Manipulating SCL Lists Available Through EIS Classes To Customize A Multidimensional Report

Jessica M. Gallegos, *Fireman's Fund Insurance Company; Novato, CA*
Matthew E. Westover, *Fireman's Fund Insurance Company; Novato, CA*

Abstract

The multidimensional report in SAS/EIS provides an interface which enables the construction of a simple cross-tabulation. A classification field or a drill-down hierarchy is crossed with another and a statistic is produced from either a SAS data file or a multidimensional database (MDDB). What happens when an elementary user desires to change the multidimensional report? This paper's goal is to provide examples of how to manipulate and customize the components within a multidimensional report at build time. Detailed SCL will follow on the use of methods and SCL lists available within specific SAS/EIS classes that allows customization of a multidimensional report. The final result is that an end user can access a multidimensional database through a multidimensional report, and alter the components through a completely transparent process.

Introduction

SAS/AF, SAS/EIS and SAS/MDDB products were chosen based upon an organizational business need. This need was to enable insurance adjusters and their managers the ability to query, subset, and manipulate the data that they produced with instantaneous results. The previous mode of accessing data in this organization was to email a request to the technical staff, and then await pending approval. Upon approval, an assigned analyst would generate the appropriate report and a hard copy would arrive via office mail. Needless to say, the process was time-consuming and exceptionally limiting to the end user. Using the SAS components, AF, EIS, and the MDDB server software, an online analytical processing (OLAP) tool was developed. This tool permitted the end user to instantaneously construct a report at their workstation, bypassing the lengthy and cumbersome process of the past.

Business Requirements

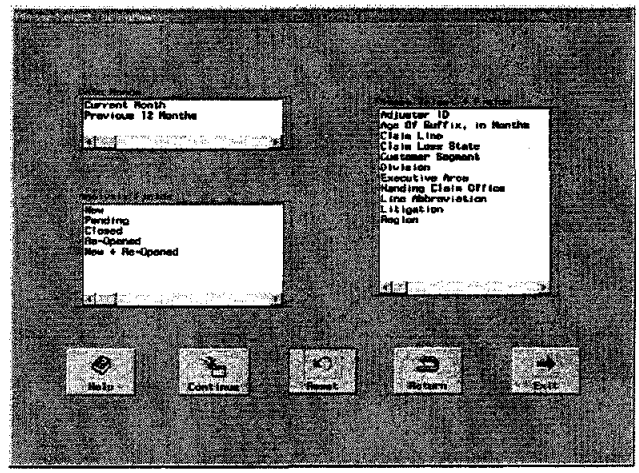
The primary requirement that end users requested for this newly created tool was first and foremost, ease of use. The typical end user has no technical training in regards to querying data from any type of database. The application therefore had to guide a user through their selection criteria and into the resulting report. The application literally walks end users through each step, defining both classification and analysis fields, therefore producing results which are neither misleading nor inaccurate. The ability to build a hierarchy or drill down order up to four levels, incorporating any of the classification fields defined within the database was also required. A five to ten second load and/or drill down time requirement for each report ensured that results would appear immediately, allowing an end user to seamlessly proceed to the next desired step.

Application Design

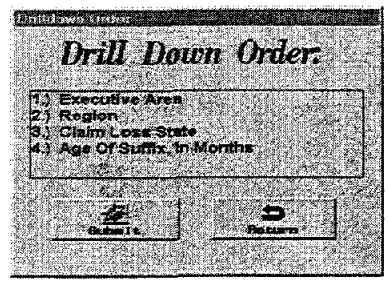
With the technical level of the users in mind, it was decided to keep the application as simple as possible. The user interface and selection criteria were designed in SAS/AF, outside of SAS/EIS. This was decided upon after preliminary testing of SAS's EIS interface produced results that the users could not easily comprehend. SAS/EIS's interface for manipulating multi-dimensional reports proved to be far too advanced for the typical end user at Fireman's Fund. An application was needed that allowed the functionality of a multidimensional report utilizing a multidimensional database, without overwhelming the end users.

