophie Marceau |
| 2 | Christmas Vacation | Chevy Chase | Beverly D'Angelo |
| 3 | Coming to America | Eddie Murphy | Arsenio Hall |
| 4 | Forrest Gump | Tom Hanks | Sally Field |
| 5 | Ghost | Patrick Swayze | Demi Moore |
| 6 | Lethal Weapon | Mel Gibson | Danny Glover |
| 7 | Michael | John Travolta | Andie MacDowell |
| 8 | National Lampoon's Vacation | Chevy Chase | Beverly D'Angelo |
| 9 | Rocky | Sylvester Stallone | Talia Shire |
| 10 | Silence of the Lambs | Anthony Hopkins | Jodie Foster |
| 11 | The Hunt for Red October | Sean Connery | Alec Baldwin |
| 12 | The Terminator | Arnold Schwarzenegger | Michael Biehn |
| 13 | Titanic | Leonardo DiCaprio | Kate Winslet |

What is a Hash Object?

A hash object is a data structure that contains an array of items that are used to map identifying values, known as keys (e.g., employee IDs), to their associated values (e.g., employee names or employee addresses). As implemented, it is designed as a DATA step construct and is not available to any SAS PROCedures. The behavior of a hash object is similar to that of a SAS array in that the columns comprising it can be saved to a SAS table, but at the end of the DATA step the hash object and all its contents disappear.

How Does a Hash Object Work?

A hash object permits table lookup operations to be performed considerably faster than other available methods found in the SAS system. Unlike a DATA step merge or PROC SQL join where the SAS system repeatedly accesses the contents of a table stored on disk to perform table lookup operations, a hash object reads the contents of a table into memory once allowing the SAS system to repeatedly access it, as necessary. Since memory-based operations are typically faster than their disk-based counterparts, users generally experience faster and more efficient table lookup operations. The following diagram illustrates the process of performing a table lookup using the Movie Title (i.e., key) in the MOVIES table matched against the Movie Title (i.e., key) in the ACTORS table to return the ACTOR_LEADING and ACTOR_SUPPORTING information.

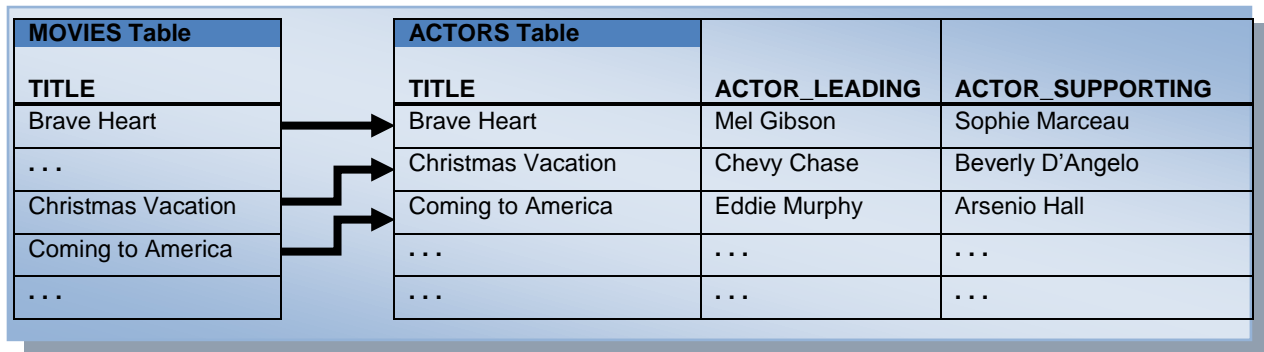


Figure 1. Table Lookup Operation with Simple Key

Although one or more hash tables may be constructed in a single DATA step that reads data into memory, users may experience insufficient memory conditions preventing larger tables from being successfully processed. To alleviate this kind of issue, users may want to load the smaller tables as hash tables and continue to sequentially process larger tables containing lookup keys.

Hash Object Syntax

Users with DATA step programming experience will find the hash object syntax relatively straight forward to learn and use. Available in all operating systems running SAS 9 or greater, the hash object is called using methods. The syntax for calling a method involves specifying the name of the user-assigned hash table, a dot (.), the desired method (e.g., operation) by name, and finally the specification for the method enclosed in parentheses. The following example illustrates the basic syntax for calling a method to define a key.

```
HashTitles.DefineKey ('Title');
```

where:

HashTitles is the name of the hash table, DefineKey is the name of the called method, and 'Title' is the specification being passed to the method.