In short, the goal was to develop an application that anyone could comfortably use to produce descriptive statistics from a multidimensional report. The full capability offered by a multidimensional report would be available, but in a controlled approach using a simplistic interface developed in SAS/AF and SAS's frame technology.

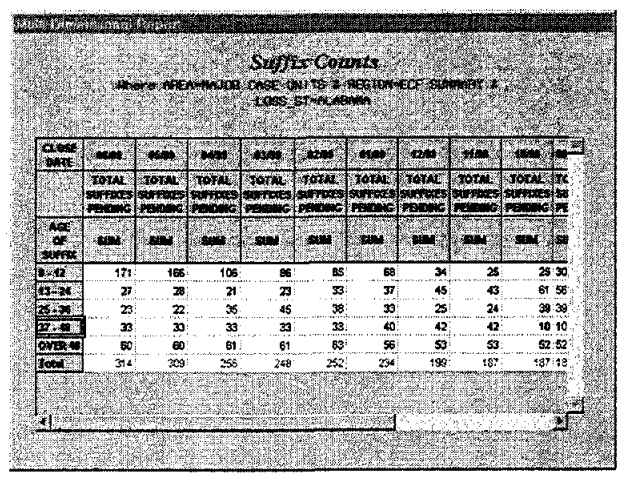
The user interface was developed in SAS/AF to include large simplistic list boxes, where the selection criteria is presented in a basic, easy to select layout. Three unique SAS/AF frames were created, each of which allows the end user to select the following: a specific view, layout, and analysis and classification fields. The final frame, below, prompts the end user to select a date range, an analysis field, and four classification fields. The four classification fields represent the drill down order to be displayed within the multidimensional report.



The drill down order is then immediately produced in a popup window for confirmation.



Finally, the multidimensional report will appear after the user submits his or her request. It is here during submission, where the multidimensional report is being generated, that modifications from compiled SCL code occur.



Modifying the Multidimensional Report

The SAS/EIS application database was built with twenty-two MULTI entries, one for each multi-dimensional database. Each MULTI entry has the potential of developing approximately 8,000 different hierarchies and 47,000 different tables. A table is defined as a single hierarchy crossed with a unique analysis field. For each MULTI entry, a default hierarchy and analysis field exist, however they have the ability to change according to the selection criteria defined by the end user. The resulting multidimensional report incorporates the user's selection criteria and all associated modifications at load time, thus being completely transparent to the end user.

The methods used for modification of the multidimensional report reside in the EMDDB_M and MDDB_M classes. The EMDDB_M class methods within this application interface with the metabase to modify the report hierarchies, drill down order, and exchange both analysis and classification fields. The MDDB_M class methods used within this application manipulate the layout of the report and the equations for computed values defined in the metabase. A sample listing of manipulations that could occur during the load time of the multidimensional report are as follows. Detailed SCL code for each can be found in the Appendix.

1.) Hierarchy Changes
2.) Redefining Drill Down Order Of The Hierarchy
3.) Changing The Default Analysis Fields
4.) Changing Default Computed Columns
5.) Redefining A Computed Column Equation
6.) Manipulation Of The POPUP Menu

Conclusion

By utilizing methods available through the EMDDB_M and MDDB_M classes, a multidimensional report can be modified at build time. The changes that occur to the report remain completely transparent to the end user. As such, an elementary user has the ability to develop a multidimensional report without the full knowledge of database dynamics or the logistics of querying a database.

References

SAS Institute Inc.(1998), *Customizing SAS/EIS Software Applications Course Notes*, Cary, NC: SAS Institute

Acknowledgments

Both of us would like to personally thank Ivy Parker at SAS Institute for her technical advise and support.

Contact Information

Jessica Gallegos
Claims, Finance & Planning
Fireman's Fund Insurance Company
777 San Marin Drive – C81
Novato, California 94998
Phone: (415) 899-3924
Fax: (415) 899-2388
Email: Jessica_M._Gallegos@ffic.com

Matthew Westover
Claims, Finance & Planning
Fireman's Fund Insurance Company
777 San Marin Drive – A20
Novato, California 94998
Phone: (415) 899-3159
Fax: (415) 899-2388
Email: Matthew_E._Westover@ffic.com

Trademark Citation

SAS/AF, SAS/EIS, SAS/MDDB are registered trademarks of SAS Institute, INC. in the USA and other countries.

Appendix

The following contains the SCL code and SCL lists used to complete the modifications to the multidimensional report. First, the selection criteria that the user defined in the AF frames must be accessible in the multidimensional report. The user criteria is defined as named items in a SCL list, named SEL_LIST.

```
SEL_LIST:  (DOMAIN_ID='D0990334'
            WORKSTATION_ID='W1114186'
            USER_ID='CLS2JMG'
            VIEW_SELECTION=0
            DSN=' '
            DSN_NAME=' '
            LAYOUT_SELECTION=0
            DATE_RANGE=0
            ANAL_LIST=(ANALYSIS_1=' '
                       ANALYSIS_2=' '
                       ANALYSIS_3=' '
                       ANALYSIS_4=' '
                       )[4]
            HIERARCHY=' '
            CHANGE_ORDER=0
            VAR_LIST=(VAR_ORD_1=' '
                      VAR_ORD_2=' '
                      VAR_ORD_3=' '
                      VAR_ORD_4=' '
                      )[6]
            DEFAULT=0
            )[2]
```

The selection list is inserted into the Global Environmental list, when submission occurs.

```
RC=LISTLEN(NEW_LIST);
IF RC=-1 THEN;
ELSE RC=DELLIST(NEW_LIST);
NEW_LIST=MAKELIST();
NEW_LIST=ENVLIST('G');
RC=SETNITEML(NEW_LIST,SEL_LIST,'SEL_LIST',1);
```

The selection list must be retrieved in a _POSTINIT_ override for that MULTI entry.

```
ENV_LIST=MAKELIST();
ENV_LIST=ENVLIST('G');
SEL_LIST=GETNITEML(ENV_LIST,'SEL_LIST');
```

1.) SCL For Hierarchy Changes

```
NEW_MDDB = TRIM(LEFT(GETNITEMC(SEL_LIST,'HIERARCHY')));
CALL SEND(_SELF_,'_SWAP_HIERARCHIES_',NEW_MDDB,'M_480','Y');
```