Hash Object Methods

Under SAS 9, the author identifies twenty six (26) known methods. The following table illustrates an alphabetical list of the available methods.

| Method | Description |
|-------------------|---|
| ADD | Adds data associated with key to hash object. |
| CHECK | Checks whether key is stored in hash object. |
| CLEAR | Removes all items from a hash object without deleting hash object. |
| DEFINEDATA | Defines data to be stored in hash object. |
| DEFINEDONE | Specifies that all key and data definitions are complete. |
| DEFINEKEY | Defines key variables to the hash object. |
| DELETE | Deletes the hash or hash iterator object. |
| EQUALS | Determines whether two hash objects are equal. |
| FIND | Determines whether the key is stored in the hash object. |
| FIND_NEXT | The current list item in the key's multiple item list is set to the next item. |
| FIND_PREV | The current list item in the key's multiple item list is set to the previous item. |
| FIRST | Returns the first value in the hash object. |
| HAS_NEXT | Determines whether another item is available in the current key's list. |
| HAS_PREV | Determines whether a previous item is available in the current key's list. |
| LAST | Returns the last value in the hash object. |
| NEXT | Returns the next value in the hash object. |
| OUTPUT | Creates one or more data sets containing the data in the hash object. |
| PREV | Returns the previous value in the hash object. |
| REF | Combines the FIND and ADD methods into a single method call. |
| REMOVE | Removes the data associated with a key from the hash object. |
| REMOVEDUP | Removes the data associated with a key's current data item from the hash object. |
| REPLACE | Replaces the data associated with a key with new data. |
| REPLACEDUP | Replaces data associated with a key's current data item with new data. |
| SETCUR | Specifies a starting key item for iteration. |
| SUM | Retrieves a summary value for a given key from the hash table and stores the value to a DATA step variable. |
| SUMDUP | Retrieves a summary value for the key's current data item and stores the value to a DATA step variable. |

Sort with a Simple Key

Sorting is a common task performed by SAS users everywhere. The SORT procedure is frequently used to rearrange the order of dataset observations by the value(s) of one or more character or numeric variables. The SORT procedure is able to replace the original dataset or create a new ordered dataset with the results of the sort. Using hash programming techniques, SAS users have an alternative to using the SORT procedure. In the following example, a user-written hash routine is constructed in the DATA step to perform a simple ascending dataset sort. As illustrated, the metadata from the MOVIES dataset is loaded into the hash table, a DefineKey method specifies an ascending sort using the variable LENGTH as the primary (simple) key, a DefineData method to select the desired variables, an Add method to add data to the hash object, and an Output method to define the dataset to output the results of the sort to.

Hash Code with Simple Key

```
data _null_ ;
  if 0 then set movies; /* load variable properties into hash tables */
  if _n_ = 1 then do;
    declare Hash HashSort (ordered:'a'); /* declare the sort order for hash */
    HashSort.DefineKey ('Length'); /* identify variable to use as simple key */
    HashSort.DefineData ('Title',
                        'Length',
                        'Category',
                        'Rating'); /* identify columns of data */
    HashSort.DefineDone (); /* complete hash table definition */
  end;
  set movies end=eof;
  HashSort.add (); /* add data with key to hash object */
  if eof then HashSort.output(dataset:sorted_movies); /* write data using hash
                                                       HashSort */
run;
```

As illustrated in the following SAS Log results, SAS processing stopped with a data-related error due to one or more duplicate key values. As a result, the output dataset contained fewer results (observations) than expected.

SAS Log Results

```
data _null_ ;
  if 0 then set movies; /* load variable properties into hash tables */
  if _n_ = 1 then do;
    declare Hash HashSort (ordered:'a'); /* declare the sort order for hash */
    HashSort.DefineKey ('Length'); /* identify variable to use as simple key */
    HashSort.DefineData ('Title',
                        'Length',
                        'Category',
                        'Rating'); /* identify columns of data */
    HashSort.DefineDone (); /* complete hash table definition */
  end;
```

SAS Log Results (Continued)

```
set movies end=eof;

HashSort.add (); /* add data with key to hash object */

if eof then HashSort.output(dataset:'sorted_movies'); /* write data using hash
HashSort */

run;
```

ERROR: Duplicate key.