2.) SCL For Redefining Drill Down Order Of The Hierarchy

```
CHG_ORD  = GETNITEMN(SEL_LIST,'CHANGE_ORDER');
IF CHG_ORD=1 THEN LINK CHGORD;
              /*-------------------------------------------*/
              /* CODE TO CREATE NEW DRILL DOWN             */
              /*-------------------------------------------*/
CHGORD:

VS_1=TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'VAR_LIST'),'VAR_ORD
_1')));

VS_2=TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'VAR_LIST'),'VAR_ORD
_2')));

VS_3=TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'VAR_LIST'),'VAR_ORD
_3')));

VS_4=TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'VAR_LIST'),'VAR_ORD
_4')));
              /*-------------------------------------------*/
              /* OPEN METABASE                             */
              /*-------------------------------------------*/
  TEMPID=OPEN('EISLIB.META_1','I');
              /*-------------------------------------------*/
              /* GET HIERARCHY CATALOG ENTRY               */
              /*-------------------------------------------*/
  FULLDSN='SASLIB.' || TRIM(LEFT(DSN_NAME)) ;
  RC=WHERE(TEMPID,'DSETNAME=' || QUOTE(UPCASE(FULLDSN)) || ' AND
ATTRIBUT="HIERARCH"');
  ANY=ATTRN(TEMPID,'ANY');
  IF ANY=1 THEN DO;
    RC=FETCH(TEMPID);
    HIERLOC=GETVARC(TEMPID,VARNUM(TEMPID,'VALUE'));
  END;
  TEMPID=CLOSE(TEMPID);
  HIERS=MAKELIST();
  RC=FILLIST('CATALOG',HIERLOC,HIERS);
              /*-------------------------------------------*/
```

```
            /* CHECK IF NEW DRILL ORDER LIST EXISTS,      */
            /* EITHER POPULATE OR CHANGE ITEMS ACCORDINGLY*/
            /*                                           */
            /* PLEASE NOTE: FOR COMPUTED COLUMNS THAT    */
            /*              USE THE DEFAULT HIERARCHY,   */
            /*              THE DEFAULT HIERARCHY MUST   */
            /*              BE DEFINED IN ORDER FOR THE  */
            /*              COMPUTED COLUMNS TO BE CREATED*/
            /*-------------------------------------------*/
  ISIT=NAMEDITEM(HIERS,'XHIER');
  IF ISIT=0 THEN DO;                  *===> DOESN'T EXIST;
    RC=SETNITEML(HIERS,MAKELIST(),'XHIER',1);
    XH_LIST=GETNITEML(HIERS,'XHIER');
  END;
  ELSE DO;                            *===> EXIST;
    XH_LIST=GETNITEML(HIERS,'XHIER');
    COL_LIST=GETNITEML(XH_LIST,'COLUMNS');
  END;
  RC=SETNITEMC(XH_LIST,'XHIER','DESCRIPTION',1);
  RC=SETNITEMC(XH_LIST,' ','LEVELS',1);
  RC=SETNITEML(XH_LIST,MAKELIST(),'COLUMNS',1);
  COL_LIST=GETNITEML(XH_LIST,'COLUMNS');
  IF THREE IN ('221','211','222','212') THEN DO;
    IF NEW_MDDB='M_1' THEN DO;
      RC=SETNITEMC(COL_LIST,'AREA','AREA',1);
      RC=SETNITEMC(COL_LIST,'REGION','REGION',1);
      RC=SETNITEMC(COL_LIST,'HCO','HCO',1);
      RC=SETNITEMC(COL_LIST,'CLAIM_LN','CLAIM_LN',1);
      RC=SETNITEMC(COL_LIST,'LN_ABRV','LN_ABRV',1);
      RC=SETNITEMC(COL_LIST,'AGE','AGE',1);
    END;
    IF NEW_MDDB='M_480' THEN DO;
      RC=SETNITEMC(COL_LIST,'AREA','AREA',1);
      RC=SETNITEMC(COL_LIST,'REGION','REGION',1);
      RC=SETNITEMC(COL_LIST,'HCO','HCO',1);
      RC=SETNITEMC(COL_LIST,'AGE','AGE',1);
    END;
  END;
  ELSE DO;
    RC=SETNITEMC(COL_LIST,VS_1,VS_1,1);
    RC=SETNITEMC(COL_LIST,VS_2,VS_2,1);
    RC=SETNITEMC(COL_LIST,VS_3,VS_3,1);
    RC=SETNITEMC(COL_LIST,VS_4,VS_4,1);
  END;
  RC=SETNITEMC(XH_LIST,'-ALL-','TOPLEVEL',1);
  RC=SAVELIST('CATALOG',HIERLOC,HIERS);
RETURN;
```

3.) SCL For Changing The Default Analysis Fields

Retrieve the items which defines the new analysis fields and apply them to the _CHANGE_ANALYSIS_ method. Here NAME1-NAME4 are the new analysis fields. Please note the SCL list order and layout for the _CHANGE_ANALYSIS_ method to execute properly.

```
  NAME1                                                    =
TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'ANAL_LIST'),
'ANALYSIS_1')));
  NAME2                                                    =
TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'ANAL_LIST'),
'ANALYSIS_2')));
  NAME3                                                    =
TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'ANAL_LIST'),
'ANALYSIS_3')));
  NAME4                                                    =
TRIM(LEFT(GETNITEMC(GETNITEML(SEL_LIST,'ANAL_LIST'),
'ANALYSIS_4')));
            /*-------------------------------------------*/
            /* CREATE LIST WITH ANALYSIS VARIABLES       */
            /*-------------------------------------------*/
  TEMP_LIST=MAKELIST();
  RC=SETNITEMC(TEMP_LIST,NAME1,NAME1,-1);
  IF NAME2^=' ' THEN RC=SETNITEMC(TEMP_LIST,NAME2,NAME2,-1);
  IF NAME3^=' ' THEN RC=SETNITEMC(TEMP_LIST,NAME3,NAME3,-1);
  IF NAME4^=' ' THEN RC=SETNITEMC(TEMP_LIST,NAME4,NAME4,-1);
  CALL SEND(_SELF_,'_CHANGE_ANALYSIS_',TEMP_LIST);
```

4.) SCL For Changing Default Computed Columns

```
            /*-------------------------------------------*/
            /* DEFINED COMPUTED COLUMNS SO OVERRIDE HAS  */
            /* THE ABILITY TO SWITCH COMPUTED COLUMNS    */
            /* GET COMPUTED_L  LIST ID.  EXAMPLE ASSUMES */
            /* THAT NO COMPUTED VARIABLES ARE INITIALLY  */
            /* SELECTED                                  */
            /*-------------------------------------------*/
COMPCOL:
  COMP_L=GETNITEMN(MODELID,'COMPUTED_L');
  IF NAME1='TR' THEN RETURN;
                        /*  GET THE METABASE IDENTIFIER  */
  METABASEID=GETNITEMN(ENVLIST('L'),'METABASE');
             /* GET THE ATTRIBUTES OF THE COMPUTED COLUMN */
             /* CAN BE EITHER 1 OR 4 COMPUTED COLUMNS.    */
  RC=LISTLEN(LISTID1);
  IF RC=-1 THEN;  ELSE RC=DELLIST(LISTID1,'Y');
  LISTID1=MAKELIST();
  CALL
SEND(METABASEID,'_GET_VAR_ATTR_',RC,MSG,'EISLIB.META_1',NAME1,LI
STID1,FULLDSN);
  IF THREE IN ('221','211') THEN DO;
    RC=LISTLEN(LISTID2);
    IF RC=-1 THEN;  ELSE RC=DELLIST(LISTID2,'Y');
    LISTID2=MAKELIST();
    CALL
SEND(METABASEID,'_GET_VAR_ATTR_',RC,MSG,'EISLIB.META_1',NAME2,LI
STID2,FULLDSN);
    RC=LISTLEN(LISTID3);
    IF RC=-1 THEN;  ELSE RC=DELLIST(LISTID3,'Y');
    LISTID3=MAKELIST();
    CALL
SEND(METABASEID,'_GET_VAR_ATTR_',RC,MSG,'EISLIB.META_1',NAME3,LI
STID3,FULLDSN);
    RC=LISTLEN(LISTID4);
    IF RC=-1 THEN;  ELSE RC=DELLIST(LISTID4,'Y');
    LISTID4=MAKELIST();
    CALL
SEND(METABASEID,'_GET_VAR_ATTR_',RC,MSG,'EISLIB.META_1',NAME4,LI
STID4,FULLDSN);
  END;
              /* GET THE LIST CONTAINING THE COMPUTATION */
              /* AND PLACE IN COMP LISTID                */
  RC=LISTLEN(COMP1);
  IF RC=-1 THEN;  ELSE RC=DELLIST(COMP1,'Y');
  COMP1=MAKELIST();
  RC=FILLIST('CATALOG',GETNITEMC(LISTID1,'COMPUTED'),COMP1);
  IF THREE IN ('221','211') THEN DO;
    RC=LISTLEN(COMP2);
    IF RC=-1 THEN;  ELSE RC=DELLIST(COMP2,'Y');
    COMP2=MAKELIST();
    RC=FILLIST('CATALOG',GETNITEMC(LISTID2,'COMPUTED'),COMP2);
    RC=LISTLEN(COMP3);
    IF RC=-1 THEN;  ELSE RC=DELLIST(COMP3,'Y');
    COMP3=MAKELIST();
    RC=FILLIST('CATALOG',GETNITEMC(LISTID3,'COMPUTED'),COMP3);
    RC=LISTLEN(COMP4);
    IF RC=-1 THEN;  ELSE RC=DELLIST(COMP4,'Y');
    COMP4=MAKELIST();
    RC=FILLIST('CATALOG',GETNITEMC(LISTID4,'COMPUTED'),COMP4);
  END;
  RC=LISTLEN(COMPUTED);
  IF RC=-1 THEN;  ELSE RC=DELLIST(COMPUTED,'Y');
  COMPUTED=MAKELIST();
  RC=INSERTL(COMP_L,COMPUTED,-1,'_ANLSYS_');
              /* INSERT ALL INFORMATION NECESSARY INTO */
              /* COMPUTED_L LIST FOR THE FOR THE       */