NOTE: The data set WORK.SORTED_MOVIES has 21 observations and 4 variables.

NOTE: The SAS System stopped processing this step because of errors.

NOTE: There were 22 observations read from the data set WORK.MOVIES.

Sort with a Composite Key

To resolve the error presented in the previous example, an improved and more uniquely defined key is specified. The simplest way to prevent a conflict consisting of duplicate is to add a secondary variable to the key creating a composite key. The following code illustrates constructing a composite key with a primary variable (LENGTH) and a secondary variable (TITLE) to reduce the prospect of producing a duplicate key value from occurring (collision).

Hash Code with Composite Key

```
data _null_;
  if 0 then set movies; /* load variable properties into hash tables */
  if _n_ = 1 then do;
    declare Hash HashSort (ordered:'a'); /* declare the sort order for HashSort */
    HashSort.DefineKey ('Length', 'Title'); /* identify variables to use as
    composite key */
    HashSort.DefineData ('Title',
                        'Length',
                        'Category',
                        'Rating'); /* identify columns of data */
    HashSort.DefineDone (); /* complete HashSort table definition */
  end;
  set movies end=eof;
  HashSort.add (); /* add data with key to HashSort table */
  if eof then HashSort.output(dataset:sorted_movies); /* write data using hash
  HashSort */
run;
```

SAS Log Results

As shown on the SAS Log results, the creation of the composite key of LENGTH and TITLE is sufficient enough to form a unique key enabling the sort process to complete successfully with 22 observations read from the MOVIES dataset, 22 observations written to the SORTED_MOVIES dataset, and zero conflicts.

```

data _null_;
  if 0 then set movies; /* load variable properties into hash tables */
  if _n_ = 1 then do;
    declare Hash HashSort (ordered:'a'); /* declare the sort order for HashSort */

    HashSort.DefineKey ('Length', 'Title'); /* identify variable to use as
                                             composite key */

    HashSort.DefineData ('Title',
                        'Length',
                        'Category',
                        'Rating'); /* identify columns of data */
    HashSort.DefineDone (); /* complete HashSort table definition */
  end;
  set movies end=eof;
  HashSort.add (); /* add data using key to HashSort table */
  if eof then HashSort.output(dataset:'sorted_movies'); /* write data using
                                                         HashSort */
run;

```

NOTE: The data set WORK.SORTED_MOVIES has 22 observations and 4 variables.
NOTE: There were 22 observations read from the data set WORK.MOVIES.