              /* COMPUTED COLUMN(S) MAY BE 1 OR 4.     */
  RC=LISTLEN(C1_L);
  IF RC=-1 THEN;  ELSE RC=DELLIST(C1_L,'Y');
  C1_L=MAKELIST();
  RC=INSERTL(COMPUTED,C1_L,-1,NAME1);
  RC=INSERTC(COMP1,GETNITEMC(LISTID1,'VARLABEL'),-1,'LABEL');
  RC=INSERTC(COMP1,'_ANLSYS_',-1,'DIM_NAME');
  RC=INSERTN(COMP1,1,-1,'POSITION');
  RC=INSERTC(COMP1,'COMP COL',-1,'ORIGIN');
  RC=COPYLIST(COMP1,'N',C1_L);
  IF THREE IN ('221','211') THEN DO;
    RC=LISTLEN(C2_L);
    IF RC=-1 THEN;  ELSE RC=DELLIST(C2_L,'Y');
    C2_L=MAKELIST();
    RC=INSERTL(COMPUTED,C2_L,-1,NAME2);
    RC=INSERTC(COMP2,GETNITEMC(LISTID2,'VARLABEL'),-1,'LABEL');
    RC=INSERTC(COMP2,'_ANLSYS_',-1,'DIM_NAME');
    RC=INSERTN(COMP2,2,-1,'POSITION');
    RC=INSERTC(COMP2,'COMP COL',-1,'ORIGIN');
    RC=COPYLIST(COMP2,'N',C2_L);
    RC=LISTLEN(C3_L);
    IF RC=-1 THEN;  ELSE RC=DELLIST(C3_L,'Y');
    C3_L=MAKELIST();
    RC=INSERTL(COMPUTED,C3_L,-1,NAME3);
    RC=INSERTC(COMP3,GETNITEMC(LISTID3,'VARLABEL'),-1,'LABEL');
    RC=INSERTC(COMP3,'_ANLSYS_',-1,'DIM_NAME');
    RC=INSERTN(COMP3,3,-1,'POSITION');
    RC=INSERTC(COMP3,'COMP COL',-1,'ORIGIN');
    RC=COPYLIST(COMP3,'N',C3_L);
    RC=LISTLEN(C4_L);
    IF RC=-1 THEN;  ELSE RC=DELLIST(C4_L,'Y');
    C4_L=MAKELIST();
    RC=INSERTL(COMPUTED,C4_L,-1,NAME4);
    RC=INSERTC(COMP4,GETNITEMC(LISTID4,'VARLABEL'),-1,'LABEL');
    RC=INSERTC(COMP4,'_ANLSYS_',-1,'DIM_NAME');
    RC=INSERTN(COMP4,4,-1,'POSITION');
    RC=INSERTC(COMP4,'COMP COL',-1,'ORIGIN');
    RC=COPYLIST(COMP4,'N',C4_L);
  END;
              /* INVOKE THE METHODS IN THE FOLLOWING   */
              /* ORDER TO MAKE SURE VARIABLES EXIST    */
              /* AND ARE PROPERLY COMPUTED             */
  CALL SEND(MODELID,'_GENERATE_COMPUTED_SCL_');
  CALL
SEND(MODELID,'_ADD_COMPUTED_VALUE_','_ANLSYS_',GETNITEMC(LISTID1
,'VARLABEL'));
  COMP_L=DELNITEM(COMP_L,'_ANLSYS_');
  IF THREE IN ('221','211') THEN DO;
    CALL
SEND(MODELID,'_ADD_COMPUTED_VALUE_','_ANLSYS_',GETNITEMC(LISTID2
,'VARLABEL'));
    CALL
SEND(MODELID,'_ADD_COMPUTED_VALUE_','_ANLSYS_',GETNITEMC(LISTID3
,'VARLABEL'));
    CALL
SEND(MODELID,'_ADD_COMPUTED_VALUE_','_ANLSYS_',GETNITEMC(LISTID4
,'VARLABEL'));
  END;
  RC=LISTLEN(WORKLIST);
  IF RC=-1 THEN;  ELSE RC=DELLIST(WORKLIST,'Y');
```

```
  WORKLIST=MAKELIST();
  RC=INSERTC(WORKLIST,GETNITEMC(LISTID1,'VARLABEL'));
  IF THREE IN ('221','211') THEN DO;
    RC=INSERTC(WORKLIST,GETNITEMC(LISTID2,'VARLABEL'));
    RC=INSERTC(WORKLIST,GETNITEMC(LISTID3,'VARLABEL'));
    RC=INSERTC(WORKLIST,GETNITEMC(LISTID4,'VARLABEL'));
  END;
  CALL SEND(MODELID,'_APPLY_CUSTOM_',0,WORKLIST);
RETURN;
```

5.) SCL For Redefining A Computed Column Equation

```
              /*-------------------------------------------*/
              /* SWITCH VARIANCE EQUATION SUCH THAT NEW    */
              /* ANALYSIS FIELDS ARE DEFINED.              */
              /*-------------------------------------------*/
COL_EQN:
  COMP_L=GETNITEMN(MODELID,'COMPUTED_L');
  RC=LISTLEN(LISTID1);
  IF RC=-1 THEN;  ELSE RC=DELLIST(LISTID1,'Y');
  LISTID1=MAKELIST();
  RC=LISTLEN(WORKLIST);
  IF RC=-1 THEN;  ELSE RC=DELLIST(WORKLIST,'Y');
  WORKLIST=MAKELIST();
                        /* IF LOSS PAID OR TRADITIONAL RATIO */
                        /* THEN DEFAULT.                     */
  IF NAME1='PL_MON' THEN RETURN;
                              /*  GET THE METABASE IDENTIFIER  */
  MB_ID=GETNITEMN(ENVLIST('L'),'METABASE');
               /* GET THE ATTRIBUTES OF THE COMPUTED COLUMN */
  CALL
SEND(MB_ID,'_GET_VAR_ATTR_',RC,MSG,'EISLIB.META_1',NAME4,LISTID1
,FULLDSN);
                        /* CHANGE EQUATION TO INCLUDE NEW    */
                        /* FIELD NAMES                       */

RC=SETNITEMC(GETNITEML(GETNITEML(GETNITEML(GETNITEML(COMP_L,'_AN
LSYS_'),'TEMP'),
     'COMPUTED'),'2'),NAME2,'NAME');

RC=SETNITEMC(GETNITEML(GETNITEML(GETNITEML(GETNITEML(COMP_L,'_AN
LSYS_'),'TEMP'),
     'COMPUTED'),'4'),NAME3,'NAME');
                        /* INVOKE THE METHODS IN THE         */
                        /* FOLLOWING ORDER TO MAKE SURE      */
                        /* VARIABLES EXIST AND ARE PROPERLY  */
                        /* COMPUTED                          */
  CALL SEND(MODELID,'_GENERATE_COMPUTED_SCL_');
  RC=INSERTC(WORKLIST,GETNITEMC(LISTID1,'VARLABEL'));
  CALL SEND(MODELID,'_APPLY_CUSTOM_',0,WORKLIST);
RETURN;
```