Search and Lookup with a Simple Key

Besides sorting, another essential action frequently performed by users is the process of table lookup or search. The hash object as implemented in the DATA step provides users with the necessary tools to conduct match-merges (or joins) of two or more datasets. Data does not have to be sorted or be in a designated sort order before use as it does with the DATA step merge process. The following code illustrates a hash object with a simple key (TITLE) to merge (or join) the MOVIES and ACTORS datasets to create a new dataset (MATCH_ON_MOVIE_TITLES) with matched observations.

```

data match_on_movie_titles(drop=rc);
  ❶ if 0 then set movies actors; /* load variable properties into hash tables */

  if _n_ = 1 then do;
    ❷ declare Hash MatchTitles (dataset:'actors'); /* declare the name MatchTitles
                                                    for hash */

    ❸ MatchTitles.DefineKey ('Title'); /* identify variable to use as key */
    MatchTitles.DefineData ('Actor_Leading',
                            'Actor_Supporting'); /* identify columns of data */
    MatchTitles.DefineDone (); /* complete hash table definition */
  end;

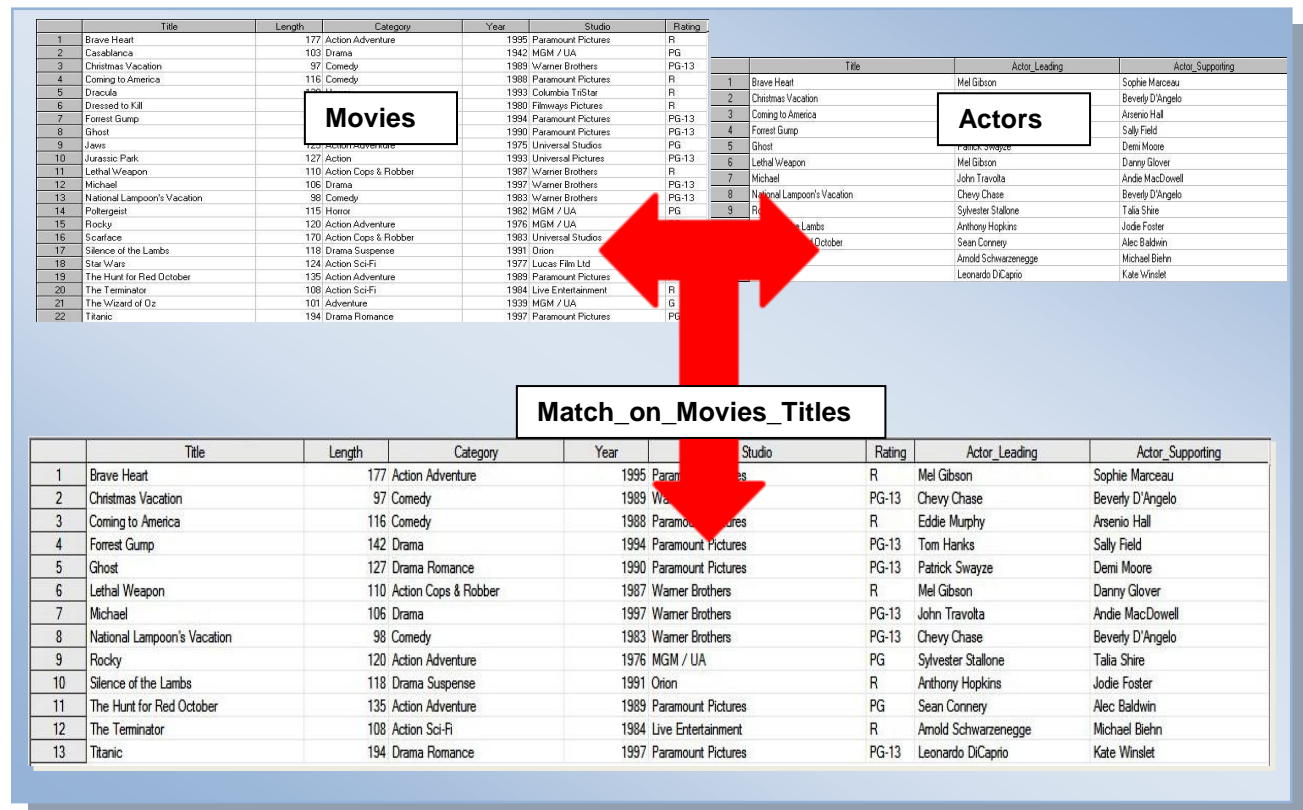
  set movies;

  ❹ if MatchTitles.find(key:title) = 0 then output; /* lookup TITLE in MOVIES table
                                                    using MatchTitles hash */
run;

```

Results

The match-merge (or join) process is illustrated using the following diagram.



Conclusion

Users have a powerful hash DATA-step construct to sort data, search datasets, and perform table lookup operations in the SAS system. This paper introduced the basics of what a hash table is, how it works, the basic syntax, and its practical applications so SAS users everywhere can begin to take advantage of this powerful memory-based programming technique to improve the performance of sorts, searches and table lookup operations.

References

- Dorfman, Paul, and Marina Fridman (2010). "*Black Belt Hashigana*," Proceedings of the 2010 North East SAS Users Group (SESUG) Conference.
- Dorfman, Paul and Peter Eberhardt (2010). "*Two Guys on Hash*," Proceedings of the 2010 South East SAS Users Group (SESUG) Conference.
- Dorfman, Paul (2009). "*The SAS® Hash Object in Action*," Proceedings of the 2009 South East SAS Users Group (SESUG) Conference.
- Dorfman, Paul, Lessia S. Shajenko and Koen Vyverman (2008). "*Hash Crash and Beyond*," Proceedings of the 2008 SAS Global Forum (SGF) Conference.
- Dorfman, Paul, and Koen Vyverman (2006). "*DATA Step Hash Objects as Programming Tools*," Proceedings of the Thirty-First SAS Users Group International Conference.
- Eberhardt, Peter (2011). "*The SAS® Hash Object: It's Time to .find() Your Way Around*," Proceedings of the 2011 SAS Global Forum (SGF) Conference.
- Lafler, Kirk Paul (2011). "An Introduction to SAS® Hash Programming Techniques," Proceedings of the 2011 South East SAS Users Group (SESUG) Conference.
- Lafler, Kirk Paul (2011). "An Introduction to SAS® Hash Programming Techniques," Proceedings of the 2011 PharmaSUG Conference.

- Lafler, Kirk Paul (2011). "An Introduction to SAS® Hash Programming Techniques," San Diego SAS Users Group (SANDS) Meeting, February 16th, 2011.
- Lafler, Kirk Paul (2010). "An Introduction to SAS® Hash Programming Techniques," Bay Area SAS (BASAS) Users Group Meeting, December 7th, 2010.
- Lafler, Kirk Paul (2010). "An Introduction to SAS® Hash Programming Techniques," Proceedings of the 2010 South Central SAS Users Group (SCSUG) Conference.
- Lafler, Kirk Paul (2010). "An Introduction to SAS® Hash Programming Techniques," Awarded "Best" Contributed Paper, Proceedings of the 2010 Western Users of SAS Software (WUSS) Conference.
- Lafler, Kirk Paul (2010). "DATA Step and PROC SQL Programming Techniques," Ohio SAS Users Group (OSUG) One-Day Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2010). "Exploring Powerful Features in PROC SQL," SAS Global Forum (SGF) Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2009). "DATA Step and PROC SQL Programming Techniques," South Central SAS Users Group (SCSUG) 2009 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2009). "DATA Step versus PROC SQL Programming Techniques," Sacramento Valley SAS Users Group 2009 Meeting, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2008). "What's Hot, What's Not – Skills for SAS® Professionals," Proceedings of the Second Annual SAS Global Forum (SGF) 2008 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Loren, Judy and Richard A. DeVenezia (2011). "Building Provider Panels: An Application for the Hash of Hashes," Proceedings of the 2011 SAS Global Forum (SGF) Conference.
- Loren, Judy (2006). "How Do I Love Hash Tables? Let Me Count The Ways!," Proceedings of the Nineteenth Northeast SAS Users Group Conference.
- Muriel, Elena (2007). "Hashing Performance Time with Hash Tables," Proceedings of the 2007 SAS Global Forum (SGF) Conference.
- Parman, Bill (2006). "How to Implement the SAS® DATA Step Hash Object," Proceedings of the 2006 Southeast SAS Users Group Conference.
- Ray, Robert and Jason Secosky (2008). "Better Hashing in SAS® 9.2," Proceedings of the Second Annual SAS Global Forum (SGF) Conference, SAS Institute Inc., Cary, NC, USA.
- Secosky, Jason (2007). "Getting Started with the DATA Step Hash Object," Proceedings of the 2007 SAS Global Forum (SGF) Conference, SAS Institute Inc., Cary, NC, USA.

Acknowledgments

The author would like to thank Kenny Bissett, SCSUG 2011 Beyond the Basics Section Chair for accepting my abstract and paper, as well as Lisa Mendez and Ed Bettie, SCSUG 2011 Conference Co-Chairs for a great conference!

Trademark Citations

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

About The Author

Kirk Paul Lafler is consultant and founder of Software Intelligence Corporation and has been using SAS since 1979. He is a SAS Certified Professional, provider of IT consulting services, trainer to SAS users around the world, and sasCommunity.org Advisory Board member. As the author of four books including PROC SQL: Beyond the Basics Using SAS, Kirk has written more than four hundred peer-reviewed papers, been an Invited speaker and trainer at more than three hundred SAS International, regional, local, and special-interest user group conferences and meetings throughout North America, and is the recipient of 18 "Best" contributed paper awards. His popular SAS Tips column, "Kirk's Korner of Quick and Simple Tips", appears regularly in several SAS User Group newsletters and Web sites, and his fun-filled SASword Puzzles is featured in SAScommunity.org.

Comments and suggestions can be sent to:

Kirk Paul Lafler
Senior Consultant, Application Developer, Trainer and Author
Software Intelligence Corporation
E-mail: KirkLafler@cs.com