6.) SCL For Manipulation Of The POPUP Menu

```
              /*-------------------------------------------*/
              /* DEFINE POPUP COMMANDS: GET LIST IDENTIFIERS */
              /* FROM APPLICATION LIST FOR POPUP COMMANDS  */
              /*-------------------------------------------*/
POPCMD:
  SELECT_L    = GETNITEML(APPLLIST,'SELECT_L');
  POPUP_L     = GETNITEML(SELECT_L,'POPUP');
  ANY_L       = GETNITEML(POPUP_L,'_ANYWHERE_');
  ACTIONSL    = GETNITEML(ANY_L,'POPUP_L');
  MAPL        = GETNITEML(ANY_L,'MAP_L');
              /*-------------------------------------------*/
              /* DEFINE HELP ON EIS POPUP SELECTION MENU   */
              /*-------------------------------------------*/
  RC=INSERTC(ACTIONSL,'HELP',-1);
  RC=SETLATTR(ACTIONSL,'ACTIVE',-1);
  RC=INSERTC(MAPL,'VIEWER',-1);
  RC=INSERTC(MAPL,'HELP_REPORT',-1);
  CALL SEND(_SELF_,'_SET_INSTANCE_METHOD_','HELP_REPORT',
            'CNALIB.CNA.POPUPCMD.SCL','HELP');
              /*-------------------------------------------*/
              /* EXPORT TO SPREADSHEET                     */
              /*-------------------------------------------*/
  RC=INSERTC(ACTIONSL,'EXPORT',-1);
  RC=SETLATTR(ACTIONSL,'ACTIVE',-1);
  RC=INSERTC(MAPL,'VIEWER',-1);
  RC=INSERTC(MAPL,'EXPORT_REPORT',-1);
  MID=GETNITEMN(GETNITEMN(APPLLIST,'VIEWERID'),'MODELID');
  CALL
SEND(_SELF_,'_SET_INSTANCE_METHOD_','EXPORT_REPORT','CNALIB.CNA.
POPUPCMD.SCL','EXPORT');
  CALL
SEND(MODELID,'_SET_INSTANCE_METHOD_','_WRITE_','CNALIB.CNA.POPUP
CMD.SCL','WRITE');
  CALL
SEND(MODELID,'_SET_INSTANCE_METHOD_','_BLDTBLE_','CNALIB.CNA.POP
UPCMD.SCL','BLDTBLE');
  CALL
SEND(MODELID,'_SET_INSTANCE_METHOD_','_NEW_','CNALIB.CNA.POPUPCM
D.SCL','NEW');
              /*-------------------------------------------*/
              /* DEFINE PRINT ON EIS POPUP SELECTION MENU  */
              /*-------------------------------------------*/
  RC=INSERTC(ACTIONSL,'PRINT',-1);
  RC=SETLATTR(ACTIONSL,'ACTIVE',-1);
  RC=INSERTC(MAPL,'VIEWER',-1);
  RC=INSERTC(MAPL,'PRINT_REPORT',-1);
  CALL
SEND(_SELF_,'_SET_INSTANCE_METHOD_','PRINT_REPORT','CNALIB.CNA.P
OPUPCMD.SCL','PRINT');
              /*-------------------------------------------*/
              /* DEFINE EXIT ON EIS POPUP SELECTION MENU   */
              /*-------------------------------------------*/
  RC=INSERTC(ACTIONSL,'EXIT REPORT',-1);
  RC=SETLATTR(ACTIONSL,'ACTIVE',-1);
  RC=INSERTC(MAPL,'VIEWER',-1);
  RC=INSERTC(MAPL,'EXIT_REPORT',-1);
  CALL
SEND(_SELF_,'_SET_INSTANCE_METHOD_','EXIT_REPORT','CNALIB.CNA.PO
PUPCMD.SCL','EXIT');
RETURN;
```